ISSN: 0975–833X

Available online at http://www.journalcra.com

International Journal of Current Research
Vol. 3, Issue, 5, pp.069-074, May, 2011

INTERNATIONAL JOURNAL OF CURRENT RESEARCH

RESEARCH ARTICLE

EVALUATION OF TEST CASES IN MODEL BASED TESTING USING SESSION INITIATION PROTOCOL CASE STUDY

[1]Satyapal Reddy Regenti and [2]Rama Sree, R.J.

[1]JBICT, MCA Department, Tirupati–517 501, Andhra Pradesh
[2]Computer Science, Rashtriya Sanskrit Vidyapeetha, Tirupati–517 502, Andhra Pradesh

ARTICLE INFO

Article History:
Received 17th February, 2011
Received in revised form
15th March, 2011
Accepted 17th April, 2011
Published online 14th May 2011

Key Words:
Model Based Testing, Session Initiation Protocol, System under Test, Qtronic Modeling Language, Graphical User interface.

ABSTRACT

The Model Based Testing is one of the modern automated testing methodologies used to generate test suites automatically from the abstract behavioral or environmental models of the System Under Test (SUT). The Model based testing can be applied in different ways and it has several dimensions during implementation that can changes with nature of the SUT. With the automatic generation of test cases, requirements change is very easy to handle with the model based testing as it requires fewer changes in the models and reduces rework. It is also easy to generate a large number of test cases with full coverage criteria. A variety of CASE tools based on models are currently in use in different industries. The Qtronic tool is one generating test cases from abstract model of SUT automatically. In this research paper the detailed evaluation of the Qtronic test case generation technique, generation time, coverage criterion and quality of test cases were analyzed by modeling the Session Initiation Protocol (SIP). Also generation of test cases from models manually and by using the Qtronic Tool. In order to evaluate the Qtronic tool, detailed experiments and comparisons of manually generated test cases and test case generated by the Qtronic were conducted. The results of the case studies show the efficiency of the Qtronic over traditional manual test case generation in many aspects.

INTRODUCTION

Mark and Bruno defined the MBT as "Model-based testing is the automation of the design of black-box tests" (Utting, 2007). "black-box tests" explains the main scope of this methodology. The MBT is usually used in functional testing and it does not require understanding the internal code of the program. The MBT is a testing method which can design the test cases automatically from the specifications. Under the help of the MBT tools, test cases generation and execution time can be reduced significantly. The requirements change in the model and that helps in saving a lot of time as compare to the manual design of the test cases. One result of a case study shows that nearly 90% cost saved after the use of MBT (Clarke, 1998). "Testing is an activity performed for evaluating product quality, and for improving it, by identifying defects and problems (Abran, 2004). The MBT provides different dimensions in testing process of the SUT and it is up to the test engineer's selection, to decide which is more suitable and effective for the application that is under test. "A Taxonomy of Model Based Testing" describe Seven different dimensions of the MBT (Utting, 2000). The cost of testing always remains major concerns for the projects, and with the MBT how this cost effectiveness can be achieved also was big question. In order to determine what advantage can be achieved with the MBT in comparison with manual testing a research was conducted (Clarke, 1998) comparing the manual testing process with the MBT. In order to compare with the MBT, he used TESTMASTER tool to generated test cases automatically from the model. The results of his comparison show that the MBT have increased the productivity to 90%. Where the automatically generated test suites detected same number of failures as compared to the hand crafted model based test suites, with the same number of tests. An increase in the number of automatic test case resulted in 11% percent increase in additional defect detection. With increase in use of the MBT technique in various felids, some researches on the use of MBT in graphical user interface (GUI) testing were also conducted. In a research conducted by Qing Xie, a frame work for testing GUI of the SUT was proposed (Xie, 2006). He concluded that the MBT is feasible and have potential in applying it in the GUI testing. Other researches were conducted in the health care and smart card industries. Marlon and his group applied MBT in the healthcare system and found that the MBT can help in fulfilling the coverage of the test cases for complex healthcare systems however there are also some challenges for applying the MBT in this particular area such as the preparation of the large amount of the test data and the training of the test analysts (Vieira, 2008). Another case study of automated test generation from a formal model of a smart card application makes it possible to automatically produce both the test cases and the traceability matrix (Bouquet, 2005).

*Corresponding author: regentisatyareddy@yahoo.com

The research groups have studied "White-Box or Code-Based testing. Systems have become more sophisticated and code lines have grown incomparably than a decade ago (Abdurazik and Offutt, 2000).In software engineering, up to 50% of the total development cost is related to testing (Agrawal and Whittaker,1993). This measurement also contains the cost of debugging, software testing is still assumed to be one the most conspicuous procedures in Software Quality Assurance. It provides a series of tasks to compare, and verify the system's real and expected behavior (Apfelbaum and Doyle, 1997). Several methods have been offered to deploy MBT: Offline Generation of Executable Tests (Beizer, 1995), Offline Generation of Manually deployable Tests (Binder, 2000), and Online Testing (Booch *et al.,* 1998).Theorem Proving (Dalal *et al.,* 1998) has been basically applied for automated proving of logical expressions. In MBT approaches, the SUT can be modeled by a series of logical predicates to declare the systems functions and behavior. Symbolic Execution is mostly applied in MBT structures (Fujiwara *et al.,* 1991). It can be used to find execution traces in an abstract model. The classical ways of defining the valid system behavior is with natural language prose in the style of Requirement Specification or Functional Specification (Jorgensen, 1995).

MATERIALS AND METHODS

The next step in QML is creation of the methods which are used in the guard or action Part in the transition of model. The method declaration and definition is very similar to the Java:

```
public void Invite()
{
SIPReq r;
r.op = "INVITE";
r.param = dst;
netOut.send(r, 1.0);
}
```

The method should be defined in the class which is defined as:

```
class SIPClient extends StateMachine {}
```

Here the "extends StateMachine" used to represent that this class is combined with
the model which has the same name as the class name.

Main Method

The final part is the main method for running the QML code which is defined as this
way.

```
void main()
{
var a = new SIPClient();
a.start(); }
```

Representation of Test Case

The test cases generated from the Qtronic are abstract test cases. They only include necessary information for executing the real testing. The tester can transfer these abstract test cases to executable test scripts. If the tester has enough experience, they might not require transferring test cases and can execute the real testing directly from these abstract test cases. In the Qtronic the test cases can be represented with the Fig. 1 and Fig. 2 which display the interaction between tester and SUT or a description of the sequence of the test steps. In these case studies the manual test cases were also generated. Because, the purpose of this paper was to compare both automatic and manual test case generation method without executing them. The format of manual test cases is similar to the test cases from Qtronic and these manual test cases are also abstract.

Case Study: SIP

This case study was about the SIP. The transactions (Invite client transactions, Non-invite client transactions, Invite server transactions, Non-invite server transactions) part of the SIP protocol was modeled for the test case generation purpose. The created models have strictly followed the requirements (RFCeditor, 2009). In this case study we have followed the dimensions of MBT that behavioral model, separate test model, deterministic, with state machine notations, test case selection criteria was structural model coverage and requirements coverage.

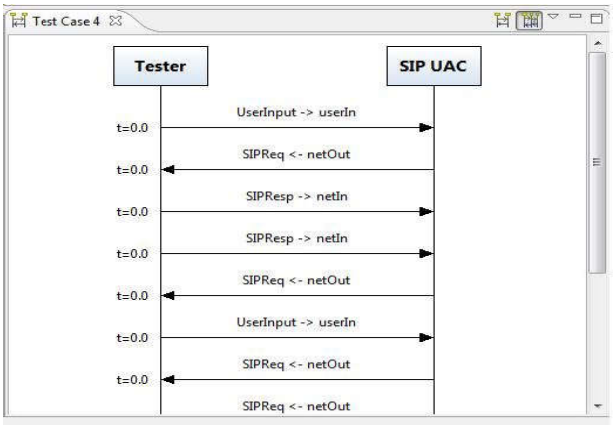

Fig 1. Qtronic Test Case Tester Interaction

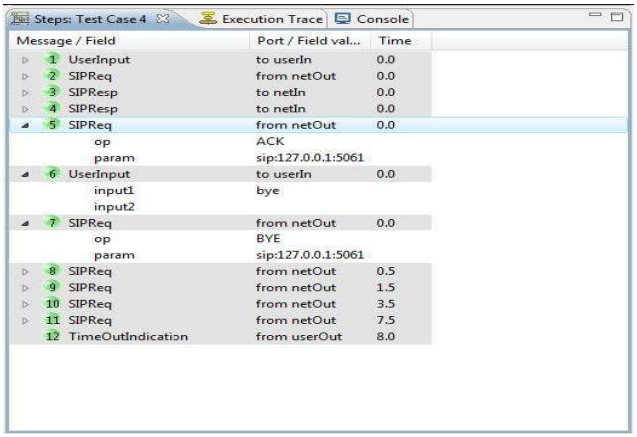

Fig. 2. Qtronic Test Case Steps

The technologies used for test case generation were manual and automatic by using Qtronic. Our main method was started from creating the models according to the SIP specifications. Then using the same model, once we generated test cases manually from the model and another time we used the Qtronic tool to generate test cases from the same model. Then we compared both manual and automatic test cases sets. Since the test selection criteria were requirement and transition coverage.

It means that the test cases should include all requirements and transitions in the model. Besides the requirement and transition coverage, we had also experiment other coverage criteria such as "2-transition", "Boundary values" etc. The results of this experiment were also included in the result section of this case study.

RESULTS

The following section describes the results of SIP case study after comparing manually produced test cases and automatically generated test cases produced by Qtronic involving time consumption, number of test cases, test case steps of both methods and also analyzed some other aspects like higher coverage criteria. The following steps includes the findings about this case study.

a) Manual test case generation took 4.5 hours and validation of these test cases took 1.5 hours, while 0.5 hours was used to fix calculation problem found in some manual test cases. Whereas in the case of Qtronic tool, the first three models (invite client model, non-invite client model and invite server model) took 0.5 hour each only in test case generation The reason was that we increased the "look ahead" depth level to the third level in order to reach the timeout requirements therefore increasing the computation time. We used the default "look ahead depth" level (the lowest) on the last model (non-invite server) and it only took 2 seconds. The Qtronic tool took 1.5 hours in total for generating the test case which was 1/4 of the Time spent on manual test case generation (Fig 3).

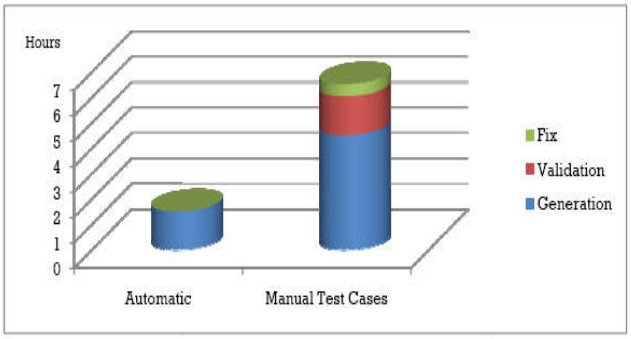

Fig. 3. Time comparison between Manual and Automatic test cases

b) In manual test case design, totally 33 test cases were created for fulfilling the requirements and transitions converge. With the same criteria, the Qtronic tool generated the exactly same number of test cases for each model (Fig 4).

Models	Qtronic Test Cases	Manual Test Cases
Invite Client	9	9
Non-Invite Client	7	7
Invite Server	8	8
Non-Invite Server	9	9
TOTAL	33	33

Fig.4. Number of test cases for SIP

c) Manual testing is good when the customers only require requirements level coverage and the model is not too complex. In this case study, the test cases generated manually are very similar to the test cases generated by the Qtronic but both of them only consider the basic requirements and transitions converge. If the customer needs to include more criteria such as the boundary value, atomic value (true, false situation) and so on it will definitely spend much more time and introduce more logic problems for manual test case generator. On the other hand, the Qtronic can do this job in one click, just required to change the coverage criteria, and the Qtronic will generate them automatically. The extra time spent can be acceptable and rarely includes logical problem if the model itself is correct.

d) In the invite client model, we included the boundary value and atomic value testing the 4 extra test cases were added by Qtronic tool. In the non invite server model, we included all criteria such as the boundary value, atomic value, control flow, two transition and implicit consumption. The result was that Qtronic added 31 test cases with only 2 seconds extra time.

e) When the tester created the manual test cases in this case study, he focused more on the exchange of messages which is the most important testing purpose. To keep the work simple and time controllable, the tester ignored some other details. On the other hand, when we checked the test cases from the Qtronic, includes more information than the manual one. The most obvious point is about the SIP message itself.

f) In manual test cases, most of steps are repeated in every test case because it's hard for manual test engineer to keep track of the transitions that are covered and uncovered. This problem resulted in testing the timeout function repeatedly as this function spends more than 32 seconds in execution every time that might cause time consuming for tester at the end. On the other hand Qtronic covers the timeout function only once and avoids the repetition.

g) During the manual test cases generation, the tester usually first decided a main stream in the model and treated other states as the leaves e.g. "state 6" (Fig.5). It resulted in increasing the complexity and steps of the test case which used to test those leaves. In case of, the Qtronic tool usually select the shortest path (Fig.5). The manual tester usually first selected state "1-2-3-4-5" as main stream and treated state 6 as leaves. When the tester, wrote the test case to test the transition between state 4 and 6, resulting test case is like "1-2-3-4-6". Where path selected by Qtronic tool includes "1-4-6" states instead.

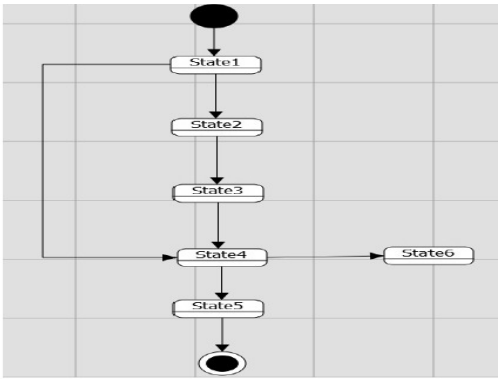

Fig.5. Example for the path selection

h) The numbers of the total test steps are reflected in the comparison points d and e. We showed that the number of test cases for both cases was exactly the same. But the result of the total test case steps of the manual one was 250 and 213 for the Qtronic tool.

DISCUSSION

In order to get familiar with the Qtronic tool, a web application called e-theater was modeled for test case generation. It was experienced that with basic knowledge of the java and state machine diagrams getting familiar with the Qtronic proved easier. It was also observed that the Qtronic tool is not much suitable for web based and GUI testing because the Qtronic was specially designed for communication domain. As "msg" keyword and time out functions shows intention of Qtronic tool towards communication sector.

Quality of test suites

The test suites are a set of test cases. As the test suites are used to test the SUT, the direct way to measure the quality of test suites is to run them and measure the number of the detected fault and the execution time. A good quality test suite should reveal more faults in a short time. However, the Qtronic and manual test suites from the case studies are not intended to be executed therefore the quality cannot be measured based on the number of faults. Instead both the Qtronic and manual test suites were generated based on the same coverage criteria. In the SIP case study requirements coverage, all state coverage and all transition coverage were selected. Because both the Qtronic and manual test suites are generated with same coverage criteria, they should generate very similar number of fault therefore with the short time consumption, less number of test cases or test steps reflects the less time of execution and the better quality in some extend. Besides more coverage criteria are also selected in Qtronic to see how many extra test cases are generated. More coverage criteria usually reveal more fault detection, if the Qtronic can apply those extra coverage criteria in acceptable time and can generate more test cases, we can say that Qtronic has ability to generate better quality test suites. Reflection of Manual and Automatic Test Case Generation: For some calculations like timer calculation in the SIP case study, manual test engineer has to calculate the timer of each message manually and needed to describe the timer with every message which is time consuming and error prone. It was very hard for the manual tester to keep track of transitions and path covered by manual test cases. The manual tester had to mark each covered requirements in the model that slow down his speed. Considering only requirements coverage in the manual test cases generation, it was lot easier than full coverage. Because of the time limitation manual tester was only considering requirements and transitions level coverage. But it is very difficult to achieve higher cover like the boundary value, full path coverage etc. manually. Since manual test engineer created hand crafted test cases, it was possible that some typing mistakes or some wrong step calculation may occurred that might have lead to a wrong test case in the end. As there was no sophisticated methods to keep track which transition is already covered or not, there were possibly repetitions of steps in some test cases that could have increased the time of testing when these test cases will be used.

We can generate test case from the Qtronic tool with one click but still it requires some time in writing QML code for the model of inbound and outbound functions. The model checking function was very helpful for checking the errors in the model and QML code before creating test cases. After modeling, the tester can select "load model" function in Qtronic to do model checking. The Qtronic tool will check both state machine diagram and QML code. The logical faults in diagram or errors in code will be displayed in a console window. If the fault is in the diagram, the console will show which transition included in this fault. If there are errors in QML code, the number of the line will be displayed with some hints or suggestions. With the Qtronic tool it is very easy to manage the requirement change problem; sometimes it was only required to change the model to fulfill new requirements even without changing QML code that saved our lot of time in this case studies.

Conclusion

This paper includes brief description of the conclusion that we have derived by analyzing the results of our case studies, reflection of test engineers, experiments and from the theory. The MBT method provides a way that involved testing in the early stages of the software development process, because requirements specifications can be directly modeled. With automatic test case generation facility, the MBT keeps greater advantages even the generated test cases are very abstract and cannot be executed directly. For instance, with automatic test cases some mathematical mistakes, wrong calculations and test coverage problems can be easily avoidable as most of the CASE tools like the Qtronic have such basic built-in facilities. The change of requirements is one of the big and painstaking problems in software development life cycle which can be easily handled in the MBT. With change in requirements only required slight changes in the models and with one more click new test suits are generated in short time. The MBT is not only facilitated with automatic generation of test cases it also have some other basic facilities like criteria selection, model checking and test case traceability matrix.

The QML similarity with Java language makes Qtronic very convenient for test engineers that require only basic knowledge about java. The Qtronic tool is targeted especially for the communication domain applications, the "timeout" function and "msg" keywords are designed especially for such areas. That might was reason that the Qtronic is not much suitable for GUI and Web testing. Like the other MBT tools, The Qtronic provides a lot of convenient functions like model checking function, different coverage selection criterion, requirements traceability matrix which proves very helpful and effective in test generation. HTML format proved very effective in test case comparison of our case studies.

The SIP case study results shows that test case generation is quick and less time consuming than manual generation because of timeout conditions and message passing involved in models which are easy to handle with the Qtronic but hard with manual process. It also shows the Qtronic efficiency of avoiding repetition of test steps, and providing power of higher coverage criteria. For future work we would like to execute generated test cases in order to analyze quality and effectiveness of test cases. In future, with better resources and

time we would like to expand the scope of our research work in order to cover other dimensions of the MBT.

REFERENCES

Mark Utting, B. L. 2007. practical model-based Testing a tools approach. San Francisco: Morgan Kaufmann.

Clarke, J. M. 1998. Automated Test Generation from a Behavioral Model. Software Quality Week. Lucent Technologies.

Alain Abran, É. d. 2004. SWEBOK. Los Alamitos, California: Angela Burgess.

Utting, Pretschner and Legeard, adapted from van Lamsweerde (Lamsweerde, 2000).

Xie, Q. 2006. Developing Cost-Effective Model-Based Techniques for GUI Testing. ICSE'06 (ss. 20–28). Shanghai: ACM.

Marlon Vieira, X. S. 2008. Applying Model-Based Testing to Healthcare Products:Preliminary Experiences. ICSE'08 (ss. 10-18). Leipzig: ACM.

F. Bouquet, E. J. 2005. Requirements Traceability in Automated Test Generation -Application to Smart Card Software Validation. ICSE'04 (ss. 15-16). St Louis, issouri: ACM.

Aynur Abdurazik and Jeff Offutt. 2000. Using UML collaboration diagrams for static checking and test generation. Proceedings of the 3rd International Conference on the Unified Modeling Language (UML 00), York, UK.

K. Agrawal and James A. Whittaker.1993. Experiences in applying statistical testing to a real-time, embedded software system. Proceedings of the Pacific Northwest Software Quality Conference.

Larry Apfelbaum and J. Doyle. Model-based testing. 1997. Proceedings of the 10th International Software Quality Week (QW 97. This paper appears in the Encyclopedia on Software Engineering (edited by J.J. Marciniak), Wiley, 2001 Ibrahim K. El-Far and James A. Whittaker: Model-Based Software Testing 1999.

Boris Beizer. 1995. Black-Box Testing: Techniques for Functional Testing of Software and Systems. Wiley.

Robert V. Binder.2000. Testing object-oriented systems. Addison-Wesley, Reading, MA, USA.

Grady Booch, James Rumbaugh, and Ivar Jacobson. 1998. The unified modeling language. Documentation Set, Version 1.3, Rational Software, Cupertino, CA, USA.

S. R. Dalal, A. Jain, N. Karunanithi, J. M. Leaton, C. M. Lott.1998. Model-based testing of a highly programmable system. Proceedings of the 1998 International Symposium on Software Reliability Engineering (ISSRE 98), pp. 174-178.

Susumu Fujiwara, Gregor V. Bochmann, Ferhat Khendek, Mokhtar Amalou, and Abderrazak Ghedamsi. 1991.Test selection based on finite state models. IEEE Transactions on Software Engineering, 17(6): 591-603.

Paul C. Jorgensen. 1995. Software Testing: A Craftman's Approach. CRC.

RFCeditor. 2009. RFC3261. Hämtat från RFC Editor: http://www.rfceditor.org/rfc/rfc3261.txt den.

****