ICT GROUP Improve Quality Services

WHITEPAPER

Faster and better testing with **Model-Based Testing**, however... **Be careful** when you select your test cases

Around the year 2010 Model-Based Testing (MBT) was at the top of the hype cycle. It was seen as a must-have. The method could be implemented with relative ease and required only limited investment. Then a range of disillusions followed. It turned out to be more complex than anticipated and the costs often outweighed the benefits. This was the reason for most organisations to stop using MBT. Which is a shame, as experience proves that MBT can definitely contribute to productivity when used in the right situations. What are these situations? And what are important factors to take into account when you are considering applying MBT?

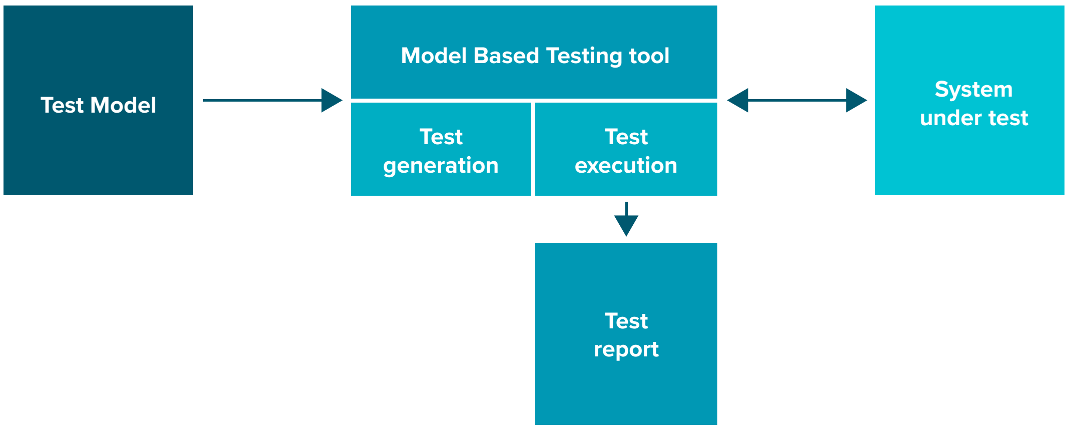
1. BENEFITS OF MODEL-BASED TESTING

MBT is a type of test automation. As with Model Driven Engineering, MBT uses a model as its foundation. This model serves two purposes:

1. It facilitates communications with stakeholders, because a model is far easier to comprehend than code.
2. The tool automatically generates the test scripts from the model and executes them. This means that you do not need to write test scripts yourself.

When using MBT you design a model based on the prerequisites you impose on a test.
These prerequisites are related to issues such as:

- Which component are we testing exactly?
- What are the risks that we want to remove or reduce?



The benefits of MBT can be observed at three levels:

1. Reduction of time and cost involved
2. Quality improvement
3. Improved communications

1.1 Reduction of time and cost involved

As is the case for other kinds of test automation, the main benefit of MBT lies in automatically performing test cases, which can lead to huge savings in terms of time. This allows you to run test cases faster, or perform more test cases in the same amount of time. During the early years people were also expecting to save time by automatically generating a test case from the model. It certainly holds true in some situations, but occasionally creating a model takes more time than programming, as will be discussed in more detail further on.

Another significant benefit of MBT is that in the case of changes, you only need to change the model and not the actual test scripts. After all, these will be automatically generated by the model, which means that tests are easier to maintain and therefore their maintenance requires lower cost.

1.2 Quality improvement

MBT contributes to quality improvement in several ways. First of all, it enables you to increase the test coverage, because the automatic execution provides more testing coverage in comparison with manual testing. In some cases, the models can become so complex that they cannot be managed manually. MBT can help you to manage this complexity. As a result, MBT helps you achieve higher quality in complex situations.

In conclusion, you can already create test models at an early stage of the project and thus find errors in the software early on. A relevant example is a project for testing a communications protocol. During the development of the test model it became apparent that the description of the protocol was incomplete and inconsistent. Even before the test model was completed the first error surfaced. This is due to the structured way of thinking that is required while developing a model.

1.3 Improved communications

Working with models improves communications with stakeholders because everyone is looking at the same model and the model is clear for everyone involved. Take for example the testing of software for an x-ray machine. The domain experts know which software risks need to be tested, but they are not familiar with software testing. By developing the test model together with domain experts, they see exactly how the test proceeds and they can indicate whether the right aspects are actually being tested.

Why is MBT used so rarely?

If the benefits are so varied, then why is MBT used so rarely? After all, MBT has been experimented with extensively during the past decade, but the number of successful applications remained relatively low. There are several reasons for this. The most important reasons are:

- Test teams, DevOps teams or project managers defined too many objectives for MBT.
- Models are not being applied at the right level of abstraction.
- MBT is being applied in domains that offer very limited benefits.
- Insufficient availability of mature tooling.

The following sections will describe these reasons in more detail and discuss how you can increase the success rate.

2. REACH PRIOR AGREEMENT ABOUT YOUR OBJECTIVES

As described in subsection 1, MBT offers several advantages. This is why different stakeholders often have differing expectations.

- The tester expects higher quality from using MBT.
- The project manager expects that testing will take up less time.
- The test analyst primarily wants to use the models as a means for communicating with users or business analysts, with the objective of avoiding confusion.

In practice one objective often stands in the way of the other objectives. For example, if you want to achieve higher quality, you will have to make the models more detailed than the existing test cases. Making the models takes time, so the overall testing time does not decrease; at least not at the first iteration. In this scenario the project manager will bail out disappointedly, because he expected to save time. On the other hand, if your entire focus is on saving time, then you are likely to create simpler test models, which means that there will be no additional quality boost. In that case if it has not been communicated clearly, the tester will be disappointed as he expected to achieve higher quality.

It is of crucial importance to reach agreement about the objective for which MBT is being used. Is the focus on increasing the test coverage, improving efficiency, or is it more important to start the tests earlier on during the design phase? Or is MBT primarily used because the test cases can no longer be managed manually? Careful prior decisions about the objective and agreement on the objective within the team will substantially increase the chance of success.

A good example is a project in which saving time was the main objective. A model from financial legislation was already available that could be re-used for the most part. Until that time, the rules of the model were verified manually. Entering this model into an MBT-tool allowed the project to achieve prompt efficiency benefits.

During another project MBT was used for the single objective of increasing quality. A model was created to imitate two parallel applications. The MBT tool provided features for simulating the timing, which allowed the tool to find any deadlocks or other errors. These errors would never have been noticed with manual testing.

3. CREATE MODELS WITH THE RIGHT LEVEL OF ABSTRACTION

Many people agree on it: modelling is difficult. One of the reasons is the fact that creating models with an appropriate level of abstraction requires a substantial amount of knowledge and expertise. Unfortunately very few testers have learned how to create good models. Although the models are sufficient for manual derivation of test cases (if they are created at all), the complexity of these models – or rather the lack of complexity – disqualifies them for representing the complexity of an actual system.

As an example we will use a model that was used during a project for developing a system for controlling bridges. The model represents the process of operating a bridge deck.

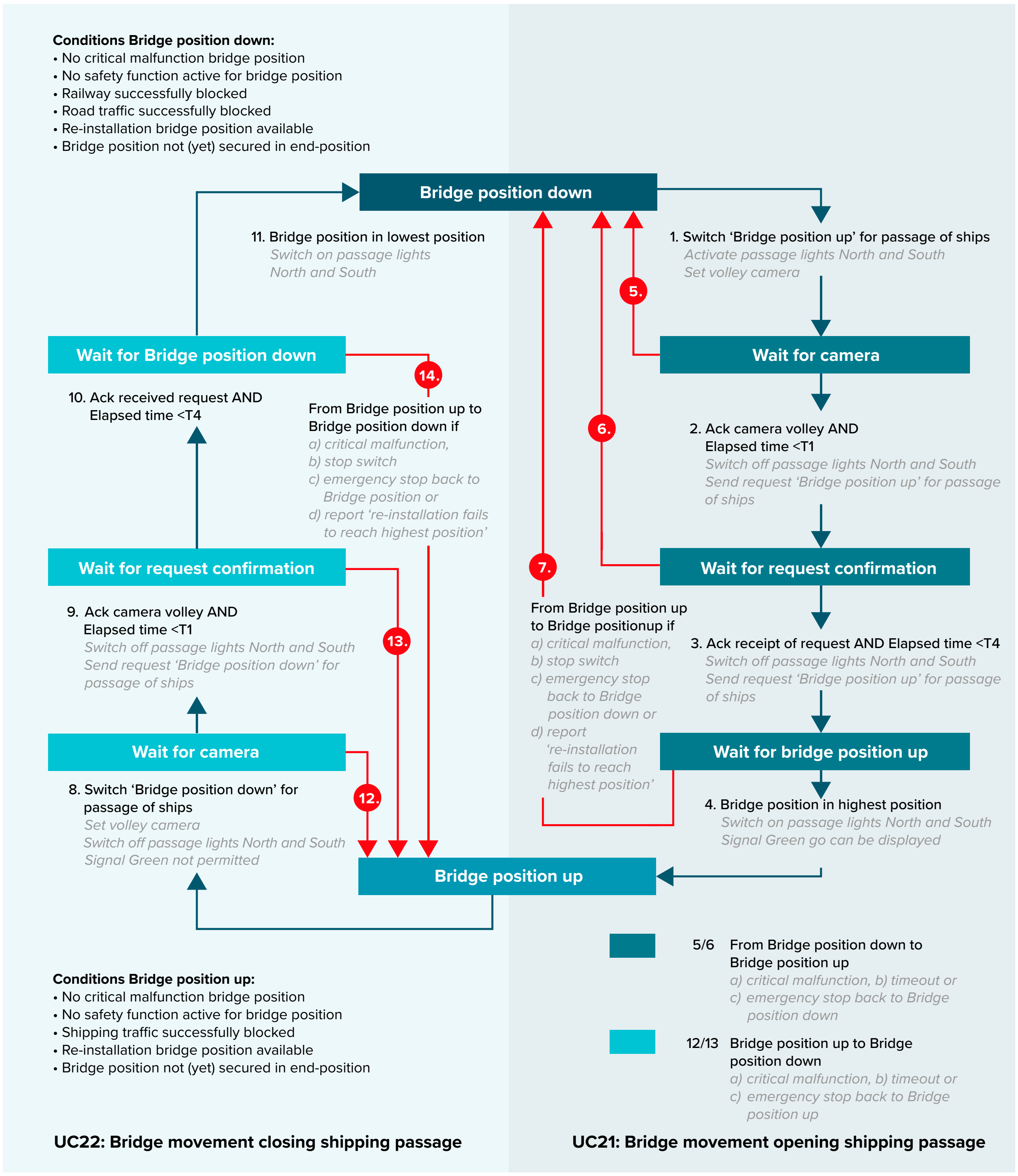

Conditions Bridge position down:
• No critical malfunction bridge position
• No safety function active for bridge position
• Railway successfully blocked
• Road traffic successfully blocked
• Re-installation bridge position available
• Bridge position not (yet) secured in end-position
Bridge position down
11. Bridge position in lowest position
Switch on passage lights North and South
1. Switch 'Bridge position up' for passage of ships
Activate passage lights North and South
Set volley camera
5.
Wait for Bridge position down
14.
Wait for camera
10. Ack received request AND Elapsed time <T4
From Bridge position up to Bridge position down if
a) critical malfunction,
b) stop switch
c) emergency stop back to Bridge position or
d) report 're-installation fails to reach highest position'
6.
2. Ack camera volley AND Elapsed time <T1
Switch off passage lights North and South
Send request 'Bridge position up' for passage of ships
Wait for request confirmation
7.
Wait for request confirmation
9. Ack camera volley AND Elapsed time <T1
Switch off passage lights North and South
Send request 'Bridge position down' for passage of ships
13.
From Bridge position up to Bridge positionup if
a) critical malfunction,
b) stop switch
c) emergency stop back to Bridge position down or
d) report 're-installation fails to reach highest position'
3. Ack receipt of request AND Elapsed time <T4
Switch off passage lights North and South
Send request 'Bridge position up' for passage of ships
Wait for camera
Wait for bridge position up
8. Switch 'Bridge position down' for passage of ships
Set volley camera
Switch off passage lights North and South
Signal Green not permitted
12.
4. Bridge position in highest position
Switch on passage lights North and South
Signal Green go can be displayed
Bridge position up
5/6 From Bridge position down to Bridge position up
a) critical malfunction, b) timeout or
c) emergency stop back to Bridge position down
12/13 Bridge position up to Bridge position down
a) critical malfunction, b) timeout or
c) emergency stop back to Bridge position up
Conditions Bridge position up:
• No critical malfunction bridge position
• No safety function active for bridge position
• Shipping traffic successfully blocked
• Re-installation bridge position available
• Bridge position not (yet) secured in end-position
UC22: Bridge movement closing shipping passage
UC21: Bridge movement opening shipping passage

Figure 1, Model for operating a bridge deck

Due to the many conditions and possible paths, this model seems rather complex at first glance, but an actual real life system is a lot more complex. The level of abstraction in this model is simply too high. You can further elaborate the model, but this effort does not outperform manual generation of test cases. After all, making these complex models will take a lot of time (i.e., several months). In addition to the fact that the structure of the models is complex, a large amount of detailed data needs to be entered to make them suitable for applying them for MBT. In those cases, creating the model and the costs involved does not outweigh the benefits.

Be alert of looping formation

Figure 2 shows another aspect of models, i.e. the fact that loops in models can in theory lead to creating an infinite number of test cases. If that is combined with an infinite amount of input data, this is called a test case explosion. This is why you cannot just make a model and then run it. You need to consider that these kinds of loops may be present while you preconfigure the tool.

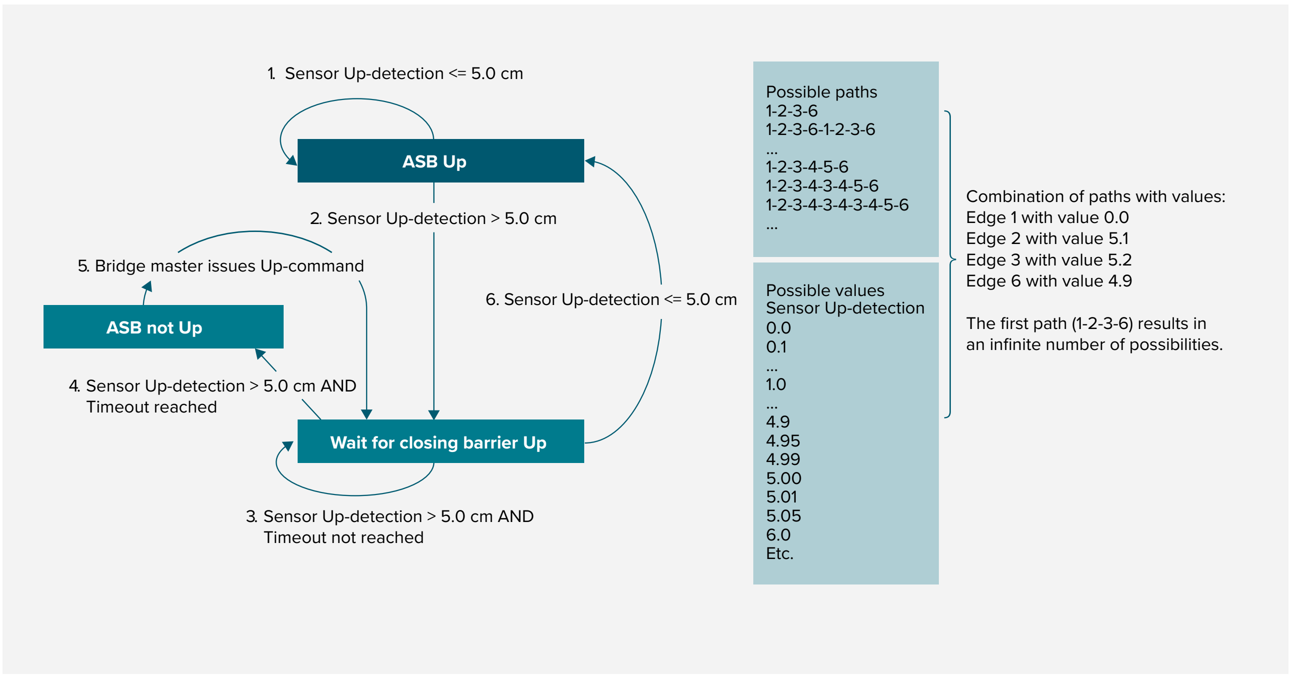

1. Sensor Up-detection <= 5.0 cm
ASB Up
2. Sensor Up-detection > 5.0 cm
5. Bridge master issues Up-command
ASB not Up
6. Sensor Up-detection <= 5.0 cm
4. Sensor Up-detection > 5.0 cm AND Timeout reached
Wait for closing barrier Up
3. Sensor Up-detection > 5.0 cm AND Timeout not reached
Possible paths
1-2-3-6
1-2-3-6-1-2-3-6
...
1-2-3-4-5-6
1-2-3-4-3-4-5-6
1-2-3-4-3-4-3-4-5-6
...
Possible values Sensor Up-detection
0.0
0.1
...
1.0
...
4.9
4.95
4.99
5.00
5.01
5.05
6.0
Etc.
Combination of paths with values:
Edge 1 with value 0.0
Edge 2 with value 5.1
Edge 3 with value 5.2
Edge 6 with value 4.9
The first path (1-2-3-6) results in an infinite number of possibilities.

Figure 2, Model of Up-detection of a barrier

Train testers to model properly

Proper modelling requires the necessary knowledge. Due to the popularity of Model Based Engineering, designers do learn how to model, but it is not a basic skill for testers. This is strange because a lot of models are used as test design techniques.

Models for testing have different focus areas than models for designing, which means that often it is not sufficient for passing on knowledge inside a DevOps-team. This is why there are so few testers that really master the skill.

Give the model a clear focus

When you create models it is important that the model has a clear focus. If this focus is missing, usually a lot of details are added to the model, which results in the model becoming too complex. In this case it is better to create several models, with a different focus for each, to keep each model readable and useable.

Take account of the modelling expertise of testers

When you determine the abstraction level of the model, make sure that you take account of the testers' available expertise and experience with modelling. As indicated above, the models are often created with an abstraction level that is either too high or too low. If the abstraction level is too high, the test does not add value, while risks in the software will remain unnoticed. If the abstraction level is too low, it takes too much time – and money – to create the model, which undermines the business case.

4. SELECT THE RIGHT AREA OF APPLICATION

MBT is not suitable for every situation. As was indicated above, in most situations the costs for developing models are simply too high and will never be earned back. This is why it is important to select the right areas of a project in which MBT will add value. This is the case when there are high risks when the software is deployed and when specific aspects are at stake, such as the reliability and safety of a system.

High product risks

If a project involves a high risk factor – due to a high chance of failing, high costs of failing, or both – you can opt to use MBT. In this case the costs of creating the model soon outweigh the benefits. However, if the risk factor is low, this is often not the case. This is why you not should use MBT for every aspect of your project, but particularly in those areas that need extra attention due to the higher risk.

Testing non-functionals

Research performed in 2019 by Anne Kramer, Bruno Legeard (Model-Based Testing User Survey: Results) has shown that 93 percent of the users applied MBT for functional testing while only 21 percent used it for testing the non-functionals, such as performance, parallelism and timing. Still, MBT is well suited to test these non-functionals.

Some MBT-tools allow *randomisation*, which allows you to test a large number of paths that cannot or are difficult to test manually. *Parallelism* can also be adequately tested with MBT. In this case the tool is used to follow several paths using multiple processes in order to find issues like dead-lock. MBT also offers distinct advantages for *timing issues*. Timing issues are hard to test manually, but the right MBT-tool allows you to test with different timings in order to find out whether this leads to problems. In that case the randomisation is used for the timing.

Practical example of testing parallelism and timing

In 'Model-Based Testing with TorXakis – The Mysteries of Dropbox Revisited' Jan Tretmans and Piërre van de Laar describe how to test parallelism and timing. Figure 3 shows the graphical representation used by the authors. The model shows the way Dropbox operates when files are read and written from multiple computers. They used a specific MBT-tool that can determine by itself how to use randomisation for executing several paths using multiple processes.

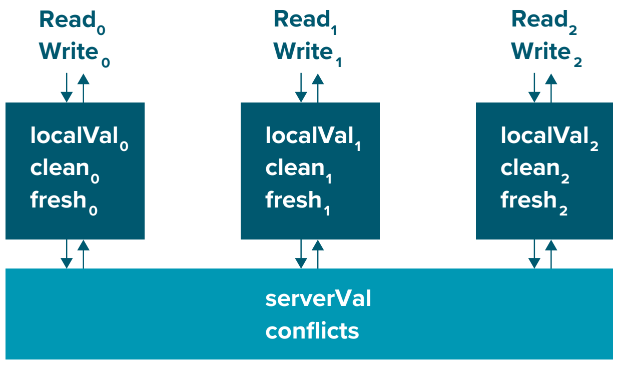

Read_0
Write_0
Read_1
Write_1
Read_2
Write_2
localVal_0 clean_0 fresh_0
localVal_1 clean_1 fresh_1
localVal_2 clean_2 fresh_2
serverVal
conflicts

Figure 3, Model for Dropbox

5. TOOLING

An important reason why MBT is not applied properly is that the tooling has not yet reached a sufficient level of maturity. Even though there is plenty of available tooling for usage in different modelling techniques, it is still very limited. The following list shows a number of issues that are still poorly handled with tooling:

- Some tools only offer a text interface. To maintain a model, a graphical way of modelling is a basic necessity, especially in the case of realistic, i.e. complex models.
- To use some of the more complex models, the tooling needs to be able to handle this complexity. One way to do this is by using hierarchies of models. There are only a few tools that support this functionality.
- There is an enormous variety in the way in which tools automate testing. Some tools only generate test cases while others can be directly connected to the System Under Test (SUT). Other tools may be connected to a SUT, but only with the help of another tool. As was indicated above, the real benefits lie in automatically executing the test cases. This means that the tool must allow this.
- The reporting options of the tools often remain limited to indicating that a test case did not run properly. In other cases the outcome is a simple figure that indicates the coverage (e.g. path coverage). In addition to the fact that it is a labour-intensive task to locate the problem (in the SUT or in the model), these kinds of reports can also not be used for managing the testing process.
- Tools offer only one modelling technique and no more than this. Because you frequently need several techniques or even combine techniques to limit product risks (data and behaviour), a single technique does not suffice. It is expected that especially in these cases tools will prove to be useful.

Over the past years new tools have appeared, but they have not (yet) resolved the aforementioned issues. Until now there has not been a single tool that offers all the right functionalities and that can be used in almost all cases. Of course one tool may be better suited for a particular project and a different tool for another project. This allows you to work with the limitations of the tools in a constructive manner. In any event it helps to know the strengths and weaknesses of the various tools, in order to make an informed decision about which tool you apply.

6. CRITERIA FOR SUCCESS

MBT offers great opportunities to reduce the cost of testing in specific situations and/or to increase the quality. Because the method has often been used in the wrong situations, many organisations have stopped using MBT prematurely. Of course this is a shame, because MBT offers added value and opens possibilities for testing cases that could otherwise not be tested, especially in situations with a high risk and in which specific non-functionals need to be tested.

Recommendations

If you want to get started with MBT, be sure to see that the following criteria are met:

- Define a clearly **focused objective**, e.g. improving quality or efficiency. Make sure all stakeholders agree on this objective. If the agreed objective is quality improvement while a project leader is pressured by his client into focussing too much on time savings, the quality increase will be disappointing.
- Be sure to provide for sufficient expertise and experience in the domain of creating models. Without this expertise it is nearly impossible to determine the **required abstraction level** of the model.
- Maintain a **limited scope** and focus on specific areas or aspects of the system, for instance areas with a high product risk or specific non-functionals.
- Make sure that the right **tooling expertise** is available, because, as mentioned above, all tools have their limitations. Alternatively, hire the expertise.

Why ICT?

It is one thing to automate a complex process and achieve a functioning system. But how do you create an ICT solution that delivers more than this? How do you facilitate more speed? More convenience? Increased sustainability? Higher returns? These are the real challenges.

ICT Group is glad to tackle this challenge. The more complicated the project, the more enthusiastic we become. And the more ambitious the target, the more we push our boundaries. This is what drives us. And that is why we have been successful within the technological and industrial market for more than forty years.

With over 1400 professionals we support companies, products and projects in getting ahead with smart, innovative, integral and most of all, challenging ICT solutions.

About the author

Eduard Hartog: After 16 years of working as a developer and project leader, Eduard has chosen to specialise in quality aspects. Today, Eduard has gained over 15 years of experience in both testing and quality management in the technical domain. He is often involved in multidisciplinary projects with an installation technology component, such as tunnels, railroads and distribution centres. In addition he often works in the communications and offshore sector. Within TestNed he is active in the Model-Based-Testing work group.

eduard.hartog@improveqs.nl
+31 (0)6 23237330