Model Based Testing (MBT)
TABLE OF CONTENTS

Abstract ......................................................................................................................... 3
Abbreviations ................................................................................................................ 4
Market Trends/Challenges ............................................................................................ 5
Solution .......................................................................................................................... 6
Best Practices ............................................................................................................... 9
Common Issues .......................................................................................................... 11
Conclusion .................................................................................................................... 12
Reference .................................................................................................................... 13
Author Info .................................................................................................................. 13
Abstract

Model-based testing is the application of model-based design for designing, and optionally to also execute artifacts to perform software testing.

Models can be used to represent the desired behavior of a System under Test (SUT) or to represent testing strategies and a test environment.

Therefore, MBT is a very likely way to approach any system/solution. Models can be as simple as a graph or flowchart. Typical models can be created using tools or pseudo code. The use of such pictorial representation in defining or validating a solution or system is called model-based testing.

A model alone will not be sufficient for the testing requirements, as testing requirement is the key part of testing, and that is based on the system under test for which the test cases have to be generated. A pictorial representation of a system/solution is not just used to simplify/abstract systems, but also to auto-generate test cases from the flow designed pictorially.

Model-based testing (MBT) is the next step in the evolution of software testing. MBT has been proven to allow more effective work and increase focus on the essence of testing.
## Abbreviations

<table>
<thead>
<tr>
<th>Sl. No.</th>
<th>Acronyms (Page No.)</th>
<th>Full form</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MBT</td>
<td>Model Based Testing</td>
</tr>
<tr>
<td>2</td>
<td>UML</td>
<td>Unified Modeling Language</td>
</tr>
<tr>
<td>3</td>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
<tr>
<td>4</td>
<td>SUT</td>
<td>System Under Test</td>
</tr>
<tr>
<td>5</td>
<td>FSM</td>
<td>Finite State Machine</td>
</tr>
</tbody>
</table>
Market Trends/Challenges

In the current market, software testing has a lot of challenges, both in manual as well as in automation. Generally in manual testing, testers are the middlemen between the developing team and the customers, handling the pressure from both sides for a complete coverage of scenarios. Following are a few challenges that come into the picture during software testing:

- Test complete application
- Regression testing
- Requirements understanding

Test complete application

Is it possible? I think impossible. There are millions of test combinations. It’s not possible to test each and every combination both in manual as well as in automation testing. If you try all these combinations, you will never ship the product.

Regression testing

When a project continues expanding, the regression testing work simply becomes uncontrolled. There is pressure to handle the current functionality changes, previous working functionality checks and bug tracking.

Requirements understanding

Sometimes testers are responsible for communicating with customers for understanding the requirements. What if the tester fails to understand the requirements? Will they be able to test the application properly? Definitely no!

To overcome these challenges, we are focusing on model-based testing to offer better and broader coverage of the system under test in terms of the full requirement and application coverage.
Solution

In model-based testing, models can be comprised of a form of the UML Finite State Machine chart to represent the complete flow of software system.

A Finite State Machine (FSM) is used to design the business workflow of the system/solution.

It is the behavior of the system under test and composed of states, events and transitions. Finite State machines are used widely in problem solving, artificial intelligence, circuit designs and etc. The below is a FSM depiction of an electric switch. The switch has only 2 different states - ‘ON’ and ‘OFF’.

There are two events: Switch On, Switch Off and the process of State Change are known as transition. The Switch is currently in the ON state. An event Switch off can change the state to the OFF state.

The Finite State Machine accordingly describes a collection of states and their associated transitions of the system/solution.

Finite State Diagrams

The finite state machines could also be represented using state diagrams. A state diagram is a tabular representation of the machine. The tabular format of the above state machine is as follows:
### Table 1: State table

<table>
<thead>
<tr>
<th>Current State</th>
<th>Event</th>
<th>ON</th>
<th>OFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch ON</td>
<td></td>
<td></td>
<td>ON</td>
</tr>
<tr>
<td>Switch OFF</td>
<td></td>
<td>OFF</td>
<td></td>
</tr>
</tbody>
</table>

The **switch** example above is a very simple one, depicting only two states. The switch has to be either in a switched **ON** state or switched **OFF** state. The table above shows that if the **Current State** is **ON** and if the **switch ON** activity is performed, there will be no change in the state. If the **Current State** is **ON** and if the **switch OFF** activity is performed, the state will be changed to **OFF**.

### Example

Let us take a hypothetical example. Consider a simple, web-based order processing application. This has a home page which offers content about the order application and other details. It has three links (Link1, Link2 and Link3) which take us to the Inventory page, Order page, and Payment page.

From any page (except Payment.html), the user can go to the home page (using Link8, Link9). A user on the payment page cannot navigate to the Inventory page directly. Depicted below is the state diagram for where we see four states and about nine transitions. The state diagram helps us understand the web-based system more easily. It also helps us in asking questions about new/possible transitions.

Negative test scenarios can also be tested using the State Diagram. For example, the state diagram below depicts that a user on the Payment page cannot go back to the Home page. Similarly, a user cannot go from the Payment page to the Inventory page directly.
Creation of a Model

Creation of a model is time consuming process, but this is a one-time investment. The creation of a model helps in discovering the application in more depth. This makes one think “what if” at all states. This assists in finding all imaginable cases, which in turn allows finding cracks in the requirement at a very initial phase. The cost-benefit of the discovery of a defect in the requirements phase is pretty obvious here.

Validating Models

Validation of models focuses on validating the transitions and verifying the states. The business intelligence behind state and transitions will be built into the model. The state transformations need to be verified in a proper manner so that it could represent the actual behavior of the system under test. A complete validated model will do the following.

- A Model will impersonate the system behavior based on varied inputs depending on the system
- Validate the output of Model or pictorial form

A large supposition is that the model is the replication of the system/solution. To simplify this, the model creator or modeler should have the entire/abstracted business logic built in the form of a model. If the model is not skilled at authenticating the output or producing test cases, then it is just a model and may not be of much use for testing.
Best Practices

Models are achievements in software development, as they help in accepting requirements, authorizing requirements, building designs, and managing other SDLC activities. UML is very handy to design a system/solution flow in the shape of models. The following are a few majorly identified benefits of model-based testing:

- Superlative approach to link within team
- Well-organized way of information capture
- High degree of automation practicable
- Capability to produce regression suites
- Defect discovery in initial phase

Most of the black box testers initially draw the application flow so that they can trace the functionality and have a clear picture of the application flow. MBT is very productive in testing, if these testers use the model-based approach in a more planned and systematic manner, the benefits will ensue. Some details on the benefits of model-based testing are listed below.

Superlative approach to link within team

Models are the way of keeping all team members within a development and testing group on same page. Members from both groups could refer to the models for better understanding so that all of them will have sufficient awareness of the system being developed or tested.

Well-organized way of information capture

Keeping the requirements and use cases in the form of models are more relevant and easily understandable as compared to the requirements or use cases mentioned in this document. The model represents the flow in pictorial form, thus the complex flow could easily be understandable.

High degree of automation practicable

The traditional way to design test cases became too laborious and expensive, but by using a model-based testing approach, the test case generation process is automated, or we could say it is the automation of test case generation. The generated test cases could easily be automated or we could generate executable test cases from the models.
**Capability to produce regression suites**

Once the test cases are generated from the model, those could be used for regression or system testing. Those test cases could be validated against the model for coverage and the covered requirements. Any minor or major release of the system will introduce new or updated features to the system, and those changes will have to be updated into models every time so the models will always be in sync with the system, and the regenerated test cases from models must have the updated or added functionality.

**Defect discovery in the initial phase**

In model-based testing, before going to development, the system behavior or flow is modeled which can then be reviewed so that any misinterpretation could be caught before further development. If not caught, later on it could lead to a defect in the development, and the associated cost is high.
**Common Issues**

Model-based testing also has the following challenges/common issues:

- Paucity of Awareness
- Paucity of Information
- Management Support

**Paucity of Awareness**

MBT is not very public in the testers/developer community – they generally do not have much awareness. In their view, MBT seems an overhead and typical to implement, but in practicality, MBT is helpful for testers and developers to test and develop systems with respect to coverage and requirements that are carefully modeled. Proper and effective training and seminars will help to address and improve awareness of MBT.

**Paucity of Information**

Most of the time during the modeling phase, adequate material might not exist to enable an optimal model. This can probably either delay the modeling process or result in an improper model, and the latter can prove more unsafe. Modeling should be seen as a phase where requirement/design validation happens.

**Management Support**

From a management perspective, the effort/cost associated with this extra task of modeling the system is very high. The ROI may not be clearly visible. In principal, this is an awareness challenge and most of the time management declines to make a onetime investment in creating the models.
Conclusion

Applying model-based testing on top of software testing provides a better and clearer picture of the system under test with a pictorial form of application flow representation which could easily be understandable in requirement perspective for broader coverage of test scenarios.

Practical fact observed in one of the Live Web Application (Brief Case Study on MBT effectiveness)

In the recent past, to judge the effectiveness of model-based testing, we did a pilot project on a live web application which contains many input forms with a large number of permutations and combination of values and validations.

- Manual test cases available before applying model-based testing is around 200
- After applying model-based testing, the achieved test case count is around 1,500
- Using model-based testing, the result achieved is 87% more test cases with broader coverage

Before applying the MBT approach, you’ll see that many scenarios and combinations are missing from existing manual available test cases. Due to the effectiveness of MBT, the test cases which were obtained after applying MBT had all the missing scenarios and combinations. Or we could say with broader coverage having all the possible combinations.

Tools available for model-based testing

There are many tools available today in the market which simplifies model-based testing.

- UML Pad
- MaTeLo
- Conformiq
- Graphwalker
- OSMO
- NModel from Microsoft
- Spec Explorer from Microsoft
References

http://en.wikipedia.org/wiki/Model-based_testing


I would also like to thank Ambica Jain from ERS-Practice & Manav Manchanda from ERS-SEG-SPP for guiding me in writing this whitepaper.

Author Info

Naveen Jain has been associated with HCL for more than five years out of his nine-plus years of experience in the IT industry. Over these years, he has worked with many key accounts before joining the Practice Group. His current role involves identifying ways to make life easier for testers by automating testing to the max and designing framework to achieve it most efficiently.
Hello, I'm from HCL’s Engineering and R&D Services. We enable technology-led organizations to go to market with innovative products and solutions. We partner with our customers in building world class products and creating associated solution delivery ecosystems to help bring market leadership. We develop engineering products, solutions and platforms across Aerospace and Defense, Automotive, Consumer Electronics, Software, Online, Industrial Manufacturing, Medical Devices, Networking & Telecom, Office Automation, Semiconductor and Servers & Storage for our customers.

For more details contact ers.info@hcl.com

Follow us on Twitter: http://twitter.com/hclers


Visit our website: http://www.hcltech.com/engineering-services/

About HCL

About HCL Technologies

HCL Technologies is a leading global IT services company, working with clients in the areas that impact and redefine the core of their businesses. Since its inception into the global landscape after its IPO in 1999, HCL focuses on “transformational outsourcing” underlined by innovation and value creation, and offers an integrated portfolio of services including software-led IT solutions, remote infrastructure management, engineering and R&D services and BPO. HCL leverages its extensive global offshore infrastructure and network of offices in 31 countries to provide holistic, multi-service delivery in key industry verticals including Financial Services, Manufacturing, Consumer Services, Public Services and Healthcare. HCL takes pride in its philosophy of “Employees First, Customers Second,” which empowers our 84,403 transformers to create a real value for our customers. HCL Technologies, along with its subsidiaries, has reported consolidated revenues of US$ 4.5 billion (Rs. 24,709 crores), as on TTM ended Mar 31 ‘13.

For more information, please visit www.hcltech.com

About HCL Enterprise

HCL is a $6.3 billion leading global technology and IT enterprise comprising two companies listed in India - HCL Technologies and HCL Infosystems. Founded in 1976, HCL is one of India’s original IT garage start-ups. A pioneer of modern computing, HCL is a global transformational enterprise today. Its range of offerings includes product engineering, custom and package applications, BPO, IT infrastructure services, IT hardware, systems integration, and distribution of information and communications technology (ICT) products across a wide range of focused industry verticals. The HCL team consists of over 93,000 professionals of diverse nationalities, who operate from 31 countries, including over 500 points of presence in India.

For more information, please visit www.hcl.com