Some Research Aspects on Model Based Software Testing – A Theoretic Perspective

[1]Ram Gopal Musunuru, [2]Koneru Lakshmi Sowjanya

[1]Asst Professor Department of CSE, PVPSIT, Vijayawada, Andhra Pradesh, India
[2]Asst Professor Department of ECE, DIET, Vijayawada, Andhra Pradesh, India

Abstract: Project is a collection of interrelated activities that are going to be executed in certain order, after the coding is completed the most important phase is software testing. It is the sequence of steps that can be applied over the product in order to achieve or isolate the errors. Though many of the software testing's existed like unit, integration, system testing and each are having its individual profound influences. As per the software practitioner's model is a simplification of reality. As there is a shift in the technology paradigm of today's industry people felt that model based software testing will produce better results hence Model Based Software Testing is introduced. In this paper the researchers explained some research aspects of Model Based Software Testing.

KEYWORDS: Testing, Model Based, Research aspects.

DOI of the Article: https://doi.org/10.46501/IJMTST0706059

Available online at: http://www.ijmtst.com/vol7issue06.html

As per UGC guidelines an electronic bar code is provided to seure your paper

To Cite this Article: Ram Gopal Musunuru and Koneru Lakshmi Sowjanya. Some Research Aspects on Model Based Software Testing –A Theoretic Perspective. *International Journal for Modern Trends in Science and Technology* 2021, *7*, 0706130, pp. 386-391. https://doi.org/10.46501/IJMTST0706059

Article Info. Received: 14 May 2021; Accepted: 12 June 2021; Published: 28 June 2021

INTRODUCTION:

Paradigm shift in today's software industry can be represented as

- Highly Complex
- Distributed development
- Increased use of tools and technology
- Quality driven
- Model based development and testing

In industry Software testing consumes a significant amount of effort and time of overall software development process hence Software Testing – A Challenge. Software Testing is

- Principle element of Software Quality Assurance
- Goal of testing is to expose yet undiscovered errors
- Exhaustive testing is practically infeasible
- Testing & debugging requires about 50% of total effort & time
- About 80% errors are due to 20% causes (Pareto rule)
- Testing is still a manual process
- Manual testing process intuitively guided
- Early error detection leads to lower costs
- Hence the need for effective testing strategies becomes important

In industry there is a shift from conventional software testing to object oriented software testing and issuesin Object Oriented Testing are as Object Oriented features renders more complexity in Testing and maintenance because of

- Conventional testing techniques for procedural systems are not always applicable or effective
- Design specification in UML which is visual and semi-formal, difficult to verify
- Requirements documented in natural language
- Requirements to UML model conversion is manual
- Problem of observability while testing
- Complexity in test design due to polymorphism
- Code reuse increases testing interfaces

2. Model Based Software Testing – Overview

Model-based testing is Software testing in which test cases are determined in entire or to some extent from a model that depicts a few (typically useful) parts of the framework under test (SUT).Automatic age of effective test techniques utilizing models of framework necessities and indicated usefulness

MBT – Diagrammatic view

- The model is an nonconcrete, partial production of the SUT's behaviour
- Test cases resulting from this model are purposeful tests on the same side by side of concept as the model
- Executable tests derived from Abstract test Suite

MBT – Features

- Testing based on models
- Automatic test suite generation from models
- Applicable to all phases of SDLC
- Easy to accommodate changes
- Tool support

Hence MBT promises considerable reduction of testing cost, effective test suite design and reducing the testing cycle. MBT –benefits are

- Early error detection
- Ambiguities in specification and design highlighted
- Managing models is easy
- Test mechanizationinexpensive and more in effect
- Able to adapt to requirement changes
- More test coverage than manual testing

The great Challenges for MBT are

- Models are from time to timeincomplete and not capable to depict
- overall behavior of a system
- All models may not be suitable intended for testing
- Formal models ensure correctness but very
- Complex and difficult to map to actual systems
- Semi-formal models difficult to automate
- Training and learning required to create models
- Testing and development becomes coupled

To explain the model in fully qualified manner we will consider construction domain as example

Which is shown in the below figure

Figure 2.1 Models: Construction Domain

And the internal view points for the same can be seen as in the below figure which explain the building construction building and individual sub plans

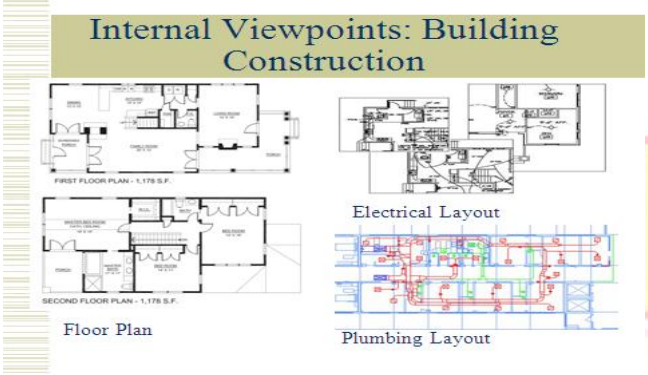

Figure 2.2 internal view points

The models in the product development cycle and building or house construction can be compared diagrammatically as

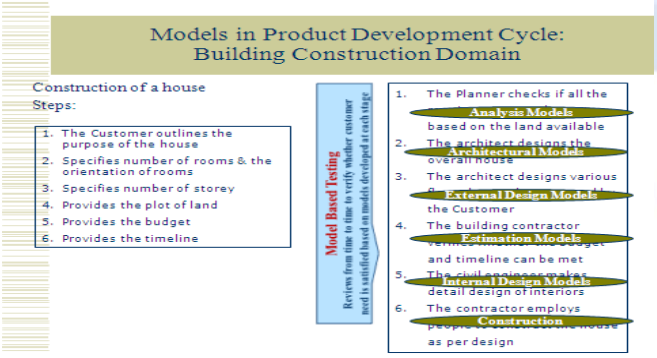

Figure 2.2 internal view points

The most recorded Benefits for modeling in Construction are

1. In depth understanding and analysis of Requirement
2. Estimation of cost, time and resources much earlier
3. Early Visualization of Solution or Result
4. Better understanding and communication with customers and people who implement the models.
5. Early clash detection among related but independent models (plumbing & electrical)
6. Improved reliability and scheduling efficiency
7. Optimum use of critical resources
8.

3. Models in Software Domain.

MBT encompasses all phases in SDLC, deals with testing based on models evolved during each phase .MBT: Constructs are

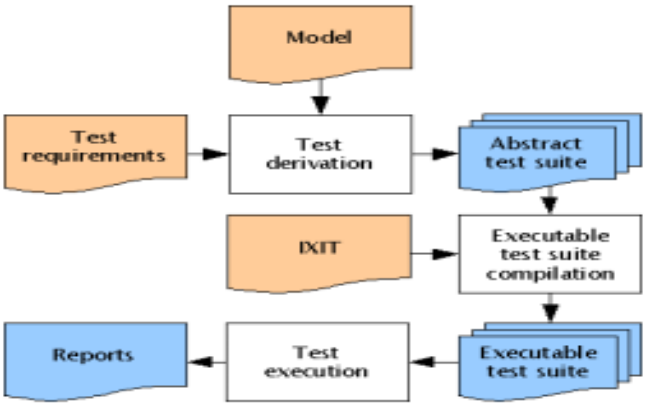

Figure 3.1 MBT: Constructs are

Here IXIT refers to "carrying out extra data" Stand for the total compendium of data that is required when the intellectual test suite is transformed into an executable one.

MBT: Activities can be represented in the diagram as

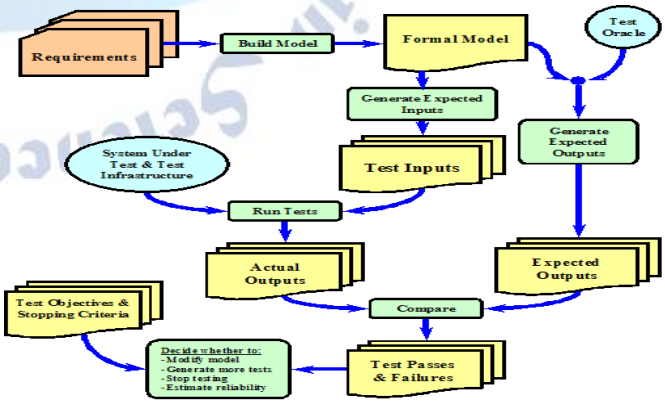

Figure 3.2 MBT: Activities

The MBT activities are

- Construct the model
- Producepredictable inputs
- Engenderpredictable outputs

- Run tests
- Compare the real outputs with probable outputs
- Resolve on supplementary actions (adapt the model, generate more tests, stop testing, estimate reliability (quality) of the software)

3.1MBT – Steps

Model Base Software Testing steps can be represented as

1. Building the model
2. Test generation algorithms from models
3. Derivation of executable test suites from Abstract test suite
4. Automatic test generation using MBT tools

Step 1: Building the Model:

Models are built during different stages in SDLC

- From specifications before the SUT is built
- Parallel to development process - design models
- From source code after the SUT is built

Different model types are used in MBT based on the application domain, ease of use some of which are: Formal, Graphical, UML, Combinatorial, etc.

Step 2: Test Generation Algorithms from Models

Different MBT techniques used for test generation from models. Techniques vary based on the type of models used .Those are

- Formal models
- Graph based models
- UML models
- State Transition based models
- Combinatorial models

Step3. Executable test suite is derived from abstract test suite

Based on models during different stages in SDLC test suite can be generated and steps are

1. From specifications before the SUT (System under Test) is built
2. Parallel to development process - design models
3. From source code after the SUT is built

4.MBT Techniques: Model based software testing is possible using Formal models,State Transition models ,Graph models,UML models and here we can notice sub individual techniques from major

4.1 Formal models:

Test Methods based on Formal models

1. Z
2. Object-Z
3. RAISE and RSL
4. VDM and VDM-SL
5. LOTOS

4.2 State Transition models

Test Methods based on State Transition Based Models

1. Markov Chain
2. Finite State Machine
3. Abstract State Machine

4.3 Graph models

Test Methods based on Graph Based Models

1. Petri-Nets, TPN
2. Data Flow Graph
3. Control Flow Graph
4. Extended Control Flow graph (ECFG)
5. Bayesian Graph
6. Distributed Scenario Graph (D-SG)

4.4 UML models

Test Methods based on UML models

1. MDE and OMG's MDA has popularized the use of modeling in Software development process
2. UML has been adopted as the standard modeling language by OMG for representing system behavior

UML models used in MBT

1. Analysis models - Use case diagrams
2. Structural models - Class and Object diagrams
3. Behavioral models – Sequence, Collaboration, Activity, State charts
4. Component diagrams

MBT techniques using UML models

1. Ensuring trace between different UML artifacts
2. Ensuring consistency between the UML models
3. Automatic generation of one model from another
4. Test generation from UML models
5. Formalization of UML to facilitate MBT

MBT techniques using combinatorialmodels, Integration of several modeling techniques for MBT

1. SOFL (**S**tructured **O**bject-oriented **F**ormal **L**anguage)
2. Combines VDM-SL, Data flow diagrams, PetriNets
3. AETG (Automated Efficient Test Generation)

5. MBT Tools:

Here we can observe the renowned testing tools for Model based Software Testing in the below table

Table 5.1 MBT Tools

MBT: Tools

Tool Name	Company	Model
AETG™ Web Service	Telcordia Technologies	Efficient Test case generation service model
ASML	Microsoft Corporation	Abstract State Machines
COW SUITE	PISATEL (Pisa Initiative on Software Architectures for Telecommunications),	UML
CONFORMIQ Test generator	Conformiq Software Ltd	UML state diagrams
GOTCHA-TCBEANS	IBM Corporation	None
JUMBL	Stacy Prowell (Ref [90])	Markov Chains
MULSAW	Research project in MIT	AAL(Alloy Annotation Language), Prioni (a tool)
PARASOFT JCONTRACT	Embedded Star™.	Simulink and Stateflow models
REACTIS	Reactis Systems Inc.	Simulink and Stateflow models

6. **Research Aspects in Model Based Testing:** Here we can observe the how research aspects in MBT based on analysis model, design model,models abstracting Code,

Testing based on analysis models is as shown in the figure.

- Formal specification of analysis models
- Ensuring trace between the Requirement and analysis models

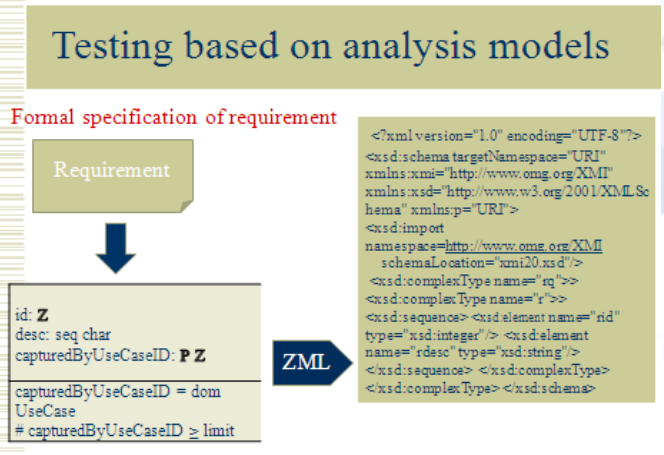

Figure6.1 Testing based on analysis models

Testing based on Design models is as shown in the figure.

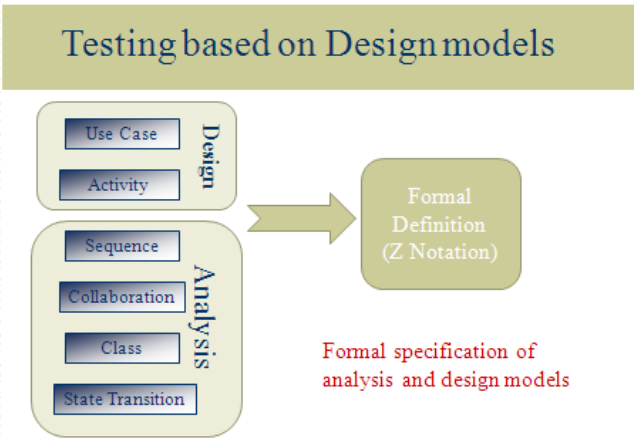

Figure6.2 Testing based on design models

- Specification of software design using formal models
- Ensuring consistency between design models
- Automatic evolution of design models
- Test path identification using design models
- Requirement based Component Design

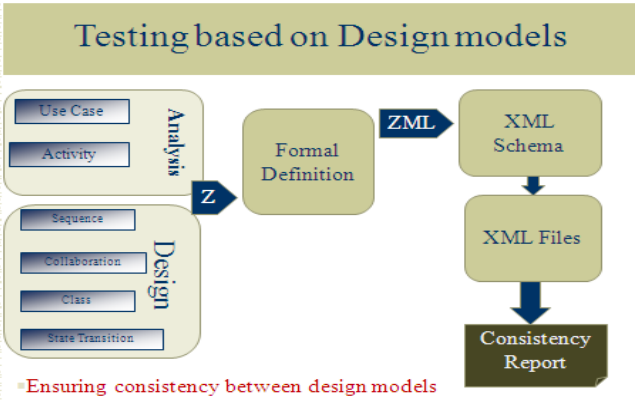

Figure6.3 Testing based on design models

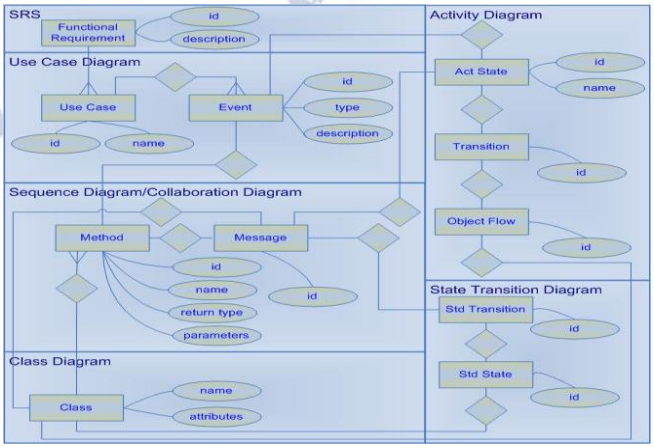

Figure 6.4 Testing based on design models

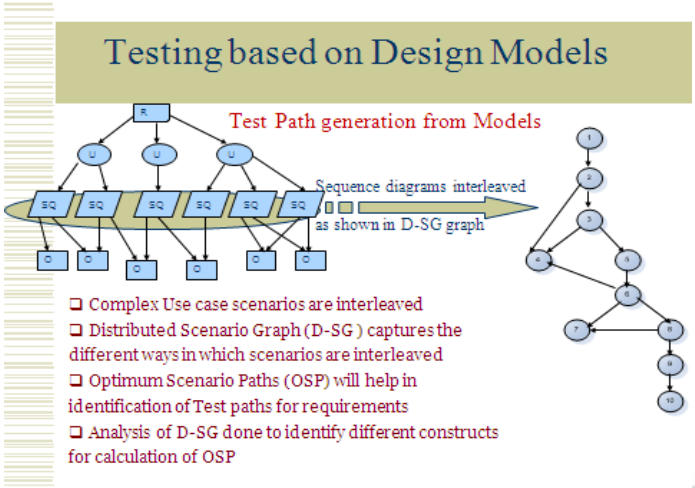

Figure 6.5 Test path generation from model

Testing based on models abstracting Code is as shown in the figure.

- Graph based analysis of object-oriented code
- Test path identification of code based on graph models
- Automatic evolution of design models (UML State chart from UML Sequence diagrams with OCL constraints) is as shown in the figure

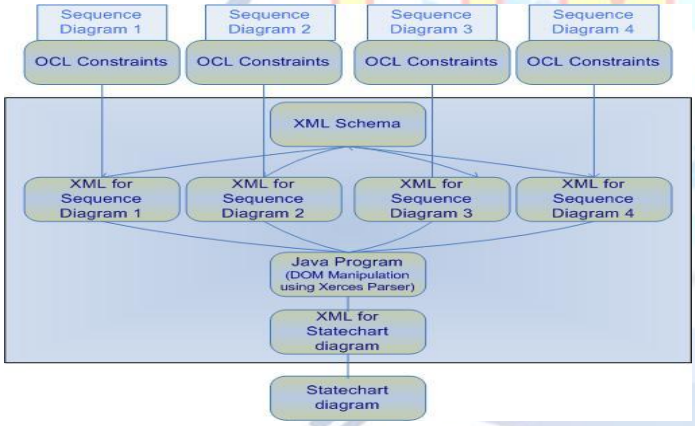

Figure 6.5 Testing based on models abstracting Code

1. SRS and UML artifacts expressed in Z notation
2. Trace rules bind together all artifacts
3. ER diagram derived from the formal representations depict the relationships between artifacts based on trace rules
4. This ensures Requirement Traceability among different stages of software development

7. Conclusion:

In this paperwe discussed about the importance of software testing, how the industry paradigm has been shifted in technological perspective, later we introduced about model based software testing need, techniques, steps, activities in detail. Here we discussed about research aspects of MBT based on analysis model, design model, models abstracting Code, Automatic evolution of design models and concluded how MBT is important in software Testing.

References

1. Roger S. Pressman, "Software Engineering; A Practitioner Approach", McGraw-Hill International Edition, Sixth Edition (2005).,ISBN 0071240837
2. Dr.K.V.K.K Prasad, "Software Testing Tools", Dreamtech, ISBN 81-7722-532-4,2008
3. Waman S Jawadekar, "Software Engineering principles and practice " The McGraw-Hill Companies,ISBN 13: 9780070583719
4. Boris Beizer, "*Software Testing Techniques*", 2nd edition, 1990 *ISBN*-10: 0442206720
5. Somerville. "Software Engineering". Addison-Wesley,1995,ISBN-0201427656.

