2012 International Conference on Software and Computer Applications (ICSCA 2012)
IPCSIT vol. 41 (2012) © (2012) IACSIT Press, Singapore

Automated Test Cases Generation from UML Sequence Diagram

A.V.K. Shanthi[1] and G. Mohan Kumar[2]

[1]Research Scholar, Sathyabama University, Chennai, India. Email: avks_15@yahoo.com
[2] Principal, Park College of Engineering, Coimbatore, India. Email: gmohankumar68@yahoo.com

Abstract. The most widely used class of automated testing tools leads senior software testers into software development blunders that a first year programming student shouldn't make. This paper proposed a novel approach to test the software at the initial stage itself so that it will be easy for software testers to test the software in the later stages. The most common style of Automated testing is Create a test case. This paper focuses on test cases generation by means of UML Sequence diagram using Genetic Algorithm which best test cases are optimized and the test cases validated by prioritization. Lastly, the results of experiments show that this method has a better performance.

Keywords: Software Testing, Test cases, UML (Unified Modeling Language), Genetic algorithm, and Sequence Diagram.

1. Introduction

This is an exciting time to be a software developer. Software systems are becoming more challenging to build. They are playing an increasingly important role in society. People with software development skills are in demand. New methods, techniques, and tools are becoming available to support development and maintenance tasks. Because software now has such an important role in our lives both economically and socially, there is pressure for software professionals to focus on quality issues. Poor quality software that can cause loss of life or property is no longer acceptable to society. Failures can result in catastrophic losses. Conditions demand software development staffs with interest and training in the areas of software product and process quality.

All the software process improvement models that have had wide acceptance in industry are high-level models, in the sense that they focus on the software process as a whole and do not offer adequate support to evaluate and improve specific software development sub processes such as design and testing. Most software engineers would agree that testing is a vital component of a quality software process, and is one of the most challenging and costly activities carried out during software development and maintenance. The software development process has been described as a series of phases, procedures, and steps that result in the production of a software product. Embedded within the software development process are several other processes including testing. Testing can be described as a process used for revealing defects in software, and for establishing that the software has attained a specified degree of quality with respect to selected attributes.

In this paper, we use UML sequence diagrams as design specifications, and implemented the idea of automated test case generation for software as an application software testing approach, to identify the most error prone paths in a software construct. Here we have proposed a novel approach for prioritization of test cases derived from the sequence diagram using Sequence Dependence Table(SDT) and Genetic algorithm.

The paper is structured in the following way: In next section, we presented related work of sequence diagrams are used for generating test cases. In section 3 our proposed methodology for Test case generation, while section 4, and describe the experimental evaluation of the proposed model. The last section concluded the paper and gives an overview of our work.

2. Related Work

Test Case Generation by means of UML Sequence Diagrams and Labeled Transition Systems by Emanuela G. Cartaxo[5], introduced UML Sequence diagram in field of test case generation. A feature is an increment of functionality, usually with a coherent purpose that is added on top of a basic system. Feature are usually developed and tested separately from the basic system as independent modules. The procedure is based on model-based testing techniques with test cases generated from UML sequence diagrams translated into Labeled Transition Systems (LTSs).

Automatic Test Data Synthesis using UML Sequence Diagrams by Ashalatha Nayak and Debasis SamantaI[6] proposed an approach of synthesizing test data from the information embedded in model elements such as class diagrams, sequence diagrams and OCL constraints. In their approach, they enrich a sequence diagram with attribute and constraint information derived from class diagram and OCL constraints and map it onto a *structured composite graph* called SCG. The test speciations are then generated from SCG. For each test speciation, we follow a constraint solving system to generate test data.

Test Case Automate Generation from UML Sequence Diagram and OCL Expression by Li, Bao-Lin Li, Zhi-shu Qing, Li Chen, Yan-Hong[7] proposed a new test cases generation approach which based on UML sequence diagrams and Object Constraint Language (OCL). In their approach, a tree representation of sequence diagrams is constructed. Firstly traversal of the constructed tree for selecting conditional predicates from the sequence diagram is carried out. Then, post- and precondition is described by OCL. They transform the conditional predicates on the sequence diagram and apply function minimization technique to generate the test data. The generated test cases achieve message paths coverage and constraint attribute coverage of all objects which relate to the message. Lastly, the results of experiments show that this method has a better performance.

Our related works are Automated Test case from UML Diagram – Class Diagram and Activity Diagram using Data Mining Approach[1],[2]. In this paper, we proposed a novel approach to generate test cases from UML Sequence diagram using genetic algorithm.

3. Methodology for Test Cases Generation

Sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams typically are associated with use case realizations in the Logical View of the system under development. We created an sequence diagram using many software but for example we have taken the Rational Rose it's an product of an IBM. UML Model file will be saving with an extension .MDL.A sequence diagram shows, as parallel vertical lines (*lifelines*), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner. For generating the test case from the sequence diagram, we first extract the necessary information from the diagram. For that write parser in java to extract all possible information from file. Based on the extracted information, an Sequence Dependency Table (SDT) is generated. With the help of SDT test path are generated, by applying the GA most prioritized test case are generated. Steps involved in generating the test case and prioritizations are as follows:

Our methodology for test cases generation flow chart is illustrate in Fig. 1.

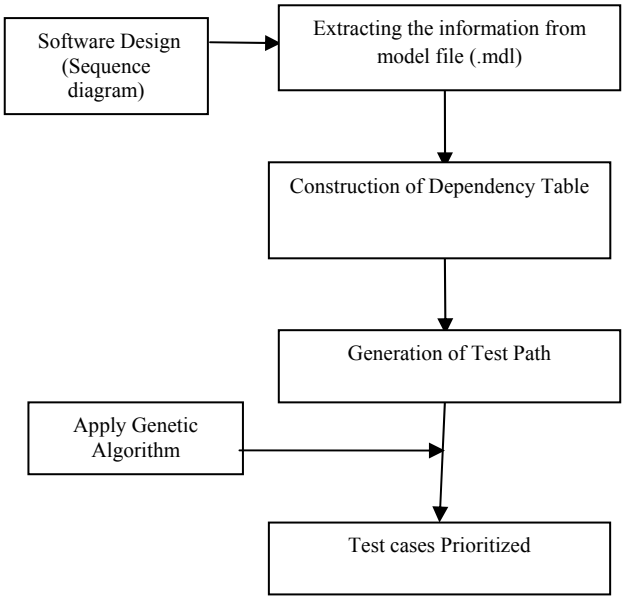

Figure 1. Flow Diagram

Table 1. Sequence Dependency Table (SDT)

Symbol	Activity Name	Sequence number	Dependency	Input	Expected output
A	Insert card	1			
B	Validate(card)	2	A	Card	True(enter the pin)
					False(end)
C	Enter pin	3	B	Pin	Pin
D	Pin	4	C	User types the pin	Receives the pin and send it to the server
E	Verify(pin)	5	D	Pin	True(enter amount)
					False(end)
F	Enter amount	6	E	Amount	Amount
G	Amount	7	F	User types the amount	Receives the amount and send it to the server
H	Check balance(amount)	8	G	Amount	True(take cash)
					False(End)
I	Take cask	9	H	Cash	End
J	End	10	I		

Algorithm for generating the test path.

1. Draw the Sequence diagram.
2. From the Sequence diagram generate the Sequence Dependency Table(SDT).
3. From the SDT, To generate the test path from the start message till all the messages are covered, repeat the following steps
   a. Store the start message in the array

b. By using the dependency the table get the next message
c. If next message is the decision message then,
    i. Copy the entire path into the next empty location and store the value.
    ii. Continue in the true side until it reaches end message.
    iii. For the false side just add the end message and continue
d. If the next message is the end message,
    i. Stop the process.
    ii. Check any other path is incomplete if its then continues with that.
    iii. Else exit
e. If the next message is the ordinary message, then add the message at the end of the current path

4. To Apply Genetic Algorithm, assigning weightage (wi) to node using the SDT table
    a. Start the process with the start message assign the value as one.
    b. Using the dependency table check the next message.
    c. Increment the value and assign it to the next.
    d. If the current message is decision, then
        i. Increment the value and assign it to the true side of the decision.
        ii. And also for the false side of the decision.
    e. Repeat the step until it reaches the end message.
5. To calculate fitness value
    a. for each node calculate the number of incoming node(a) and the number of outgoing nodes(b)
    b. $F = \sum_{i=1}^{n} (ai * bi) + wi$
6. Select the initial test data by random and calculate the fitness value for test data.
7. Generate the random number r
    a. if $r<0.8$ perform crossover
    b. else if $r<0.2$ perform mutation
8. Repeat this process until all the message paths have been covered
9. Best Test path generated.
10. End.

4. Case Study

The proposed method is evaluated by the Sequence diagram (Fig 2) of Banking System created using rational rose is used for generating automated test case from it.

Java code is being developed to extract the information of Activity diagram in Fig. 3 and also to map those information typically in abbreviated form to easy handling, as in Fig. 4.

The Possible test cases generated is shown in Fig 5.

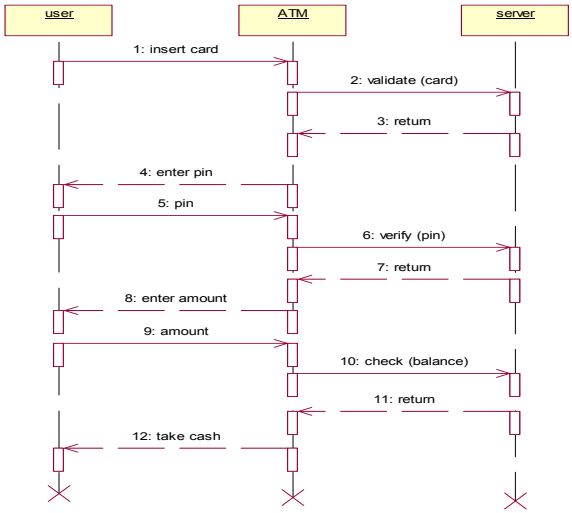

Figure 2. Sequence Diagram

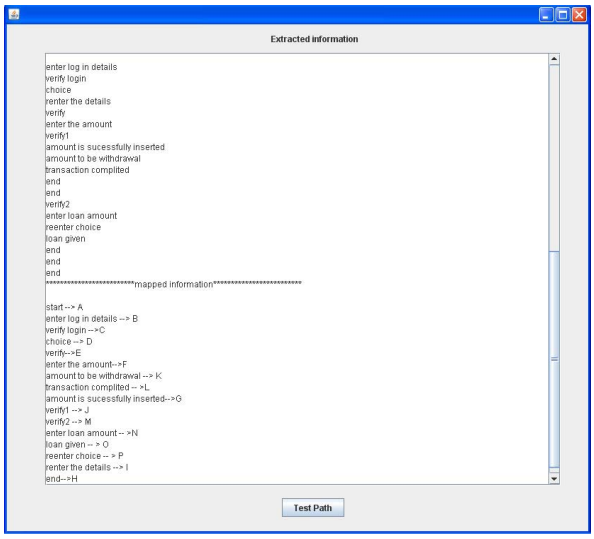

Figure. 4. Mapping Information

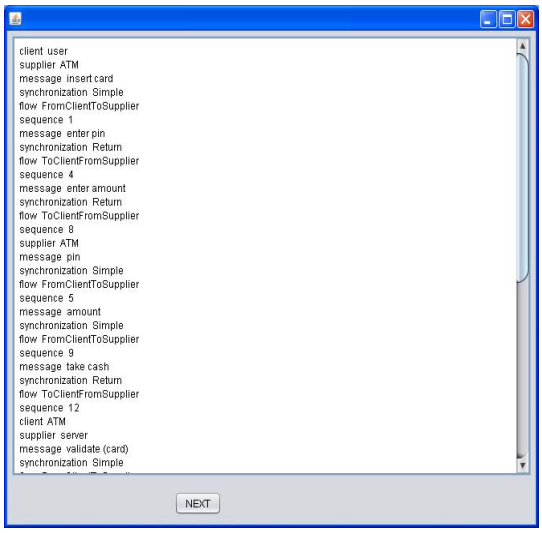

Figure. 3. Extracted Information

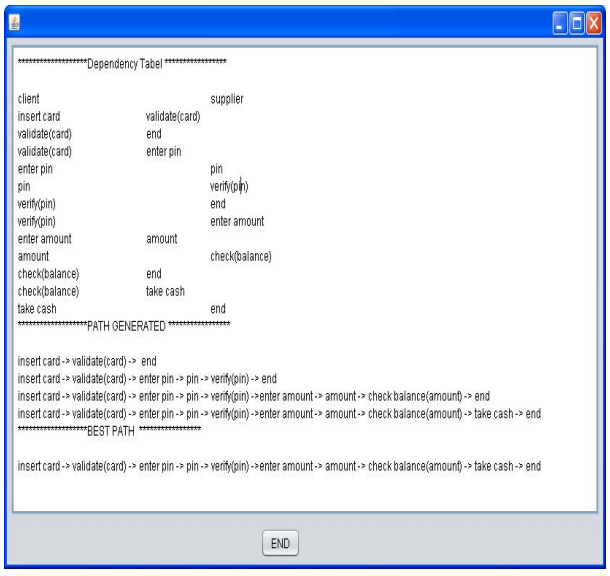

Figure 5. Sample Test Cases

Table 2 and Table 3 shows the possible and best test path generated from the sequence diagram**.**

Generated test path for the given Sequence diagram

Table 2. All Possible Test path

insert card -> validate(card) -> end
insert card -> validate(card) -> enter pin -> pin -> verify(pin) -> end
insert card -> validate(card) -> enter pin -> pin -> verify(pin) ->enter amount -> amount -> check balance(amount) -> end
insert card -> validate(card) -> enter pin -> pin -> verify(pin) ->enter amount -> amount -> check balance(amount) -> take cash -> end

Best path

Table 3. The Best Test Path

insert card -> validate(card) -> enter pin -> pin -> verify(pin) ->enter amount -> amount -> check balance(amount) -> take cash -> end

Test case generation from design specifications has the added advantage of allowing test cases to be available early in the software development cycle, thereby making test planning more effective.

5. Conclusion

In this paper we presented the test case generation by means of UML Sequence diagram using Genetic Algorithm which best test cases are optimized. Our approach is significant to identify location of a fault in the implementation, thus reducing testing effort. Moreover our novel method for test case generation inspires the developers to improve the design quality, find faults in the implementation early, and reduce software development time. In future, it is possible to build an automatic tool using this approach. This automatic tool will reduce cost of software development and improve quality of the software.

6. References

[1] A.V.K. Shanthi, Dr.G.Mohan Kumar, "Automated Test Case From UML Diagram Using Data Mining Approach", CiiT International Journal of Software Engineering and Technology, Vol3.No3, March 2011.

[2] A.V.K. Shanthi, Dr.G.Mohan Kumar, "Automated Test Cases Generation For Object Oriented Software", Indian Journal of Computer Science and Engineering, Vol:2, issue 4,Sep2011.

[3] M.Prasanna, S.N.Sivanandam, Venkatesan, R.Sundarrajan, 15, 2005,"A SURVEY ON AUTOMATIC TEST CASE GENERATION", Academic Open Internet Journal.

[4] M.Prasanna, K.R. Chandran, " Automatic Test Case Generation for UML Object diagrams using Genetic Algorithm", Int. J. Advance. Soft Comput. Appl., Vol. 1, No. 1, July 2009

[5] Emanuela G. Cartaxo, Francisco G. O. Neto and Patr´ıcia D. L. Machado, "Test Case Generation by means of UML Sequence Diagrams and Labeled Transition Systems", IEEE 2007.

[6] Ashalatha Nayak, Debasis Samanta: "Automatic Test Data Synthesis using UML Sequence Diagrams", in *Journal of Object Technology*, vol. 09, no. 2, March{April 2010, pp. 75{104,

[7] Li Bao-Lin, Li Zhi-shu, Li Qing, Chen Yan Hong ," Test Case automate Generation from UML Sequence diagram and OCL Expression", International Conference on Computational Intelligence and Security 2007, pp 1048-52.

[8] Monalisa Sarma Debasish Kundu Rajib Mall, "Automatic Test Case Generation from UML Sequence Diagrams", 15th International Conference on Advanced Computing and Communications 2007, pp 60-65.

[9] P. Samuel, R. Mall, A.K. Bothra,2008 "Automatic test case generation using unified modeling language (UML) state diagrams ", Published in IET Software.

[10] Z. Michalewicz. *Genetic algorithms + data structures = evolution programs (3rd ed.)*. Springer-Verlag, London, UK, 1996.

[11] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA, 1998.

[12] R. P. Pargas, M. J. Harrold, and R. Peck. Test-data generation using genetic algorithms. *Softw. Test., Verif. Reliab.*, 9(4):263–282, 1999.

[13] Booch G., Rumbaugh J., & Jacobson I., The Unified Modeling Language User Guide, Addison- Wesley, 1999

[14] J. Rumbaugh, I. Jacobson, G. Booch, The UnifiedModeling Language reference manual, (Addison- Wesley 1999).

[15] P. L. Navarro, D. S. Ruiz, and G. M. Perez, A Proposal for Automatic Testing of GUIs Based on Annotated Use CasesAdvances in Software Engineering, Vol. 2010, Article ID 671284, doi: 10.1155/2010/671284,2010.

[16] Kim Hyungchoul, Kang Sungwon, Baik Jongmoon,Inyoung Ko, Test Case generation from UML Activity Diagram, Eight ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallal / Distributed Computing (SNPD 2007) - Volume 03. pp 556-561. IEEE, 2007.

[17] Chen Mingsong, Qiu Xiaokang, and Li Xuandong,Automatic Test Case Generation for UML Activity Diagrams, AST'06, May 23, 2006, Shanghai, China.

[18] P. Nanda, Dr. D. P. Mohapatra and S. K. Swain, Generation of Test Scenarios Using Activity Diagram, In Proceedings of SPIT-IEEE Colloquium and International Conference, Mumbai, India,vol-4, pages 69-73, February 2008.

[19] W. Linzhang, Y. Jiesong, Y. Xiaofeng, H. Jun, L. Xuandong, and Z. Guoliang. Generating test cases from UML activity diagram based on gray-box method. In *11th Asia-Paci¯c Software Engineering Conference (APSEC04)*, pp. 284-291, 2004.

[20] C. Mingsong, Q. Xiaokang, and L. Xuandong. Automatic test case generation for UML activity diagrams. In *2006 international workshop on Automation of software test*, pp. 2-8, 2006.

[21] Sangeeta Sabharwal, Ritu Sibal and Chayanika Sharma**,** Applying Genetic Algorithm for Prioritization of Test Case Scenarios Derived from UML Diagrams, IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 3, No. 2, May 2011