CONFORMIQ

A Conformiq White Paper

What you Need to Consider when Selecting an MBT Tool

Interest in automating test design and model-based testing (MBT) has increased quite significantly over the last few years, as traditional test design approaches have started to reach their limits. At the same time, industry experts in fields such as financial services, retail, insurance, banking, telecommunications, and web-based services have started to see and understand the benefits of an automated, model-based approach to the quality assurance function and the continued relevance and success of their businesses.

The first and perhaps surprising observation that people often make when they start to look at automating test design and MBT is the variety of completely different approaches, including both academic and commercial tools. MBT can mean numerous different things, but generally speaking, MBT is based on computer-readable models that describe aspects of the system to be tested with a format and accuracy that enables fully-automatic or semi-automatic generation of test cases.

The three main approaches to MBT are 1) **graphical test modeling**, 2) **environment model-driven test generation**, and 3) **system model-driven test generation**. There are also others, but these three are the main approaches.

All these model-based testing approaches can produce the same fundamental end result – that is, they can all be used to create executable test cases and test documentation. However, this is not the main point. The key difference is efficiency - *what users need to do in order to get tests out*.

Graphical test modeling is the simplest of the approaches listed above, and consists of *modeling the test cases themselves in a graphical notation.*

Environment, use case, or **usage models** describe the *expected environment of the system under test (SUT)*. That is, these models describe how the SUT is used, and how the environment around the system operates. These models represent the tester – not the system that we are testing. The models include *testing strategies*, that is the input selection, and hand-crafted output validators, or *test oracles*.

System model-driven test generation is the third main approach to model based testing. Here, the *model represents the actual, desired behavior of the system itself*. Conformiq 360° Test Automation including Conformiq Creator™ and Conformiq Designer™ are examples of a system model-driven approach.

With both graphical and environment MBT approaches, the process of test design, (the process of deciding how to test, what to test and what not to test) can be a manual activity. These approaches to MBT rely on manual test design, so they speed up parts of the test design process but still leave a lot of work to the manual process of thinking through all the necessary test steps and combinations. This introduces a lot of risks, such as missed test coverage, and it takes a lot of time, especially when the requirements change.

As the most advanced of these MBT technologies are system-model driven approaches, we will focus on them, since they are the only approaches that are used to

actually **automate the test design**. Because there are different automated test design tools, each with sometimes subtle differences, it is useful to understand what is important when selecting a solution for automating the test design. The *fully automated* design capability clearly delivers the maximum efficiency. It doesn't need user intervention to design or select the test cases and execution results. Conformiq 360° Test Automation is unique because it delivers the most automation of the complete test oracle and thus the most efficiency of any MBT tool, with efficiency improvement a key benefit.

Modeling / tool ease of use

Since all MBT tools and methods start with a model, obviously the modeling notation and environment needs to be such that you can understand and feel comfortable working with it. Great test generation features of an MBT tool are close to useless if you cannot understand how to use them. There are many drawing and modeling tools, so it is important to select a tool that fits your needs. Some tools have overkill, with many non-modeling functions, which makes learning and using that type of tool more complicated than needed. Some contain drawing tools that don't restrict the models to constructs used for test generation; these allow users to introduce non-functional notations that will need to be changed later during the import into the test design tool.

The optimal solution is a modeling tool tailored to your targeted needs: in this case – system design for test generation. Also it is wise to learn if a third party modeling tool is required, or if one comes with the test design software. Even modest tool costs add up for larger volumes. More importantly, you would need to continually match differing releases for compatibility, putting you in the middle of two vendors to get an immediate fix when a defect is found. Ask yourself: is selecting a third party tool worth the risk and effort?

Modeling expressivity

You want to be able to rigorously express the system behavior; an MBT tool that does not allow you to do that is quite useless. For example, does the tool support multithreaded / multicomponent modeling? Hierarchical decomposition? Concepts for model reuse? Advanced arithmetic? If these system operations can't be expressed in the models, the test design tool can't cover them. Ask what are the constructs your SUT needs modeled to express its behavior when choosing a tool to automate test design.

There needs to be a careful balance between expressivity and ease of use, since as you simplify modeling notation to make it easier to understand and work with, you are inevitably sacrificing expressivity, and vice-versa. This is the reason why Conformiq offers two alternative modeling notations. The more traditional modeling notation, which is based on Java and UML, is highly expressive and can be used to meaningfully describe extremely complex systems, but requires programming skills so users need to be relatively technical. Embedded software typically needs these capabilities, which are included in Conformiq Designer. Higher-level / system-level models that do not require such extensive support for expressivity, such as ERP, enterprise applications, and web services, can be modeled using simple non-programming Conformiq Creator notation, using Structure Diagrams of the actions and Activity Diagrams of the process. This graphical modeling, using domain specific entities, makes it easy to use by testers and subject matter experts without any programming background. Both Conformiq modeling tools are designed to create system representations for testing without carrying the overhead of additional non-modeling complexity.

Generation of great quality tests

The quality of the test cases is by far the most important thing in quality assurance. If the quality of your tests is low, it really does not matter how fancy your testing processes are, or how cool the tools you are using for test execution. When you are about to automate the test design, you really need to assess the quality of the test cases that the testing automation / MBT solution produces. Remember, the quality of the output that the tool generates (test cases) cannot be of higher quality than the input (model); nor can the generated test cases be more capable than the test execution framework.

Therefore you must pay close attention to the quality and completeness of the models created.

Look for a test design tool that offers a great variety of different test generation heuristics. Relate these heuristics to your particular needs. If your system is manipulating string values, the tool should have some extensive support for expressing string patterns (such as regular expressions). If your system is performing a lot of numeric calculations, make sure that the tool has support for boundary value analysis and other equivalence class partition methods. The more test design heuristics the tool supports, the more breadth of test coverage and flexibility it delivers for testing different types of designs. You need to ask: what do you need now, and what will you need for full coverage?

Scalability

Test generation is a very computationally intensive task. The more complex the system, model and test heuristics used, the more computing resources are needed. A very critical capability is for the tool to automatically optimize the minimum number of test cases from all those possible; otherwise the computing resources needed become quite high. This optimization is key since there may be thousands of test combinations generated based on your test coverage heuristics, while you want the tool to determine the minimum number of cases to achieve the target coverage without duplication. The tool needs to automatically optimize test case combinations down to just the few unique test cases required.

Beyond just optimization is the additional need for fast test design times. Clearly this is the cornerstone of using MBT tools. In the MBT process, the model must be created correctly to match the requirements and include the positive and negative paths. If the model isn't good or complete, the test cases will not be, either. Further, you need to know if you have achieved coverage of all your test targets, and both of these needs can't be known until after test cases are generated. Thus, although it may not be immediately apparent, this means that with less advanced MBT tools, you will iterate generating test cases until you achieve your goals. If it takes 1 hour to generate tests and you need to do it 10 times to get the correct test cases, then an advanced tool such as Conformiq with 360° Test Automation can achieve the same results in 10 minutes for each iteration, and result in much better efficiency.

Test engineers may devise models that are beyond the capabilities of some MBT tools, which simply choke when given such a model. In real use, complex models are often the norm. Unfortunately scalability is something that is quite often ignored while running initial proof-of-concept projects. This means that organizations can easily invest in a tool that does not scale to real world industrial problems. With these tools, models must then be simplified, meaning reduced coverage and/or more manual effort is needed. This can be very expensive and, unfortunately the realization is often made too late that a poor investment was made. Therefore, it is important to stress the engine and the complete automation process from modeling through to test case output and even automated test execution prior to making a tool decision. Remember, all tools look fully capable for industrial use during demos and simple proofs of concept.

Conformiq has invested a significant amount of effort over the years to bring the performance of the test generation core engine to a level to manage real industrial-sized problems. A short blog about this is available on the Conformiq website, shedding light on what happens under the hood and how the Conformiq tool can be deployed on a cluster server or cloud for fast test generation. Although it may seem easy to automatically split the model across multiple computation cores, the real trick is to do it deterministically. This means that regardless of which cores and their order used, the generated test cases are always the same every time. Conformiq's test generation core is a magnitude of order stronger than any other tool on the market today, and is the only one using multicore processing.

Integrations with other tools and integration API's

The full benefits of using an advanced tool for automating test design with MBT are not realized if it is used as a standalone tool. Instead, it should be tightly integrated with other SDLC tools used to set

requirements and document, manage, and execute test cases. It is important to notice that most of the MBT tools do not actually execute the generated test cases, but instead export them into various different test execution environments. This is primarily because many MBT users have already invested in test execution infrastructure tools prior to moving to include automating the test design. Therefore, instead of replacing your existing test execution system when deploying MBT, you should look for a tool that integrates with your existing framework. This same approach applies if manual test execution is employed; you look for a tool that integrates into your way of working.

Test execution infrastructure is just one piece of the overall solution, and the MBT tool should integrate with your other SDLC tools as well: tools like requirement management, test management systems, version control systems, and so on. If there is no out-of-the-box integration with your selected tool, the MBT tool should offer integration APIs that allow it to easily integrate with your tool. Investing in a tool that does not have proper integration APIs can be risky, as that can limit your freedom to upgrade your testing infrastructure in the future and support multiple test execution platforms. To this point, Conformiq was originally architected with open interfaces to enable users to surround it with their own selected best-of-breed tooling.

Test case review and documentation

Related to the quality of the test cases is the need for the tests to be understandable and very easy to review. You should not take it at face value that tools just magically create good quality tests, and then not spend any time on reviewing them - just the opposite. The tool should allow you to understand why every test case is needed. Tool report generation capability starts from “simple things,” like generating understandable and meaningful names and high level descriptions for test cases. For example, naming test cases as “Test #1”, “Test #2”, and so on, does not help you understand what the tests cover, which then requires manual test case renaming. Additionally the report formats must be very flexible, as every project and test lead will want their own unique format.

In order to be sure that you have not missed anything in the test design, the tool must promote quality, and convince the test lead and stakeholders that the generated test suite indeed meets all the requirements. That is, you must be able to assess the coverage of the test suite. You must be able to relate the tests back to the functional requirements and also to the model, with tools for detailed analysis of each and every test case. What requirement does each test case satisfy, and what part of the model does it cover? If necessary, you must be able to walk through each test case step by step and simulate it against the model to gain full understanding of the tests.

Conformiq 360° Test Automation delivers unsurpassed capabilities and flexibility for automatically documenting test cases, with auto-generated meaningful unique names and detailed test descriptions for investigating tests.

Model analysis and debugging

Since system models are developed by humans, they may contain errors and omissions. A model that performs arithmetic can, for example, perform a division by zero, while a concurrent model can have model-level thread scheduling that causes the threads to deadlock. The bigger and more complex models get, the more important it is for the MBT tool to provide different means for analyzing potential issues in the model itself, simulating the model in order to gain an understanding, identifying erroneous scenarios and missing coverage, and then linking any failed test execution results back to the model, for better understanding of the root cause of a failed test case. These are important capabilities that often go unnoticed during the very first proof-of-concept pilots. Again, investing in a technology that does not support full-fledged model analysis can in real use deliver much less than anticipated efficiency gains.

Conformiq Designer, for example, while performing test generation, verifies that the model is internally consistent, i.e., the tool checks for the absence of internal computation errors (such as division by zero). If the model happens to contain an internal error, Conformiq Designer will produce a comprehensive report

that details the circumstances under which the problem occurred, graphically pin-pointing the problematic location in the model, and show a full execution trace to the problem. For further analysis of the problem, you can start a “model debugger” which is an infrastructure that allows you to analyze various issues in the model and get a better understanding of the automatically designed and generated test cases. Conformiq Creator has a “Live Check” capability that notifies you of the remaining information needed to complete the model and a graphical debugger to help fix model defects. With these tools, model debugging and analysis is much faster and less error prone, which streamlines the whole modeling / test generation process. Debugging real world models can be the most time consuming part of deploying MBT. This becomes increasingly important if MBT is used in an agile development process.

The whole premise of advanced MBT automatic test case generation is that the model is complete and correct. Bad models give bad results. The effort in creating a good model is the hidden “cost” of deploying poor solutions for MBT. Therefore, it is very important to choose a solution that allows you to quickly create good models and understand them. So, be sure to look for a solution like Conformiq that offers fluent model debugging and analysis capabilities.

Support and user community

System model driven automated test design and MBT requires a different skill set than traditional manual testing, Testers must fully understand how the high-level system works, not just think about some use cases. Also, working with the models requires a different mindset than traditional testing, so some test engineers may feel slightly intimidated by these tools, but ease of use can conquer this.

Many best practices have been collected over the years, with documentation available on how to address the issues associated with this testing transformation. Conformiq can not only provide a state-of-the-art testing solution, but also training, documentation, and best practices around these and other questions both before and during deployment. Best practices include topics ranging from how the new testing process affects project staffing, to how models should be created to enable reusability on other projects.

Overall, sales Proofs of Concepts (PoCs) are intended to do exactly what they say – Prove the Concept. They do not prove the capability for use on real programs. While full testing cannot be done as a PoC, you can and should select PoCs that expose the positives and negatives for all the test automation MBT tools you evaluate. Make the PoC really meaningful to you, run multiple PoCs to demonstrate tool differences, and be sure a specific tool will work for you. Although difficult to determine the real engine differences, that understanding is THE most critical learning to ensure successful adoption. And please contact Conformiq to run a POC with us!

Author Kimmo Nupponen, Chief Scientist at Conformiq, has been developing automated test design software for over ten years. He understands what is really needed for real world use and the important differences between tools, especially in their engines.

www.conformiq.com

USA
4030 Moorpark Ave
San Jose, CA 95117
Tel: +1 408 898 2140
Fax: +1 408 725 8405

FINLAND
Westendintie 1
02160 Espoo
Tel: +358 10 286 6300
Fax: +358 10 286 6309

SWEDEN
Stureplan 4C
SE-11435 Stockholm
Tel: +46 852 500 222
Fax: +358 10 286 6309

GERMANY
Maximilianstrasse 35
80539 Munich
Tel: +49 89 89 659 275
Fax: +358 10 286 6309

INDIA
29 M.G. Road Ste 504
Bangalore 560 001
Tel: +91 80 4155 0994