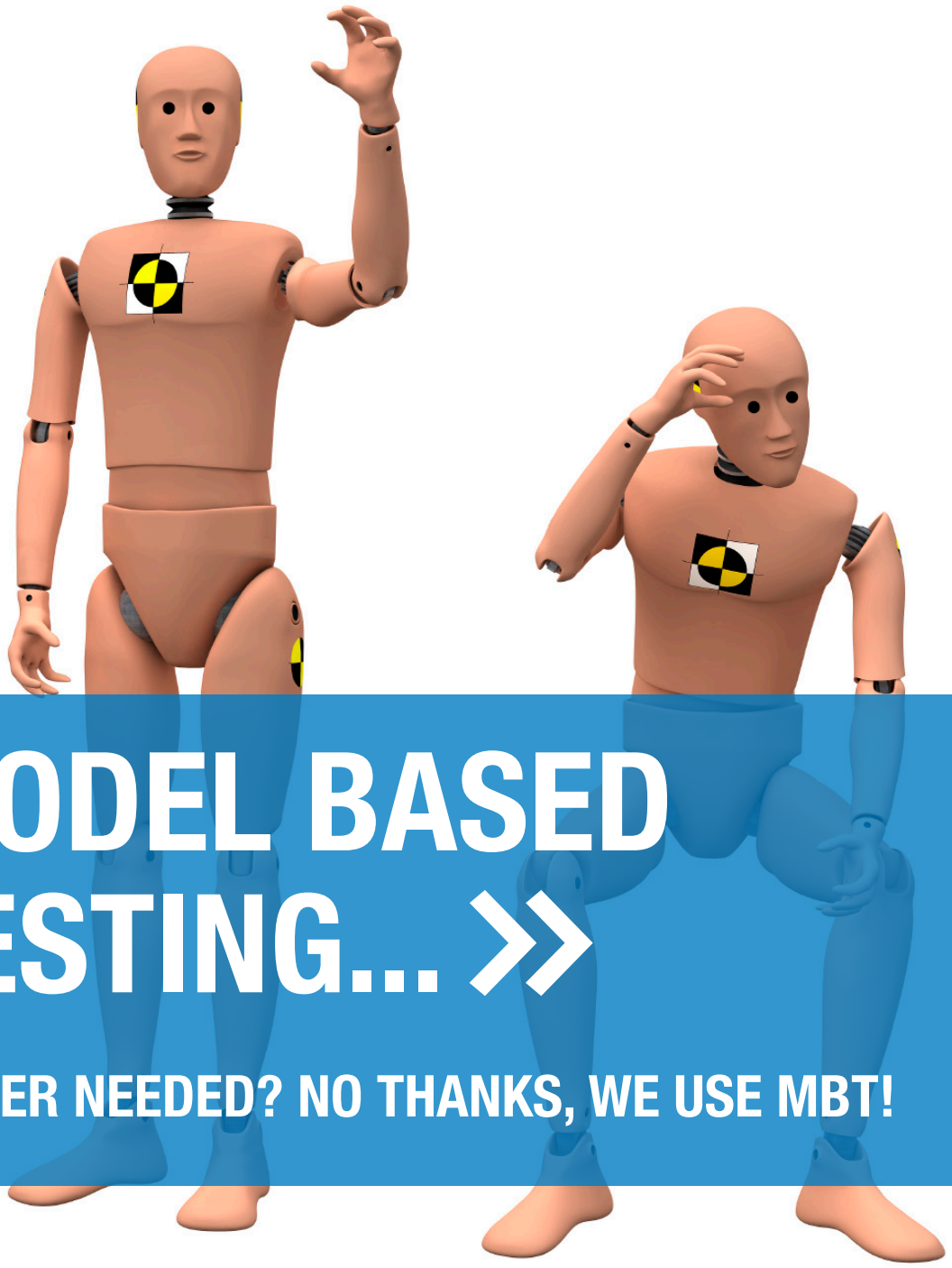
Atos Origin



# MODEL BASED TESTING... »

## TESTER NEEDED? NO THANKS, WE USE MBT!

**Whitepaper**

Atos Origin vision on MBT

by Elise Greveraars

# TABLE OF CONTENT

# INTRODUCTION

Atos Origin Nederland B.V. founded a working Group in 2007 to learn more about Model Based Testing (MBT).

The working group investigated 2 subjects, namely:
1. The current situation on MBT
2. The use of MBT for testing business processes and chains

To investigate the current situation regarding MBT, Atos Origin Nederland B.V. has performed a case study in cooperation with Smartesting®[1], Borland[2] and HP[3]. To investigate the use of MBT for testing business processes and chains Atos Origin Nederland B.V.[4] has started an investigation project in cooperation with Laquso[5] and Smartesting.

The information in this paper is the outcome of the working group. The focus for using MBT in this white paper is that of functional black box testing for administrative systems using test models that describe the expected behaviour of the System Under Test (SUT).

This white paper will go into detail regarding the current state of MBT and its future. It will explain what MBT is, and will provide an answer to the question whether we will still need testers in the future or if testers will be redundant once MBT has been applied.

# THE PRESENT SITUATION OF MODEL BASED TESTING

## INTRODUCTION

Model based testing has become a very popular term within the last few years. It is best known in the world of embedded systems. But what is the added value of MBT in the administrative world and what is needed to start using MBT?

This chapter will explain more about the present situation of MBT.

## PURPOSE AND DEFINITION

In general the idea of MBT is to create functional test models based on requirements. The requirements are thoroughly reviewed by creating test models. Once created, these test models are used for generating test cases. Generated test cases can be used for manual and/or automated test execution.

The definition of MBT in Wikipedia[6] is the following:
"Model-based testing is software testing in which test cases are derived in whole or in part from a model that describes some (usually functional) aspects of the system under test (SUT)."

This definition indicates that the only purpose of model based testing is to derive test cases from a test model. However the creation of test models also has substantial added value as an activity of a review process.

For this reason the definition of MBT can be refined to:
"Model-based testing is software testing in which a functional test model is created that describes some of the expected behaviour (usually functional) of the system under test (SUT). The purpose of creating this test model is to review the requirements thoroughly and/or to derive test cases in whole or in part from the test model. The test models are derived from the requirements."

A distinction can be made between informal and formal MBT. The difference between formal and informal MBT is that formal MBT uses formal test models that comply to certain standard modelling rules while informal MBT doesn't use formal test models. Test cases can be automatically generated from formal test models and must be manually generated from informal test models. Paragraph Informal Model Based Testing and Formal Model Based Testing will explain informal and formal MBT in more detail.

## INFORMAL MODEL BASED TESTING

Often testers start drawing some sort of test model while reading requirements. These test models help to get a better understanding of the requirements which are often only textual documents.

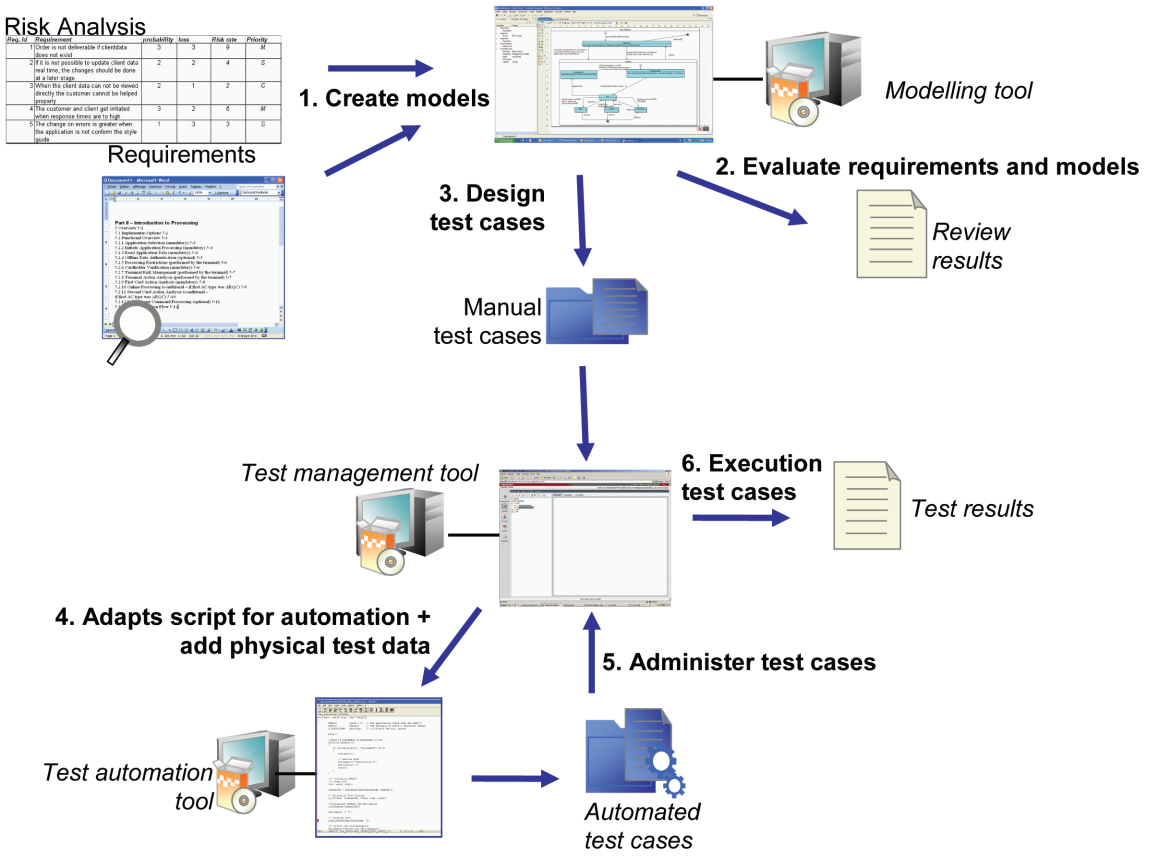
When drawing these test models many questions will arise. Sometimes because specifications contradict each other, contain incorrect information, are unclear or, even worse, not present. By drawing test models and asking questions, the tester will gain a good insight into the system to be tested and the tester will help the functional designer to get the specifications right. Drawing such test models will help improve the quality of the requirements and will assist in finding defects at a very early phase. Creating test models is also one of the first process steps in informal model based testing, see Figure 1.

So what exactly are the process steps of informal MBT?
Before starting to create test models it is important to consider what functionality needs to be modelled in which order. The answer to this consideration can be found in the Risk Analysis (RA)[7]. The RA should be the basis

for each test project. It contains information about which specifications have a high risk score for a certain version of the system. Based on this information a choice can be made on what to model, for example only model the highest risks requirements or model the highest risks requirements first and the lower risks requirements later. The creation of a RA is preferably done in the business analysis phase but is possibly performed later.

Once the risk information is clear the modelling can start, see step 1 in Figure 1. The notation of test models used for informal model based testing doesn't have to comply with certain modelling rules. The only rule for modelling is that the test models to be created are readable for the parties/persons involved in testing. Parties or persons involved can be for example other testers, functional designers, end users etc. These parties will have to determine themselves what is a readable form. Any tool which can create test models can be used for informal model based testing. An example of an informal test model is Figure 2.

**FIGURE 1: PROCESS STEPS INFORMAL MODEL BASED TESTING**

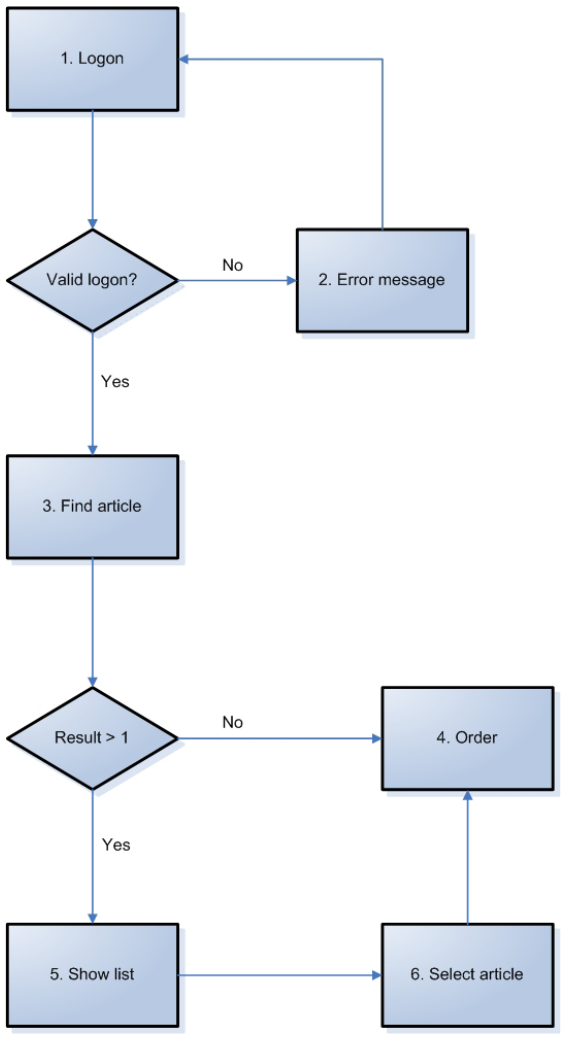


During the creation of test models incorrect, unclear or missing specifications will be found. This information should be registered, monitored and communicated to the parties involved, see step 2 in Figure 1. When needed the specification should be updated and released. After each release of the specifications the test models need to be screened and adjusted.

Once the specifications and the test models are approved by the parties involved, the design of the test cases can commence, see step 3 in Figure 1. The test models will be used as a guideline for designing the test cases and will give the test designers a good overview of test cases to be designed. The design of test cases is a manual action when using informal MBT.

In general test cases will be stored in a test management tool to be executed manually, see step 6 in Figure 1 or they will be adapted for automation test cases, see step 4 in Figure 1, to be executed automatically, see step 5 and 6 in Figure 1.

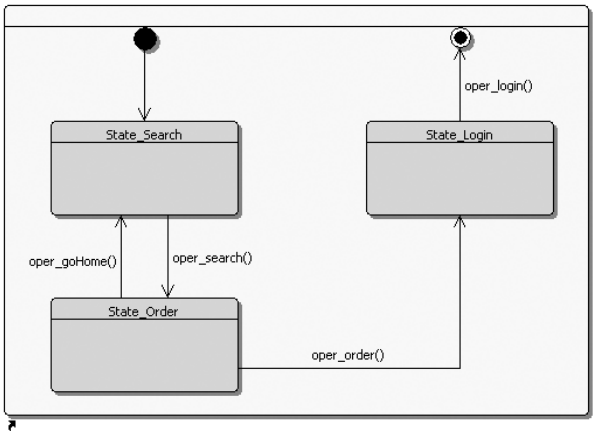
**FIGURE 2: INFORMAL MODEL**



## FORMAL MODEL BASED TESTING

One of the biggest advantages of formal MBT is the possibility to automatically generate test cases based on test models. To automatically generate test cases a tool is needed which should be able to interpret the test models.

**FIGURE 3: FORMAL MODEL**

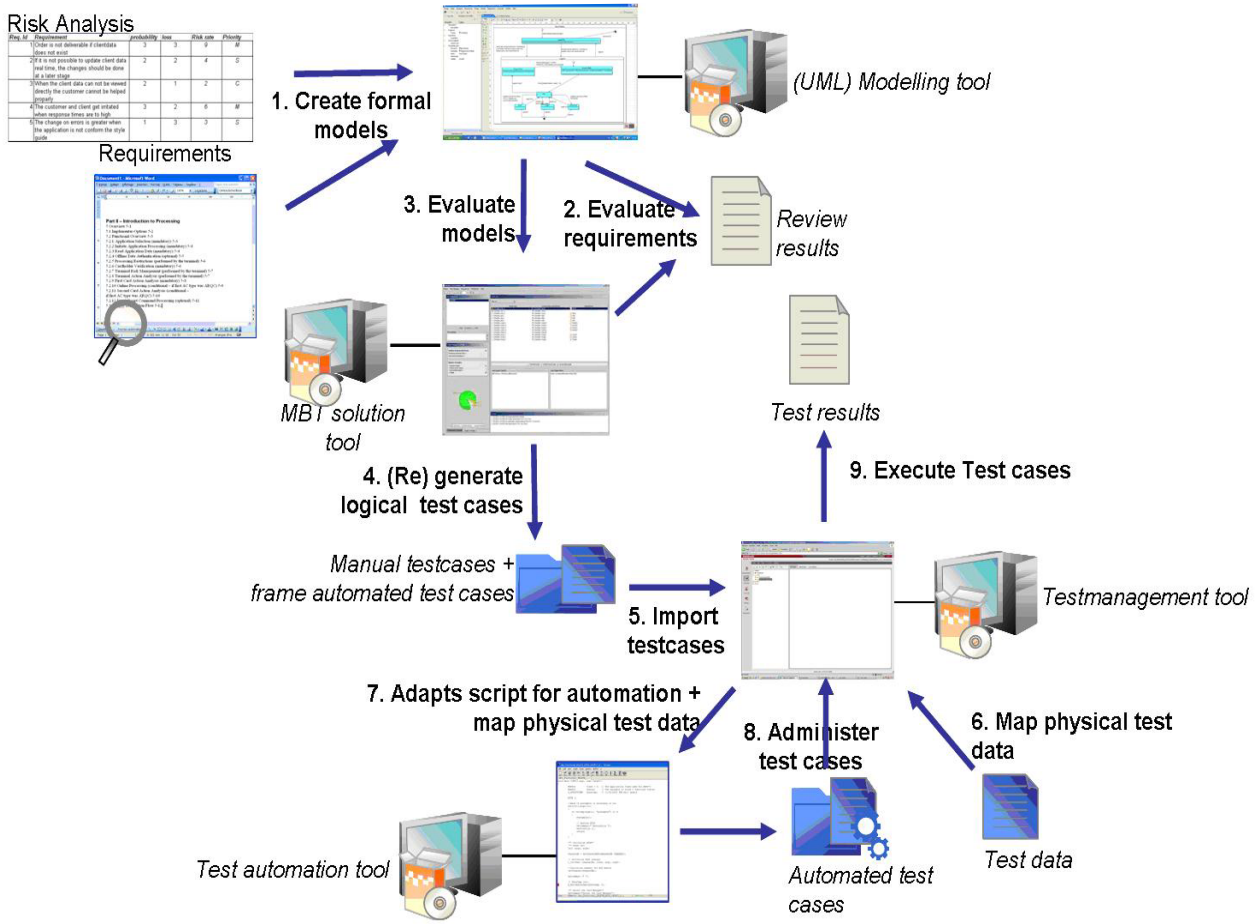



When using formal MBT it is not an option as with informal MBT to let the parties involved determine how the notation of a test model should appear. Accurate test models should be designed and formal rules should be followed for modelling. This way the test generation tool can check and interpret the test models and transform them into test cases. An example of a formal test model is Figure 3.

An example of a formal modelling notation is UML but many other modelling notations exist[8].

So what exactly are the process steps of formal MBT?
As with informal MBT it's important to consider what functionality needs to be modelled and in which order. This information should be extracted from the Risk Analysis (RA). Based on this information a choice can be made for example to only model the highest risks requirements or to model the highest risks requirements first and the lower risks requirements later, see step 1 in Figure 4.

## FIGURE 4: PROCESS STEPS FORMAL MODEL BASED TESTING




The test models to be designed must be formal and accurate. To be able to design such test models more sophisticated modelling tools are needed, for example Borland Together or IBM Rational Software Modeler. Sometimes model based solution tools[9] also offer modelling functionality. In this white paper the assumption is made that a separate tool is used for modelling purposes.

During the modelling phase incorrect, unclear or missing specifications will be found. These specification defects need to be registered, monitored and communicated to the parties involved, see step 2 in Figure 4. In formal MBT the test models will be offered to the model based solution tool for simulation. By simulating the test model the specifications will also be evaluated, see step 3 in Figure 4. Simulation also helps in finding interpretations defects which are possibly introduced by designing the test model. The findings of the simulation should be registered, monitored and communicated to the parties involved.

The specifications and test models will have to be fine tuned based on the findings and will have to be approved by parties/persons involved in testing and capable of understanding the formal test models. Parties or persons involved can be for example functional designers.

Once the test models are designed and approved the model based solution tool can generate the logical test cases, see step 4 in Figure 4.

The generated test cases can be for example HTML or XML output. Some model based solution tools also offer a plug-in for a test management/execution tool to make it possible to directly import test cases into the test management tool, see step 5 in Figure 4. The generated test cases will be logical test cases to be used for manual execution and a framework with automated test cases

The generated logical test cases can be adapted to physical test cases by mapping test data, see step 6 in Figure 4 after which the test cases can be executed manually see step 9 in Figure 4.

To automatically execute test cases the generated framework of automated test cases should be adapted, see step 7 in Figure 4. The generated test cases are abstract test cases and should be made concrete to make them executable. So for example abstract field names should be mapped to real field names used in the system, test data should be mapped to test cases etc. The adapted test cases will be administered in the test management tool, see step 8 in Figure 4 and can be executed automatically, see step 9 in Figure 4.

## POSITION OF INFORMAL AND FORMAL MBT IN THE V-MODEL

This paragraph will describe the position of informal and formal MBT in the V-model[10]. The V-model shows the relationships between each phase of the development life cycle and its associated phase of testing, see Figure 5.

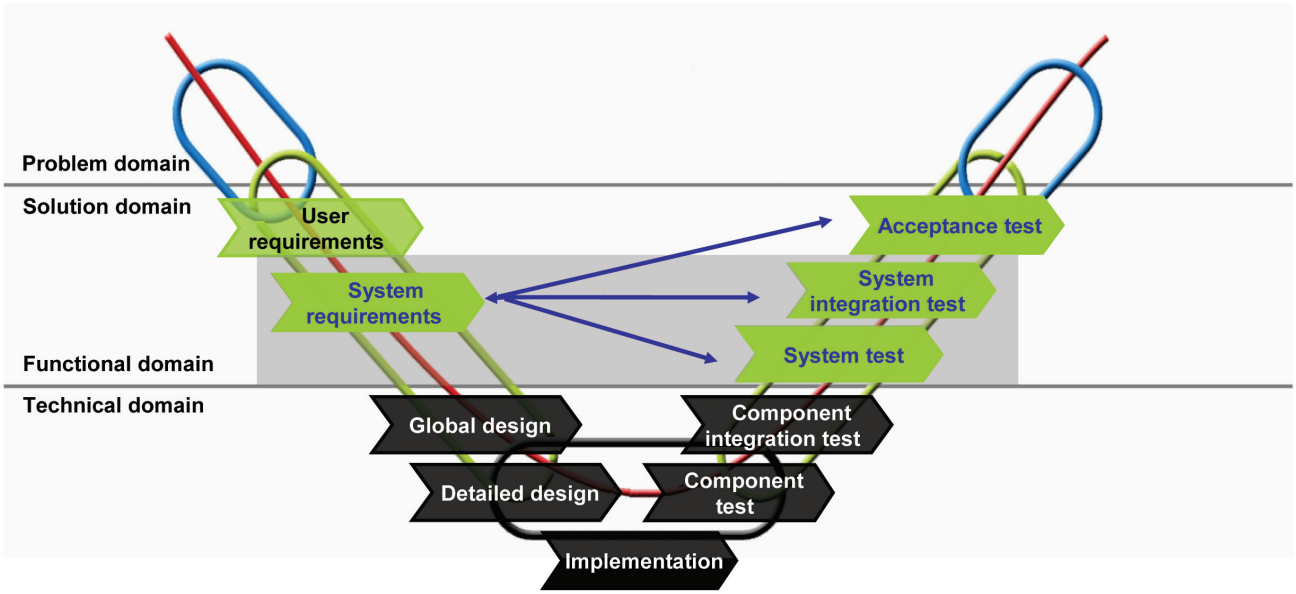
In the problem domain the wishes, opportunities, problems and policies are identified. These will be converted to a solution in the solution domain. The solution domain can be split up in a functional and technical domain.

Informal MBT can be used for both the functional domain and the technical domain within the solution domain. It will probably be used most in the functional domain for system, system integration and acceptance testing.

Test models as created in formal MBT are mostly based on the system requirements, see Figure 5. This makes formal MBT most suitable for system and integration testing. Formal MBT is less suitable for acceptance testing since acceptance testing should be based on user requirements. However parts of the test cases generated by MBT can possibly be re-used for acceptance testing.

The most common kind of testing with MBT in general is functional testing but can also be some kinds of robustness testing[11]. Automated test cases generated from formal models can possibly also be used for performance testing. MBT is less useful for usability testing.

**FIGURE 5. POSITION OF FORMAL MBT IN V-MODEL**




## MODELLING

This paragraph will give some background information on Model Driven Engineering (MDE) versus MBT and on how to deal with modelling when starting MBT.

### MODEL DRIVEN ENGINEERING (MDE) VERSUS MBT

Having some more background information on MBT let's have a look on how MBT is related to MDE.

What exactly is MDE[12]? MDE refers to a range of development approaches that are based on the use of software modelling as a primary form of expression. Sometimes models are constructed to a certain level of detail, and then code is written by hand in a separate step. Sometimes complete models are built including executable actions. Code can be generated from the models, ranging from system skeletons to complete, deployable products. With the introduction of the Unified Modelling Language (UML), MDE has become more popular.

MDE and MBT are both seen as the next step in the evolution of software development and are both based on the modeling of business processes, systems and programs as a primary form of development and testing. However, the purpose of models in MDE and MBT differ. The purpose of MDE is to produce code while the purpose of MBT is to produce test cases.

The content of the models created for MDE and MBT also differ. MDE models contain technical information that is needed for the production of code. This information is not needed for creating test cases. For this reason MBT test models should only contain functional information.

## TO REUSE MODELS OR NOT TO REUSE MODELS

So, can MDE models be reused for MBT projects? The answer to this question depends on the settings of the project worked on.

When there are no models available for the project, test models should be created from scratch. The advantage of creating test models from scratch is that the test model can be fully customized for test purposes. Another big plus is the fact that the test models to be created are not used to generate code. This gives total independency between the implementation and the testing. A down side of creating test models from scratch is the fact that nothing can be reused. The setup of initial test models often takes much time.

When working on a project that is applying MDE, models can possibly be reused for MBT. Reuse is only possible when the MDE models are not used to generate code. Any functional error in the model will generate incorrect implementation code and the generated test cases will contain the same error. The design of the system and the design of test cases should be as independent as possible[13].

When reuse is possible it is important to customise the models for test purposes and to review the models thoroughly with the specifications. As mentioned before the content of MDE and MBT models differ. For this reason all irrelevant information must be striped from the MDE models. Irrelevant information is for example implementation information and information that is not identified as a high enough risk to model.

To be able to create useful test cases it is important to model the expected behaviour of the system and the structure of the system. Models that contain information about the expected behaviour are called dynamic models; models that only contain information about the structure of the system are called static models. Possibly the available MDE models are only static models or dynamic models. In this case the missing models should be created.

Another option to reuse models is to make use of models created by reverse modelling an operational system. These models can possibly be used to create a regression test set to check if the functionality that is working in the current situation, will still be working in a situation in which certain parts of the system are updated. The updated parts should be modelled separately but the parts that have not been changed could be tested with test cases generated from the reverse models. When reusing reversed modelled models it is again very important to strip all irrelevant information from the models and to make sure static and dynamic models exist to be used for test generation.

When reusing MDE models there is no or little independency between implementation and testing. This and the fact that models should be customized for test purposes can be disadvantages of reusing models.

Combinations of the solutions mentioned are also possible for modelling.

## MODELLING GUIDELINES

What are some useful guidelines for modelling? First of all you have to choose a model notation for your test models. The choice of the model notation can be based on several aspects. One aspect that can influence this decision is that the company or the project prescribes or has a specific preference for test models to be used. In this case it is essential to look for a model based solution tool that supports these kinds of test models.

Other aspects that can influence the decision are the available model based solution tools. The model notation to be chosen should be supported by a model based solution tool.

Once the model notation is chosen it's important to consider what functionality needs to be modelled in which order, also see paragraph Informal Model Based Testing and Formal Model Based Testing.

When starting to model keep in mind that the purpose of the test model is to generate functional test cases. For this reason the model should only contain functional information and no technical and/or implementation data. Model only functionality that needs to be tested, the test model does not have to specify all the behaviour of the system.

The test models to be created will describe the structure of the system and the expected behaviour of the system. The test models to be designed should be accurate enough for testing the desired functionality but should also have a reasonable size and should be easy to maintain[14]. These two goals can be in conflict at times. The person responsible for designing the test model should be capable to decide what should be modelled to satisfy the test objectives and how much detail is useful.

Test models can make use of parameters to represent test data. These parameters can be replaced for actual test data after the test cases are generated.

## TEST GENERATION CRITERIA

The previous paragraph described the design of test models. When using formal MBT the model based solution tool will use the test models to generate test cases. To be able to generate test cases that will meet the test objectives, the model based solution tools will use certain generation criteria.

The generation criteria to be used depend on the test model and model based solution tool to be used. In the following example a finite state diagram will be used as a test model, see Figure 6. The model based solution tool can for example use the following generation criteria to create test cases:

- State coverage

Test cases will be generated that will visit every state in the model at least once. In the example of Figure 6, two test cases will be generated to meet this criterion. The test cases can be for example the following transition combinations: A, B and C, D, G.

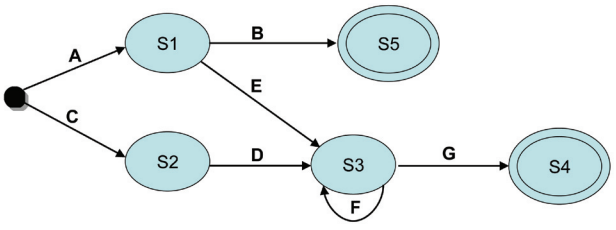
- Transition coverage

Test cases will be generated that browse every transition in the model at least once. In the example of Figure 6, three test cases will be generated to meet this criterion. The test cases can be for example the following transition combination: A, B and A, E, F, G and C, D, G.

- All transition pairs

Test cases will be generated that will combine all incoming transitions with all outgoing transitions in a state. In the example of Figure 6, five test cases will be generated to meet this criterion. The test cases can be for example the following transition combination: A, B and A, E, F, G and A, E, G and C, D, F, G and C, D, G.

**FIGURE 6. FINITE STATE DIAGRAM**



Test cases can also be generated on the basis of test data generation criteria. These criteria will choose data values as test inputs. The data values will be chosen for example on basis of boundary values, random values, equivalence class values, etc.

The test generation criteria mentioned here are only a few examples, many other generation criteria exist[15]. Generation criteria can also be combined in order to generate test cases.

As can be seen in the example of the finite state diagram the generation criteria will control the choice of tests and determine the depth and size of test cases. The depth and size of test cases generated with the generation criterion 'state coverage' is less than when using for example the generation criterion 'all transitions pairs'.

The generation criteria to be used should depend on the risk score of the modelled requirement. The generation criteria for a high risk requirement must be more extensive than the generation criteria for a low risk requirement. In the current situation model based solution tools do not yet support the control of generation criteria.

## TEST ORGANIZATION

This paragraph will give more background information on the impact MBT can have on the current test organisation. A new role will be introduced; the test constructor role and the impact on the traditional roles will be described.

### IN GENERAL

When applying MBT it is very important to have the commitment of all parties, this includes management. MBT requires an adapted way of working from the traditional testing. Test models will have to be designed in an earlier phase of the project compared to the traditional design of test cases.

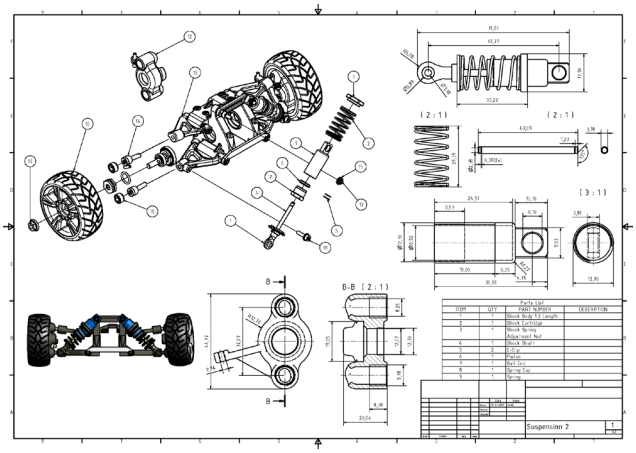
By designing test models the testers are able to formulate critical questions early. The issues raised by the testers will help the functional designer in writing correct requirements. For this, much interaction will be needed between the functional designer and the tester. Management will have to support and propagate this way of working. They will also have to invest in training and possibly in tooling; commitment is essential.

### TEST CONSTRUCTOR

Model based testing requires additional skills to traditional testing skills. Required skills are modelling skills, skills on transforming written specifications into models and a basic knowledge regarding programming. For this reason a new role or function is introduced for MBT, the test constructor.

According to Wiktionary[16] a constructor is a person who:
Builds or forms by assembling parts and draws models by following precise specifications and using geometric tools and techniques.

**FIGURE 7: A CONSTRUCTION**



An example of work of a constructor is shown in Figure 7.

Based on this definition, the definition of a test constructor is:
A test constructor is a person who builds or forms the necessary test models by following precise specifications, using a good level of abstraction for test purpose and using tools and techniques for modelling and the generation of test cases.

The test constructor should have domain and testing knowledge. He or she will have much interaction with the different parties involved in the project to cross check the models with the available information. For this reason the test constructor should also have good social skills.

The test constructor should be involved in the test project as early as possible. By designing test models early in the process, many defects can be found before one line of code has been written.

It is the responsibility of the test constructor to choose a good level of abstraction to satisfy the test objectives.

### TRADITIONAL TEST ROLES/FUNCTIONS

Traditional test roles or functions will still be needed when introducing MBT into the organisation. To use MBT it is important for the traditional roles to gain general knowledge on MBT, modelling and tooling.

In addition to gaining this knowledge the test manager will have to integrate MBT into the test strategy of the company.

The person responsible for the test automation, the test automater role or function, will play an important role in adapting the generated automatic test cases to executable test cases. In order to do so, he or she will have to gain more knowledge of the MBT tooling and the generated automatic test cases.

MBT takes over (parts of) the role or function of the test designer. However the test designer can also play a role in the functional modelling and will be needed to design test cases which are not covered by MBT.

The test executer will be responsible for executing the manual test cases which may or may not have been generated by MBT.

## MARKET DEVELOPMENT

This paragraph will give information about the current market development concerning model based solution tools and about the current adoption phase of MBT.

### TOOLS

There are already many commercial and non commercial MBT tools on the market[17]. The purpose of these tools is to generate test cases based on test models. Some tools also support modelling functionality and/or test execution functionality.

The test tools can be grouped in online and offline model based testing tools[18]. Offline model based testing tools will generate a finite set of tests to be executed later. This allows automatic test execution in third party test execution platform. Online model based testing tools will connect directly to a system under test and test it dynamically. Online model based testing tools are common in the embedded industry while offline model based testing tools are more common in the administrative domain.

The different tools available support different models notations like for example UML, OCL, B notation, Spec#, FSM, etc. Some tools also offer plug-ins for modelling tools like for example Borland Together or IBM Rational Software Modeler. This plug-in will check the consistency of models.

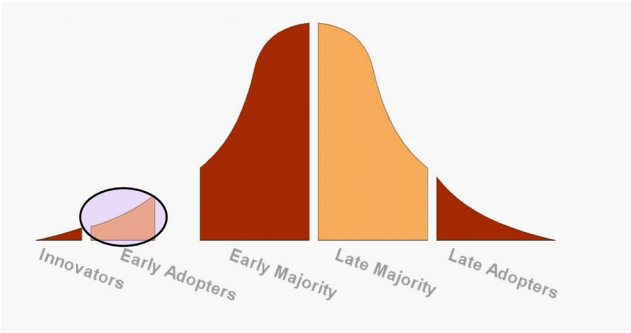
Each tool provides its own test case output. Test cases are generated as HTML, XML, ASCII, Comma Separated, TTCN-3 output etc. Some tools also offer plug-ins like for example exporting test cases to HP Quality Center, etc.

Different generation criteria are used by different tools. The used generation criteria also depend on the supported models.

### ADOPTION

The majority of testers are already applying some form of informal MBT. As with MDE[19] the adoption of formal MBT is still in the early adopter's phase of the technology adoption lifecycle model[20], see Figure 8. The technology adoption lifecycle model describes the adoption or acceptance of a new product or innovation, according to the demographic and psychological characteristics of defined adopter groups. Early adopters are in general young and educated people tended to be community leaders.

**FIGURE 8: TECHNOLOGY ADOPTION LIFECYCLE**




The gap between the early adopters and the early majority still needs to be crossed. This gap is also called the chasm[21]. The early majority are in general more conservative people who are open to new ideas, are active in the community and can influence neighbours.

# CASE STUDIES & LESSONS LEARNED

This chapter will describe the results of 2 case studies. In one case study informal test models were designed and used for review purposes, the second case study is a formal MBT project to gain experience on MBT.

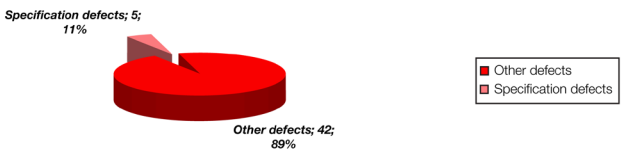
## CASE STUDY INFORMAL MODEL BASED TESTING FOR REVIEW PURPOSES

In this case study a comparison is made of specification defects found in two versions of a newly built application. Both versions were designed by the same functional designer and had been previously informally reviewed.

Based on function points both versions are of an equal size. For version 2 informal test models were created as part of the activity intake test basis, for version 1 no test models were created.

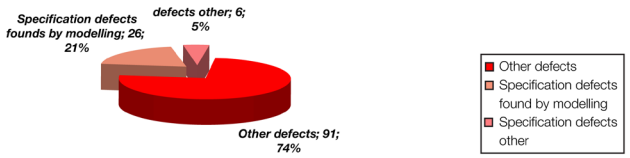
The results of this case study show that in version one out of all defects 11% were specification defects found prior to testing, see Figure 9.

**FIGURE 9: SPECIFICATIONS DEFECTS IN VERSION 1**




In version two out of all defects 26% were specifications defects found prior to testing, see Figure 10.

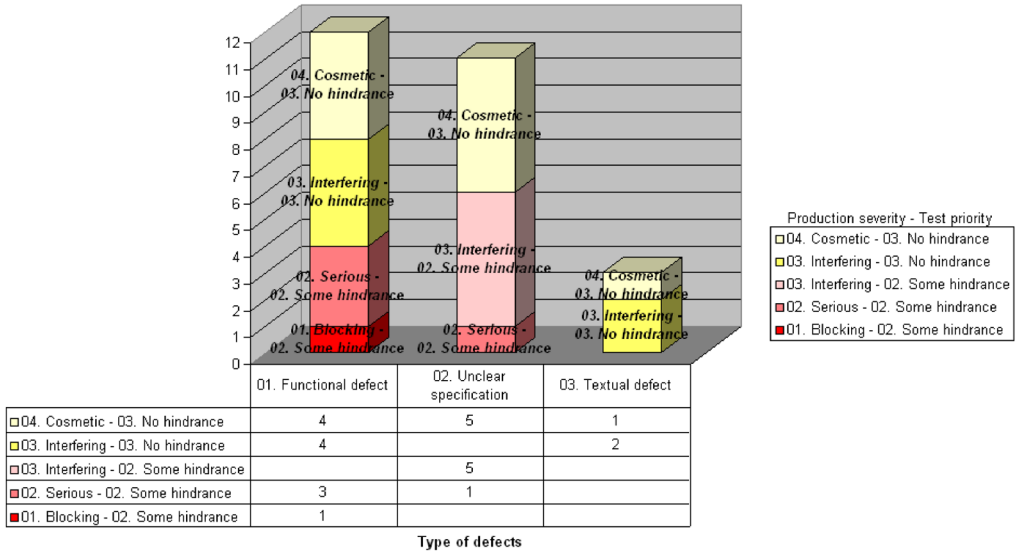
**FIGURE 10 SPECIFICATIONS DEFECTS IN VERSION 2**




When evaluating the types of specification defects in version 2, the following types of specification defect were found:

- Functional defects
- Defects about unclear specifications
- Textual defects

## FIGURE 11: SPECIFICATIONS DEFECTS FOUND BY MODELLING



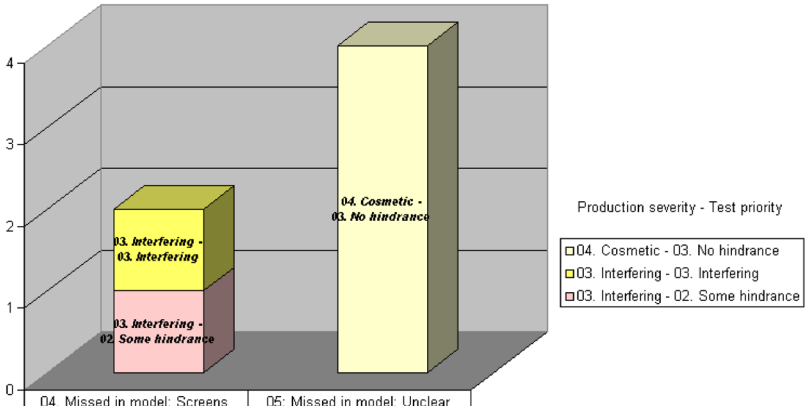

| | 01. Functional defect | 02. Unclear specification | 03. Textual defect |
|---|---|---|---|
| 04. Cosmetic - 03. No hindrance | 4 | 5 | 1 |
| 03. Interfering - 03. No hindrance | 4 | | 2 |
| 03. Interfering - 02. Some hindrance | | 5 | |
| 02. Serious - 02. Some hindrance | 3 | 1 | |
| 01. Blocking - 02. Some hindrance | 1 | | |

Type of defects

The production severity of the specification defects found ranged from blocking to cosmetic as can be seen in Figure 11.

At the end of the case study we found that a few specification defects were not found when creating the test models for version 2. The specification defects that were not found related to screen specifications which were not well defined and with unclear specifications. The

## FIGURE 12: SPECIFICATIONS DEFECTS FOUND BY MODELLING




| | 04. Missed in model: Screens specs not well defined | 05: Missed in model: Unclear specification |
|---|---|---|
| 04. Cosmetic - 03. No hindrance | | 4 |
| 03. Interfering - 03. Interfering | 1 | |
| 03. Interfering - 02. Some hindrance | 1 | |

Type of defects

unclear specifications were cosmetic issues. Because it is not possible to model the screen specification, it was not surprising that these specification defects were missed.

### LESSONS LEARNED

One of the lessons learned during this case study was that it is very useful to create test models to support the review process. A lot of missing and unclear specifications were identified early. Some time had to be invested for the creation of the test models but this time paid dividends in the later phases due to the knowledge the tester had gained early on by creating the test models. This knowledge was very useful to the testers during the design and execution of test cases.

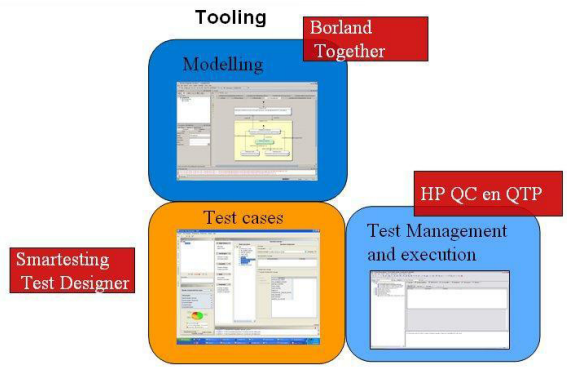
Another lesson learned was that it is important to start modelling during the review process and not as part of the activity intake test basis. At this stage the functional designers do not want to review the created test models anymore. They consider this as double work as the specifications have been finalised. When starting to model during the review sessions more commitment can be expected from the functional designers on reviewing the test models.

## CASE STUDY FORMAL MODEL BASED TESTING

The purpose of this case study was to gain knowledge on formal MBT and to judge the usability of formal MBT. The case study was performed in cooperation with the model based solution supplier Smartesting.

For this case study a team of test specialists was formed and trained in 3 weeks on formal modelling and tools to be used. The approach used was incremental and iterative, starting small and simple and expanding gradually. By applying this approach the team could quickly get an impression of using MBT and gradually gain knowledge on the test models to be designed and tooling to be used.

**FIGURE 13: TOOL COMBINATION**




The following tool combination was used, see Figure 13:

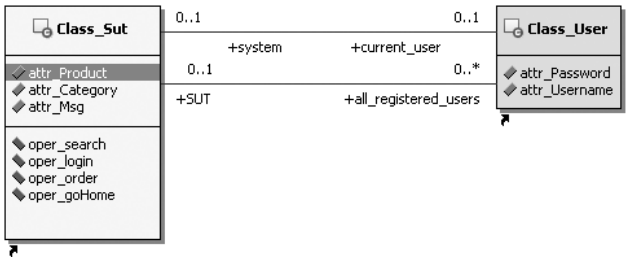
- Borland Together 2006 release 2 for Eclipse; to design the test models
- Smartesting Testdesigner; to evaluate the correctness of test models and to generate the test cases
- HP Quality Center; for test management functionality
- HP QuickTest Professional; to adapt and automatically execute the generated test cases

The following UML model notations were used for the case study and supported by Smartesting Test designer:

- Class diagram, see Figure 14

Class diagrams describe the relations between entities like for example people, things and data and describe the attributes and possible operations of the entity.

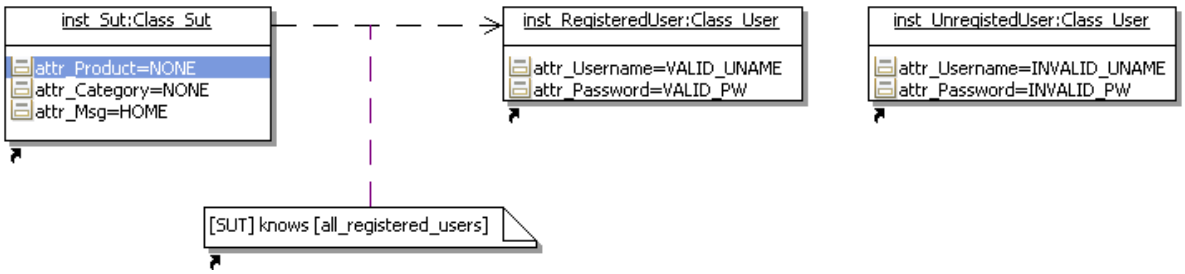
**FIGURE 14. CLASS DIAGRAM**




- Object diagram, see Figure 15

An object diagram shows a subset of elements from the class diagram. This subset describes the relation and value at a specific point in time. The object diagrams are used in MBT to describe the initial state to use for generating test cases.

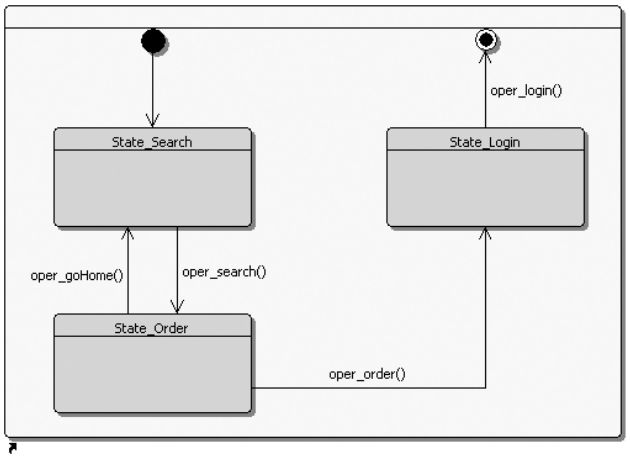
**FIGURE 15. OBJECT DIAGRAM**




- State diagram, see Figure 16

A state diagram describes all possible states of a system and the transitions needed to reach a certain state.

**FIGURE 16. STATE DIAGRAM**



## FIGURE 17: OCL CODE

```
---@REQ: ACC_MNGT/AUTH
if param_user = ENUM_USERNAMES::INVALID_UNAME then
    ---@AIM: AUTH_Invalid_Username
    attr_Msg = ENUM_MSG::LOGIN_ERROR
else
       let user_found:Class_User = all_registered_users->any(attr_Username = param_user) in
           if user_found.attr_Password = param_pass then
               ---@AIM: AUTH_Success
               self.current_user= user_found and
               attr_Msg = ENUM_MSG::LOGIN_SUCCESS
           else
```

The conditions needed for a transition to take place are described in the pseudo language Object Constraint Language (OCL), see Figure 17.

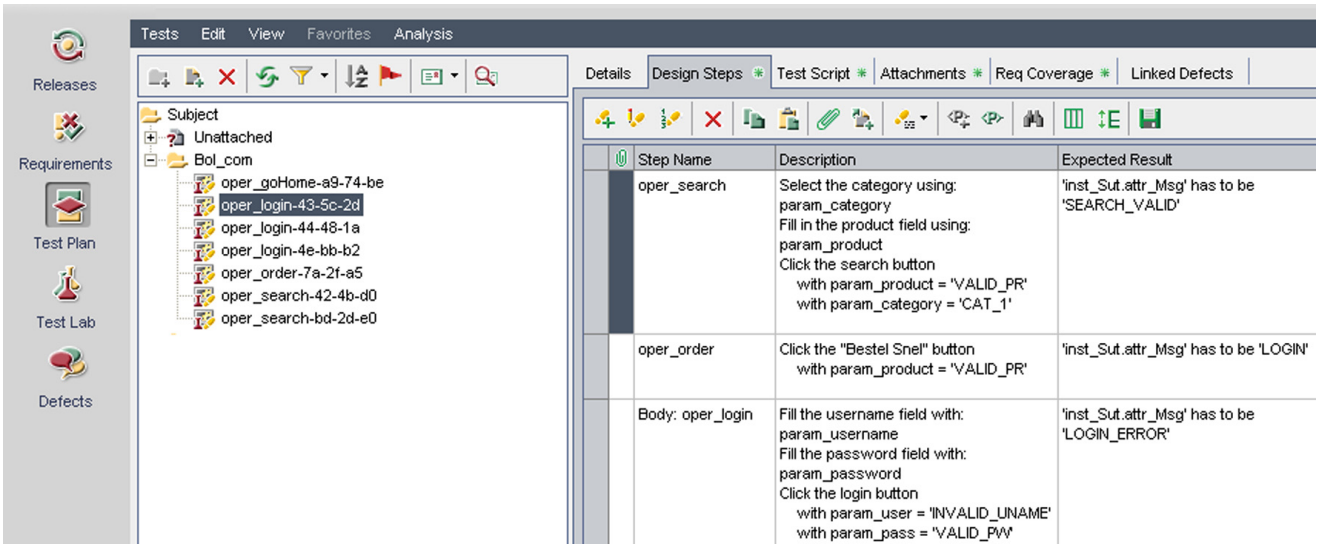
For the generation of test cases Smartesting Test Designer supports the following generation criteria:

- Transition coverage
- Each condition in the decisions (D/CC criteria) coverage
- One data value per equivalence class coverage

The first generation criterion is based on the state machine, the two other generation criteria are based on OCL.

Figure 18 shows an example of a generated test case. This test case will first search for a valid product to order. The user should login before ordering. In this test case an invalid user name is provided that will result in an error message.

## FIGURE 18: AN EXAMPLE OF A GENERATED TESTCASE



## LESSONS LEARNED

The case study was very successful in gaining knowledge of MBT. The testers were very enthusiastic to learn about MBT. For them MBT was a new and exiting challenge. They found modelling to be a very creative task which could be learned easily when education in modelling and tools is provided. In order to design the test models correctly much interaction was needed with other parties.

The test models gave the testers a better overview than the textual specification and were very useful in communicating with functional designers. However, knowledge is needed to understand the test models. For this reason the test models used are not considered to be business user friendly.

The test generation did save a lot of time on designing test cases manually. The greatest added value of MBT is to be gained when using test automation. Beside manual test cases Test Designer of Smartesting will automatically generate a framework of automated test cases which will reduce the time to create/design automatic test cases significantly.

At the start of the case study time was required to design the test models. The initial set-up of the test model was more time consuming than updating the test models. The advantage of creating test models in an early phase is the early detection of specification defects.

Unfortunately it is not possible yet to make a difference in the test generation for high or low risks requirements. Currently test cases are generated from the requirements as if they all have the same risk level. When executing test cases manually, this means that the test cases have to be sorted out by hand. The selection of test data and adaptation of automated test scripts can also provide extra work which may not have been needed. Currently, the only option to reduce the number of generated test case is to reduce the modelling of requirements.

The naming and the readability of the generated test cases still have room for improvement. The test name for example doesn't contain enough information about the test case, see Figure 18. This issue was improved after finishing the case study.

The tool Test designer from Smartesting was found easy to use. The tool also supports the usage of references to requirements in the test models. These requirements with linked test cases generated from the test model are administered in HP Quality Center. This makes traceability possible between the test models, requirements, test cases and test execution.

In general MBT can certainly help to improve the test process and test results. Savings in time and money are expected once the people are trained in modelling and initial test models are designed.

# THE FUTURE OF MODEL BASED TESTING

This chapter will describe some developments to be expected in the future. One of the developments is risk based test generation, the other development is business model based testing in which testing can be applied on acceptance level.

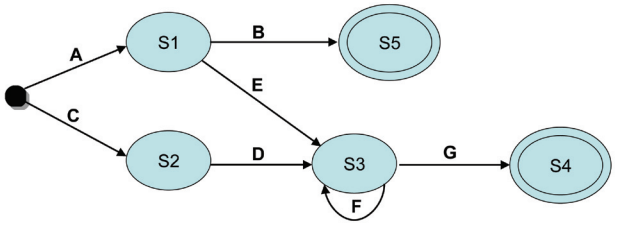
## RISK BASED TEST GENERATION

It would be very useful to be able to control the generation criteria for generating test cases in link with the level of risk of the tested functions or requirements. This in not currently available in the model based testing tools.

The test manager/coordinator must be able to tell the model based solution tool for example to:

- Generate test cases based on the generation criteria 'all transition pairs' and on the test data generation criteria equivalence classes for high risk requirements
- Generate test cases based on the generation criterion 'state coverage' and to use random values as test data generation criteria for low risk requirement

**FIGURE 19. PROBABILITIES IN A FINITE STATE DIAGRAM**



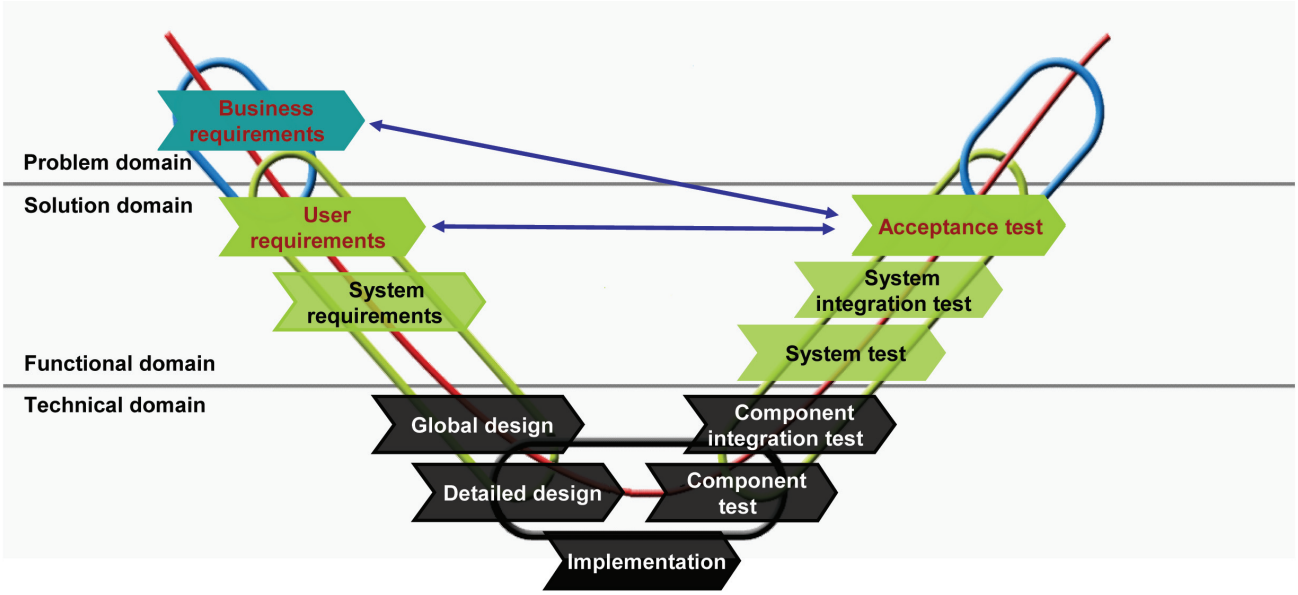

Besides this it must be possible to generate test cases based on probabilities. Figure 19 shows a state diagram in which the following 3 transition paths are possible; A, B and A, E, F, G and C, D, G. The probability will express the likelihood of a transition path to be used. The probability can be for example 15 % for the transition path A, B, 25 % for the transition path A, E, F, G and 60% for the transition path C, D, G. For the creation of test cases it should be possible to take this probability factor into account.

The expectations for the future are that more model based solution suppliers will solve this issue by providing the functionality of some sort of dash board in which the test manager can control what generation criteria are used for what risk of a requirement and in which he or she can control the test case generation based on probabilities.

## BUSINESS MODEL BASED TESTING

In the current situation, formal MBT is mainly based on system requirements and test models that describe these system requirements. This makes it less or not suitable for acceptance testing, see Position Of Informal And Formal MBT In The V-model. In the acceptance test the accepting party (also called demanding party) wants to determine whether what has been asked for is actually being delivered and whether it can do with the product what it want to/must do[22]. The accepting party must verify if the product provides the necessary support to the business process.

**FIGURE 20. POSITION OF FORMAL MBT IN V-MODEL**




To be able to use MBT for acceptance testing, test models must be based on the user requirements and business requirements. The test models must describe business processes and end-to-end testing and must be readable for the business users. To bridge the gap between the usage of the current MBT and to use MBT for acceptance testing, Business Model Based Testing (BMBT) will be introduced. Figure 20 shows the position of BMBT in the V-model.

What exactly is BMBT?
Business model-based testing (BMBT) is business process and software testing in which a test model is created that describes aspects of the business process including the system under test (SUT). The purpose of creating this test model is to review the requirements thoroughly and/or to derive test cases in whole or in part from the test model.

By applying BMBT aspects of the business processes will be validated and verified. Now what do we mean by a business process? One of the definitions is[23]:
A business process is a collection of activities that takes one or more kinds of input and creates an output that is of value to the customer.

The activities of a business process mentioned in the definition are often a combination of manual and automated processes. The manual processes are being executed by clients, employees or others and are supported by software solutions. The automated processes are being executed by software solutions. Dependencies exist between the activities in a business process and between the different business processes.

An example of a business process and possible dependencies:
A customer in the USA places an order via internet in dollars. When the shipment is worth more than € 700 import duties will have to be paid and customs will have to provide papers for this shipment. The exchange rate lists will be updated hourly. So for the customer in the USA, shipment terms depend on the exchange rate lists and its update process. The behaviour of the client can also depend on the changes in this exchange rates list. If the dollar rate is unfavourable, the client can decide to cancel his order.

The test models used in BMBT will describe these business processes and the dependencies of these business processes. The visualisation in a test model will help improve the quality of the business and user requirements in a very early phase. It will make business owners more aware of the consequences of certain choices that have been made.

In the example given above it is possible that the business owner becomes more aware of the fact that the client may cancel his order due to the shipment terms. Based on this information the business owner might decide to offer a discount or to divide the shipment to avoid the shipment terms.

The test models designed in BMBT will also be used to generate test cases for the acceptance testing. The generated test cases will be used to test if business processes and chains are working correctly. Because of the different scope of MBT and BMBT different type of defects will be found when executing the test cases. BMBT testers will detect defects which traditional testing would not have found.

For BMBT the same distinction as with MBT can be made between informal and formal BMBT. For formal BMBT test models must be used that comply to certain standard modelling rules such as the Business Process Modelling Notation (BPMN)[24]. At this moment, it is not possible to apply formal BMBT since no model based solution tools exist that support the simulation of the test models and the generation of BMBT test cases. Simulating the test models will help in checking the correctness of the test models and business processes.

Since a business process is a combination of manual and automated processes, manual and automated test cases will exist in parallel in the future of formal BMBT.

Informal BMBT does not have to comply with a standard modelling notation and can already be applied at present since no special test case generation tooling is needed. The generation of test cases will be done manually.

The test models designed for both formal and informal BMBT will be used as input for the design of test models for MBT. The test models are supplementary to each other.

Currently Atos Origin Nederland B.V. is already applying informal BMBT in their test method Business Process Validation[25]. Formal BMBT is being further developed by Atos Origin, Smartesting and Laquso.

# CONCLUSIONS & RECOMMENDATIONS

This chapter will describe the conclusion on MBT and some recommendations on how to best start using MBT.

## CONCLUSIONS

Testers will certainly still be needed when using MBT. To use MBT those executing the traditional roles or functions will have to be educated on modelling, MBT and tools used. A model based test designer, the test constructor, will be needed in addition to the traditional roles or functions.

By designing test models knowledge is gained on the system and defects are found at a very early stage. This will reduce much development and testing time. The test models are very useful for the creation of test cases. This is the case for formal and for informal test models. Formal models are partically useful since these can be used for automated test case generation. It is important to model risk based, so start modelling the high risk requirements.

Formal MBT will heavily reduce the maintenance on test cases; this is especially the case when applying automated testing. Instead maintenance on test models will be needed. Maintaining test models will be by far less time consuming than maintaining large test repositories.

Formal MBT will provide very high test coverage but unfortunately it is not possible yet to generate test cases risk based. Model based solution providers will probably implement risk based test generation in the near future.

In general MBT can certainly help to improve the test process especially on the level of system testing and system integration testing. Reduction in time and costs are to be expected once the initial test models are designed. Because of this the adoption of formal MBT will continue to increase. Besides this growth there will be an increasing need for business model based testing which can be applied on acceptance level.

## RECOMMENDATIONS TO START USING MBT

So now you want to start using MBT, how to best start?

Informal MBT can always be used. Tooling to design test models for example will be handy but are not required for informal MBT. First try to find out what are the most important requirements. Once these are clear start reading the specifications and just start creating test models. Be surprised by the questions that will arise and the knowledge that will be gained by doing so. Once these test models exist, discover how easy it is to use them as the basis for designing the test cases.

A first step can also be to start a small project to gain experience on formal MBT. This will require the necessary tools like a modelling tool, an MBT tool for the generation of test cases and test management and execution tooling. Decide what model notation to use and make sure the people who will work on this project will get a proper training in modelling, MBT and tooling.

Start modelling as early as possible in the project and start to model simple functionality to gain experience. Gradually more complex functionality can be modelled. Test models can get complex when not applying the correct level of abstraction. For this reason it is important to design test models that contain enough test information but will stay relatively small at the same time.

When applying MBT always create test models based on the risks of the requirements. Consider if regression tests are to be expected for certain functionality. If no regression is to be expected it might not be useful to invest substantial time in the initial set up of a formal test model. On the other hand take into consideration that investing time in creating the test model will help to find specification defects in an early phase.

And last but not least; hop on board and get enthusiastic!!

# REFERENCES, LINKS AND FURTHER READINGS


[1]Smartesting, a model based solution provider: http://www.smartesting.com

[2]Borland, an international IT products and solutions provider: http://www.borland.com

[3]HP, an international IT products and solutions provider: http://www.hp.com

[4]Atos Origin Nederland B.V., an international IT services provider: http://www.nl.atosorigin.com

[5]Laquso, laboratory for quality software http://www.laquso.com

[6]Wikipedia, model based testing: http://en.wikipedia.org/wiki/Model_based_testing

[7]TMap Next for result-driven testing, UTN pubishers 2006, ISBN10: 9072194802, ISBN13: 9789072194800, Tim Koomen, Leo van der Aalst, Bart Broekman, Michiel Vroon

[8]Practical Model Based Testing, a tools approach, Elsevier Science & Technology 2006, ISBN10: 0123725011, ISBN13: 9780123725011, Mark Utting and Bruno Legeard, paragraph 3.1.1. Notations for modelling

[9]Tooling that will validate test models and will generate test cases.

[10]TMap Next for result-driven testing, UTN pubishers 2006, ISBN10: 9072194802, ISBN13: 9789072194800, Tim Koomen, Leo van der Aalst, Bart Broekman, Michiel Vroon, paragraph 2.3.3

[11]Practical Model Based Testing, a tools approach, Elsevier Science & Technology 2006, ISBN10: 0123725011, ISBN13: 9780123725011, Mark Utting and Bruno Legeard, paragraph 1.1. What do we mean by testing?

[12]Wikipedia, model driven engineering: http://en.wikipedia.org/wiki/Model-driven_engineering

[13]Practical Model Based Testing, a tools approach, Elsevier Science & Technology 2006, ISBN10: 0123725011, ISBN13: 9780123725011, Mark Utting and Bruno Legeard, paragraph 2.3. Models: Build or borrow?

[14]Practical Model Based Testing, a tools approach, Elsevier Science & Technology 2006, ISBN10: 0123725011, ISBN13: 9780123725011, Mark Utting and Bruno Legeard, paragraph 1.2. What is model based testing?

[15]Practical Model Based Testing, a tools approach , Elsevier Science & Technology 2006, ISBN10: 0123725011, ISBN13: 9780123725011, Mark Utting and Bruno Legeard, chapter 4. Selecting your tests

[16]Wiktionary, constructing: http://en.wiktionary.org/wiki/construct

[17]List of commercial MBT tools: http://www.cs.waikato.ac.nz/~marku/mbt/CommercialMbtTools.pdf

[18]Presentation Model based testing tools by OlliPekka Puolitaival, VTT Technical Research Centre of Finland, presented at ECMDA-FA, june 2008, Berlin: http://www.cs.tut.fi/tapahtumat/testaus08/Olli-Pekka.pdf

[19]Keeping the Forest in View when Representing and Manipulating Large-Scale Models by Houari Sahraoui, DIRO, Université de Montréal, presented at ECMDA-FA, june 2008, Berlin.

[20]Technology adaption curve: http://en.wikipedia.org/wiki/Technology_Adoption_LifeCycle

[21]Marketing book, Crossing the Chasm, HarperCollins Publishers, ISBN10: 0060517123 | ISBN13: 9780060517120, Geoffrey A. Moore

[22]TMap Next for result-driven testing, UTN pubishers 2006, ISBN10: 9072194802, ISBN13: 9789072194800, Tim Koomen, Leo van der Aalst, Bart Broekman, Michiel Vroon, paragraph 6.1

[23]Wikipedia, business process definition http://en.wikipedia.org/wiki/Business_process#Definition

[24]Wikipedia, Business Process Modelling notation (BPMN): http://en.wikipedia.org/wiki/BPMN

[25]Business Process Validation, Atos Origin 2007, Klaas Smit & Gerlof Hoekstra