An Improved Model for Model based Software Testing

Abdul Rauf

College of Computer and Information Sciences
Al-Imam Muhammad ibn Saud Islamic University (IMSIU)
Riyadh. Saudi Arabia

Summary
Generation of test cases is the conflicting issue in software industry. There are different methods, techniques and models are proposed to generate the test cases and minimize the test cases and cost of testing and improving the effectiveness and efficiency of the test cases. Test case generation is a method to identify the defects form the software. MBST test case generation is involved to generate the test case by the usage of models. These models can help to generate the automated test cases for SUT.
Main goal of this literature survey is to identify the models, techniques and model which are used for generation of automated test cases.

Keywords:
Software testing, Test cases, MBST, Test Case Generation.

1. Introduction

MBST requires a subset of the requirements be modeled using a formal notation like FSM, UML etc. MBST describe the Software testing in abstract terms.
MBST is a sub part of Grammar based testing which includes the string mutation using FSM, Model checking, Valid Strings and Traces are tests.
Section 2 provides a literature survey of MBST methods and models. Section 3 Provides conclusion.

2. Literature Review

Y. Yin, B. Liu, et al [1] highlight the relationship of embedded real time system testing with UML and then propose the test case generation mechanism for real time embedded system by using extended UML. The proposed method of automated test case generation of real time embedded system testing is effective, efficient and maintainable, so it also reduces the testing cost.
Researchers extend the UML Class Diagram, Sequence Diagram and Activity Diagram to fulfill the requirement of test-case model and propose the model-based test case generation for real time embedded systems testing. They verify this proposed model by implementing it on avionic system.
In embedded real time system class diagram shows the static behavior of test case generation. By analyzing the system researchers use extended class diagram in modeling of SUT by using OO methodology.
For the description of the complicated functions of the real time system used layer activity diagram in which upper layer represents the main work flow and the lower layer represents expanded activity nodes. During the test case generation search activity diagram is used to extract all independent paths for specific functions from staring node to ending node.
The sequence diagram is used to describe the dynamic behavior during test case generation.
The activity diagram is micro description of the test case generation because it cannot explain the interaction between objects. While sequence diagram is microscopic description of the test case generation because it can explain the interaction between objects.
The process of the test case generation includes the static modeling, dynamic modeling, searching interacting scenario and generation of formal description of the test case.
C. Lizhi, Z. Juan, L. Zhenyu [2] proposed the technique to generate test cases using Color Petri Net (CPN) model. Their proposed technique based on three testing coverage criteria. MBST generates test cases from high level software specification via a model in formal methods like., FSM, UML, CPN etc. But the researcher proposed approach focused on test data generation from CPN. As compare to FSM, UML, CPN provides a dynamic simulation and automated analysis of the state space.
For the generation of test cases state coverage criteria requires all states in the state space to cover the test cases.
CPN combines the strengths of ordinary Petri Net with the high-level programming language and it is often used for the analysis and simulation. The proposed techniques of test case generation all the state space of the model generating automatic states by using CPN Tool.
W. Bang, Z. Chunhua et al [3] propose the improved automated unit test frame work i-N unit. Which resolve the redundancy problem in test method based on MDA to generate unit test cases by seperating the test code and test data.
MDA based approach generate automatic test cases for the SUT on .Net Platform. The idea of i-N unit is explained by the figure.

Manuscript received January 5, 2017
Manuscript revised January 20, 2017

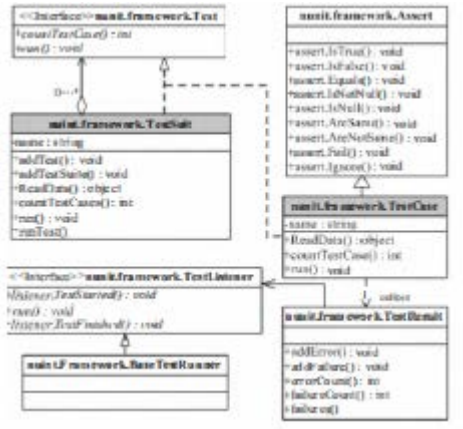

Fig. 1 Class Diagram of the i-NUnit.

The main steps of generating test cases for I-N unit are as follows:

- Build a platform –independent model of SUT.
- Transform platform- independent model in to the i-Nunit model called horizontal transformation.
- Transform the model built in step 2 in to test called vertical transformation.

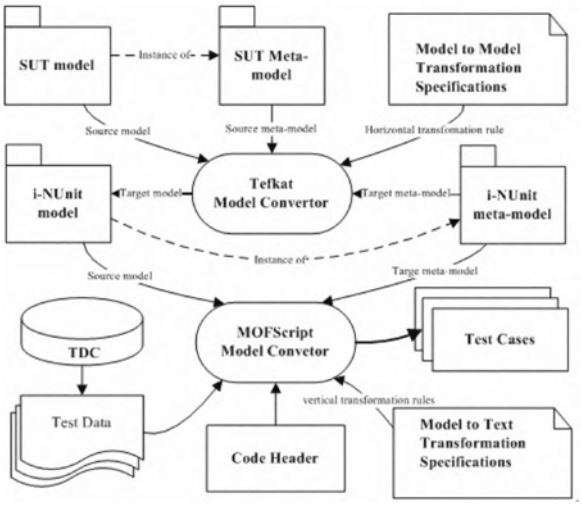

Fig. 2 Model of MDA –Based automated Test Case generation of Test cases

The TDC (Test Data Container) is added in i-N unit frame work to isolate the test data from test code. i-N unit generate test cases automatically through two layer model transformation.

P. Simon, J. Claude et al [4] propose a method and tool for automatic synthesis of test case from scenario and a state-based design model of the application by using UML framework.

The method described by the researchers is prototype tool that support the automated synthesis of UML test cases from UML Test objectives and a UML System Model. Researchers focus on the following issues which are concerning the conformance testing in a UML frame work.

- A complete process with a formal basis to synthesize the test cases from UML model.
- A formal operational semantic for UML model.
- A scenario based language with in the UML frame work to express test objective and test cases.

The proposed synthesis method has four main parts i.e.,

- Formal specification derivation.
- Formal objective derivation.
- Test synthesis on the formal method.
- UML Test case derivation.

Figure 3 represents the semantics of the UML model and the test objective.

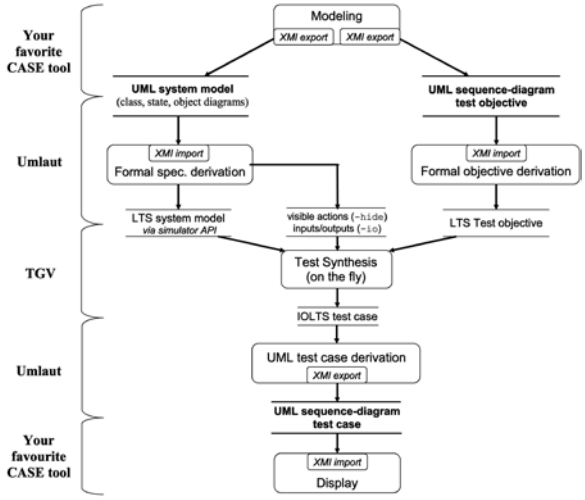

Fig. 3 The synthesis Method

For tool purpose researchers use Umlaut frame work to automatically compile UML model. TGV Tool is used to derive the automated test case generation.

M. Zhong, J. Xingan et al [5] propose Partheno-Genetic Algorithm (PGA) for test case generation problem. This paper based on the test instruction generation project of DC16Vo1 Chip and use PGA to address the instruction-set test generation problem.

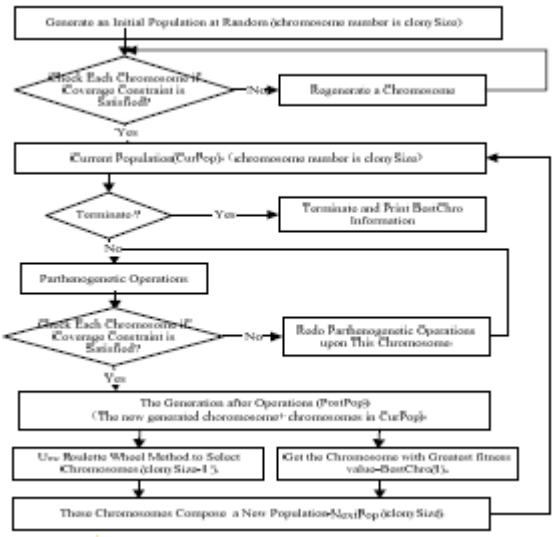

Fig. 4 Flow Chart of the algorithms (PGA)

Y. Yuan, L. Zhongjie, S. Wei [6] propose a graph search based approach to BPEL test case generation. For the representation of BPEL this approach extends the CFG & BFG. The proposed BPEL test generation method contains four steps.

- Transform BPEL to BFG.
- Transverse the BFG to generate Test Path.
- Filter in feasible test paths and generate test data for feasible test path.
- Generate abstract test case by combining Test Data and Path.

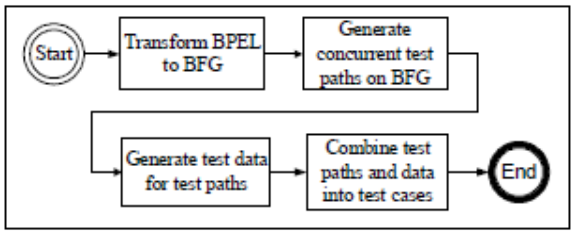

Fig. 5 Process of the proposed test generation method

When test path and test data are ready then they combined into test case automatically.

R. Matthias, H. Michael [7] propose a method of Traceability Driven Model Refinement for Test Case Generation to reduce the gap between manual technique and automatic technique.

The method of test case generation has four steps.

- Requirement Specification
- Activity diagram of the behavioral model
- Behavioral model consist of state charts
- Test cases are processed automatically

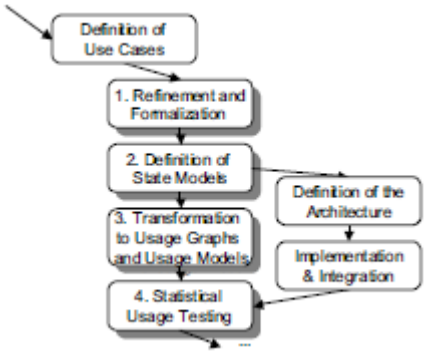

Fig. 6 Overview the Test Case Generation Process

H. Ying, C. Rong, D. Zhenjun [8] propose an automated GUI testing method for J2ME software. They adopt the automated test cases generation based on FSM. The test cases are translated from the script which drives J2ME Program under test.

For J2ME software GUI testing is more difficult because of weak language characteristics. To overcome this difficulty FSM-Based automated testing approach is used. In this approach the SUT and the specification file are taken as Input and FSM model is structured automatically.

```
Algorithm TR FSM-basedMethod (JP, Documents)
Input: JP represents the J2ME program under test;
       Documents represent some file of JP, SF is one
       file in Documents;
Output: TR represents the test report
1.  IF SF = = Null Then
2.     SF←GeneratingSF (JP, Documents);
3.  END IF;
4.  FM←GeneratingFSM (JP, SF);
5.  IF! FSMJudgment (FM) THEN
6.     ErrorShow ();
7.  END IF;
8.  ELSE THEN
9.     TS←GeneratingTestCases (FM);
10.    IP←GeneratingIP(JP, TS);
11.    runResult←Run (IP);
12.    TR←GeneratingTR (runResult, SF);
13. END ELSE;
14. RETURN TR·
```

Fig. 7 FSM based testing Method

The method for test case generation is based on WP-method in FSM-Based testing. The advantage of test case generation is general applicability and full fault-detection capability.

K. Hyungchoul, K. Sungwon et al [9] propose the method of test case generation by UML activity diagram that reduce the number of test cases generated for a particular test case. Researchers design an I/O explicit Activity Diagram Model from an ordinary UML activity diagram and then transform it into directed graph.
The directed graph is used to extract test scenarios and test cases. The IOAD explicitly shows external input and external output.
To generate test cases from IOAD model has two principles i.e.,

- Principles of black box testing
- Single stimulus principle

The test case generation procedure is:

- Drive a system of activity Diagram
- Drive IOAD model activity Diagram model
- On the basis of principles construct graph from IOAD Model
- Transverse nodes based on all path test coverage criterion
- Generate test scenarios

This method avoids the state explosion problem.
L.P. Beatriz et al [10] propose a model driven testing approach for automatic test case generation for SPL. The proposed approach is based on UML 2.0, UML Testing profile and QVT language.
The proposal of MD Testing for SPL is explained in figure8.

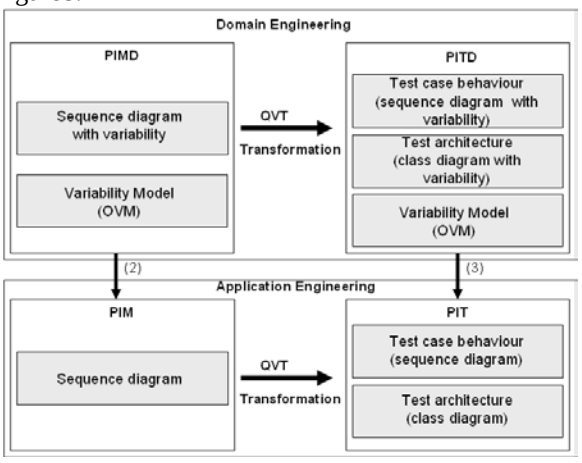

Fig. 8 Model driven testing for SPL

The approach is implemented for PIM model in domain and application engineering using QVT language.

3. Conclusion

Automated test case generation is major challenge in software industry for test case generation of SUT. Different researchers focus on Model based test case generation and they propose different testing models for test case generation. Like., UML, extended UML, class diagram for i-Nunit based on MDA test method, PGA, BPEL test case generation method etc.
All these methods and models are implemented in different environment and produce efficient and effective results in their specific environment. These models and methods are reducing the test case and effort for generating test cases.
There is need to develop a meta model for test case generation which is used to produce test cases for all type of environments and reduce the effort of test case generation and for the selecting the model for the particular environment.

References

[1] Y. Yin, B. Lin et al, "Test Cases Generation for Embedded Real-time Software Based on Extended UML", 2009, International Conference on Information Technology and Computer Science, IEEE.
[2] C. Lizhi, Z. Juan et al, "Generating Test Case Using Colored Petri Net" 2010, IEEE.
[3] W. Bing, Z. Chunhua et al, "MDA-Based Automated Generation Methoid of Test Cases and Supporting Framework", 2010, IEEE
[4] P.Simon, J. Claude et al, "Test Synthesis from UML Models of Distributed Software", 2007, IEEE Transactions on Software Engineering
[5] M.Zhong, et al "Partheno-Genetic Algorithm for test Instruction Generation", 2008, IEEE, The 9th International Conference for Young Computer Scientists
[6] Y. Yuan et al, "A Graph-Search Based Approach to BPEL4WS Test Generation", 2006, International Conference on Software Engineering Advances
[7] R. Matthias, H. Michael, "Traceability-Driven Model Refinement for Test Case Generation", 2005, IEEE, 12th International Conference and Workshops on the Engineering of Computer-Based Systems
[8] H. Ying et al, "Automated GUI Testing for J2ME Software Based in FSM", 2009, IEEE
[9] K. Hyungchoul et al, "Test Cases Generation from UML Activity Diagrams" , 2007, IEEE 8th ACIS International Conference on Software Engineering, AI, NW and Parallel/ Distributed Computing
[10] L.P. Beatriz, "Model-Driven Testing in Software Product Lines", 2009, IEEE