INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH & DEVELOPMENT

ISSN 2278 – 0211 (Online)

Model Based Testing Using UML Diagram

Simrandeep Kau
Department of CSE, CGC, Gharuan, Punjab, India
Rupinder Singh
Assistant Professor, CSE, Chandigarh University, Punjab, India

Abstract:
Software testing is essential phase of Software development. Testing of software is a process which takes 50% time of total time needed to develop software product. To reduce the time & cost effort in testing by developing the test cases at earlier level of developing model based testing is one of the famed phenomena. In this paper we have compared different graphs used in model based testing on the basis of various criteria's. Model dependency graph is one of most accurate and independent graph which tells the every relation, dependencies among the different modules of software.

Key words*: Software testing, test cases, UML based testing, control flow graph, sequence graph, criteria based graph*

1. Introduction

A software engineering task is contrived a search problem by defining a suitable candidate solution representation and a fitness function to differentiate between solution candidates. Recently software engineering has started to catch up with trend that the artefact to be optimized is often simulated system (SUT). Software Engineering is era include search based engineering which help in make optimal decision regarding test cases, model design related to simulation[13]. In today scenario, software design is used to make accurate decision by considering different factor and create balance[4].One of the main difficulties in software engineering is that the requirements of the customer are prone to change while software is being developed.

In each iteration, requirements are specified in a black box view i.e. black box testing that associates stimulus data (inputs) generates responses (outputs). All documents are subject to thorough reviews, requirements are traced through the documents [8].As in Behaviour model, Autofocus is a tool for developing graphical specifications of embedded systems those are design on basis on a simple or well defined syntax or semantics. It supports different views on the system model: structure, behaviour, interaction, and data type view. Different view include different graph which represent properties of system or model. Each model communicated through directly channel. By studying graph of given model or system various aspect come into existence such as control flow, data dependence and data flow etc [3].

As in Behaviour model, Autofocus is a tool for developing graphical specifications of embedded systems those are design on basis on a simple or well defined syntax or semantics. It supports different views on the system model: structure, behaviour, interaction, and data type view. Different view include different graph which represent properties of system or model. Each model communicated through directly channel. By studying graph of given model or system various aspect come into existence such as control flow, data dependence and data flow etc [3].

UML language is used to design the various model by using different parameter which support functional of system [11]. Unified modelling languages are standard languages for writing blueprint for design model. Autofocus components having a common global clock such that they all perform their computations simultaneously. Each clock cycle consists of two steps: firstly each component reads the values on its input ports and computes new values for local variables and output ports such that read input data and generates resultant output [4].

2. Testing

Testing is important part during development of a system. Almost 50% time of entire system which under development is devoted to testing. The more testing performed on system, high quality of software is produced. As in given figure, V-Model in which it is shown that testing is performed at each and every level. Software testing incorporates verification and validation (V&V) technique [16]. Verification and validation uses reviews, analysis and techniques to determine whether a software system and its intermediate products fulfil the expected fundamental capabilities and quality attributes [3].

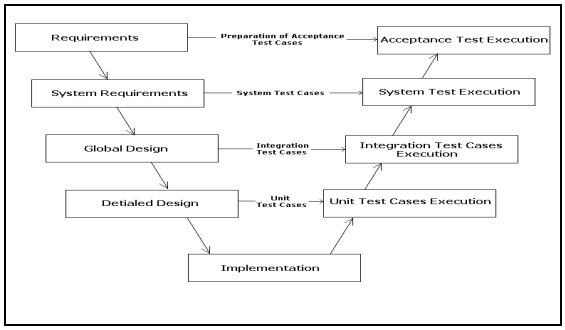

Figure 1:V-Model in Software Engineering [3]

The main objective of testing to produced software system having high performance with no error i.e. high accuracy. Different type of testing is used to perform on system at different various level such that [9]:

- **Unit testing:** This type of testing is used to perform on individual module i.e single module in order to check error in smallest part of software system.
- **Integration testing:** Integration testing is used to perform when two or module of software sytem is combined or integrates together to form larger module in order to find error or bugs [16].
- **System testing**: This testing is used in order to confirm the end to end requirement of system such that functional or non functional attribute of system [11].
- **Acceptance testing**: As name indicate acceptance usually performed by end user in order to check whether system meet all requirement which are mention in SRS document .On basis ,user will decided whether to accept or reject the system[11].
- **White box testing**: This testing relates with internal functional code of developed system in order to find error in coding [9].
- **Black box testing:** Black box testing deal with external design of module such that it deal with model design of system and find error in it[7].
- **Regression testing:** regression testing is used to perform on SUT (system under test) until system generates the exact output which required by system[7].

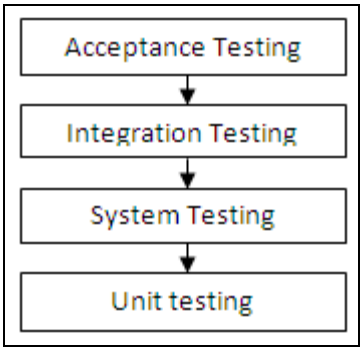

Figure 2: Level of testing [7]

Software testing is procedure which defines some different criteria under system is going to under testing in order to maximum error so that error should be removed by find various optimal solution. Testing optimization include the various criteria such that coverage, boundary graph, control, data and flow graph etc. Further, after this test case are generated by applying various criteria [15].

3. Graphs in model based testing

Its Graphical representation used to describe various objects within class or system, class to class, their relations, attributes, properties and intermediate dependencies. Graphical notation helps to understand the flow of information and control in a particular system or software.

In discrete mathematics, Graph as treated as a object and study as object. It describe the abstract view of whole system how it look like and how it going to work in real time environment. There are various types of graphs in the field of computing and computer engineering like as:

4. Control flow graph

In a control flow graph which mainly consists of basic block, i.e. a straight-line piece of code used to represent without any jumps or jump targets. Two specified block are used i.e. the entry block and exit block. The CFG is essential to many compiler optimizations and static analysis tools. Reach ability is graph property which help in optimize various decision by using different parameter [8].

Example:
0: (A) to = read num
1: (A) if to mod 2 == 0
2: (B) print to + " is even."
3: (B) Go to 5
4: (C) print to + " is odd."
5: (D) end program
We have 4 basic blocks: A from 0 to 1, B from 2 to 3, C at 4 and D at 5. In particular, in this case, A is the "entry block", D the "exit block" and lines 4 and 5 are jump targets. A graph for this fragment has edges from A to B, A to C, B to D and C to D.

5. Control Dependence Graph:
Control dependence graph used to represent the control dependencies. Vertices represent executable statements and arcs represent direct control and a distinguished entry vertex [10].
Example:
1: read i
2: if (i==1)
3: print"POS"
Else
4: i=1
5: print i
6: end

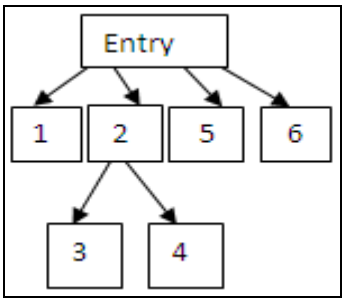

Figure 3: Control Flow Graph [10]

- If statement 2 determines whether statement 3 is executed, statement 3 is control dependent on statement 2.
- If statement 2 determines whether statement 4 is executed, statement 4 is control dependent on statement 2.

Statements that are guaranteed to execute are control dependent on entry to the program. Control flow graph can be used to control dependence graph .It keep track of each execution on program [10].

6. Program Dependency Graph
A Program Dependency Graph of a program is a graph that has nodes assigned to each statements of the program and directed edges represented dependence.
The rule is defined that an edge from a statement s_1 to another statement s_2 exists, whenever some dynamic instances, v, of s_1 shares a dependence with a later dynamic instance of s_2[2]. Typically, a PDG has two types of dependence edges: a data-dependence edge and a control-dependence edge. A data-dependence edge from s_1 to s_2 means that the computation performed in s_2 depends on the value computed in s_1. A control-dependence edge from s_1 to s_2 implies that s_2 may or may not be executed depending on the Boolean outcome of s_1, for instance, as if-statement. Consider the bubble sort algorithm on an array n[ ], as of the Algorithm 2 shown below:
Bubble-Sort (intnumbers[])
1: for i = array size - 1 **down to** 0
2: do **for** j = 1 **to** i
3: **do if** numbers [j - 1] > numbers[j]
4: **then** temp = numbers [j - 1];
5: numbers [j - 1] = numbers[j];
6: numbers[j] = temp;
On data dependent point of view, the value of i in the first for-statement 1 controls the Boolean expression in the second for-statement 2, and j controls the rest of the program from statement 3-6. From the aspect of control dependency, the execution of line 4-6 depends on the Boolean outcome of the if-statement in line 3[12].

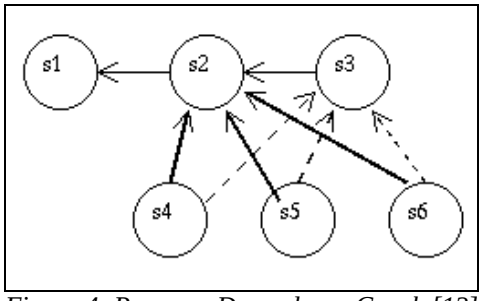

Figure 4: Program Dependency Graph [12]

Data dependence graph + Control dependence graph=Program dependence graph

6.1. Sequence Graph

A Sequence graph explain show the messages that pass between use case over time for one use case and explain the different object that take part in use case. A sequences graph shows the dynamic model that provide to evolving system in dynamic view. The sequence of message shows the external behaviour which actually shows interaction between the object. It has two element header and body i.e. header or start point in graph and body constitute the internal part of graph [1].

6.2. Coverage Graph

Coverage graph used to represent all path in order represent all path such that requires choosing test cases in such a way that all transitions of the specification are covered. The coverage criterion is the strongest path criterion [8]. This criterion is satisfied because it use represent all of all possible path in the model; the test suite contains at least one test case which enforces an execution of this path in the implementation. Path coverage is in general impossible to achieve and impractical for real life [1].

In real life, it is impossible to achieve path coverage which general impossible to describe.

Test case according to structural criteria is generated and coverage increase depend upon test cases generated [6].

Coverage graph consist of various criteria on basis of which different type of graph can be plotted such that Structure graph, Functional Criteria, Stochastic Criteria and Control Flow Oriented Coverage Criteria.

- **Structure Criteria:** In structure graph usually major contribution is test case where all transition are specified in some particular structure such that test suite consist of at least one test case which help to compel an execution for the path[6].
- **Functional Criteria:** In modelling environment, functional criteria are method of select test case. Basically such model is used as scenario model or user profile and includes the user enabled functionalities. Such that test case specified used to identify the input which used to test the implementation and also used to estimates the expected output behaviour of the implementation. Another there is possible of attack trace i.e. any user or especially an non-user attack on SUT (system under test) and control the selection process for test cases [6].
- **Statistical Criteria:** Statistical criteria derived result from analysis of the expected user behaviour or system usage, respectively.. In this case, test case selection is done randomly. In contrast, if some functions are frequently used or represent important functionalities, test cases connected to these functionalities are preferred [10].
  Statistical criteria can also be referred to as stochastic criteria. Basically from analysis of the expected user behaviour or end user behaviour or system user, statistical criteria derived resultant. All parts of the implementation, or all its functionalities have equal probability of execution is the simplest case.
  Test case connected to those functionalities that are instantly needed and used to represent the necessary information [6].
- **Control Flow Oriented Coverage Criteria:** In control flow oriented coverage criteria rely on the notions basically based on Decision and condition. A condition is an elementary Boolean expression which cannot be divided into further Boolean expressions. A decision can be seen as a control point in the specification at which the control flow can follow various paths [14].
  The decision coverage criterion also known as branch coverage requires as its outcomes consist (i.e. true or false) of every decision that is specified in produced. For example, the IF (A∧B) then, where A and B are conditions. It is required that at least one of the test case (A∧B) evaluate to true and one (A∧B) evaluate to false.
- **Model dependence graph**: Model dependence exhibit all the feature which essential for describing any particular model design .while design any s development of any software product, Modelling is important part of development cycle which usually take place before the actually coding phase start execute[2].

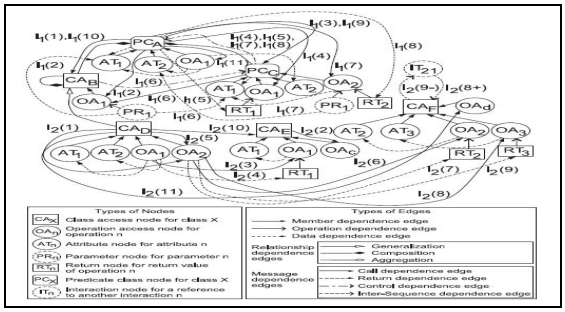

Figure 5. MDG of generic system [5]

Model dependence graph illustrated the behaviour and support model based testing to perform testing .It express all feature such analysis of different use case model ,show control flow and data flow dependence. Also MDG used to represent various criteria which exhibit the quality and quality for sequence and coverage graph. Model based graph which simply show modular design and show various path or all path[5].

Graph	Complexity	Feature	Dependence	Cost	Reliabilities
CFG	LESS	Show control flow	Show flow by connect vertices	Less	Highly Easy to use
DFG	Less	Show data flow	Show depend of travel from one vertices to another	Less	Easily but carefully
PDG	Normal	both control and data dependences	Program exhibit both properties control nad data dependence	High	Highly used
CBG	Medium	Show coverage path of model	Display all path or vertices of model	High	Highly used in testing
SBG	High	Show actuall execution sequence	Show all path or vertices and execute sequence of model	High	High in anlaysis and test case generation
MDG	Very high	Show control,data, coverage ,sequence and criteria using various parameter	All feature control flow data,sequence, coverage and criteria		Take less nad include all parameter

Figure 6: Comparison of Model based testing using various graph

7. Conclusion

To achieve the testing at early stage of software development cycle where we have only architecture of the software model based testing is best suited approach. After analysis the various graphics technique for software testing like flow, data dependency, control flow, sequence coverage, criteria base, model dependency graph we reach at the conclusion that model dependency graph is most favourable among them. It tells the various dependencies of variables or elements in single graph so that testing effort will optimize and accurate rather than consulting and comparing different graphs to achieve the result.

8. References

1. Vikas Panthi and Durga Prasad Mohapatra " Automatic Test Case Generation using Sequence Diagram" (IJAIS)-Foundation of Computer Science FCS, New York, USA Volume 2– No.4, May 2012 .
2. J.T. Lallchandani and R. Mall, "Integrated state-based dynamic slicing technique for UML models", Software, IET, vol. 4, No. 1, pp. 55–78, 2010
3. Weighhofer, M., Fraser, G. and Wotawa, F. 2009. Using coverage to automate and improve test purpose based testing. Information and Software Technology, 51, 2009, pp .1601-1617.
4. M. Harman, A. Mansouri, and Y. Zhang, "Search Based Software Engineering: A Comprehensive Analysis and Review of Trends Techniques and Applications," Technical Report TR-09-03, Dept.of Computer Science, King's College London, Apr. 2009.
5. J.T. Lallchandani and R. Mall, "Slicing UML architectural models," ACM SIGSOFT Software Engineering Notes, vol.33, No.3, pp. 1–9, 2008.
6. Christophe Gaston and Dirk Seifert "Evaluating Coverage Based Testing", 2005.
7. R. S. Pressman, "Software Engineering – A Practitioner's Approach", McGraw Hill Education Asia, 2005.
8. P. Samuel, R. Mall, and S. Sahoo, "UML Sequence Diagram Based Testing Using Slicing", IEEE Indicon 2005 Conference, pp. 176–178, 2005.

9. R. D. Craig, S. P. Jaskiel, “Systematic Software Testing”, Artech House Publishers, Boston-London, 2002.
10. F. Huber, B. Sch¨atz, and G. Einert. Consistent Graphical Specification of Distributed Systems. In Proc. FormalMethods Europe, pages 122 – 141, 1997.
11. ANSI/IEEE Standard 1008-1987, “IEEE Standard for Software Unit Testing”, pp.1-23, IEEE Computer Society, 1997.
12. M. Jackson. Software Requirements and Specifications. Addison Wesley, 1995.
13. P. Frankl and S. Weiss. An Experimental Comparison of the Effectiveness of Branch Testing and Data Flow Testing.IEEE TSE, 19(8):774–787, 1993.
14. F. Tip, “A Survey of Program Slicing Techniques”, Journal of Programming Languages, vol. 3, No.3, pp. 121-189, 1995.
15. IEEE Standard 1059-1993, “IEEE Guide for Software Verification and Validation Plans”, pp.1-87, Computer Society, 1993.
16. E. F. Miller, “Introduction to Software Testing Technology,” Tutorial: Software Testing & Validation Techniques, Second Edition, IEEE Catalogue No. EHO 180-0, pp. 4-16