



# Experiences from Introduction and Deployment of MBT at Ericsson

Håkan Fredriksson

Ericsson AB

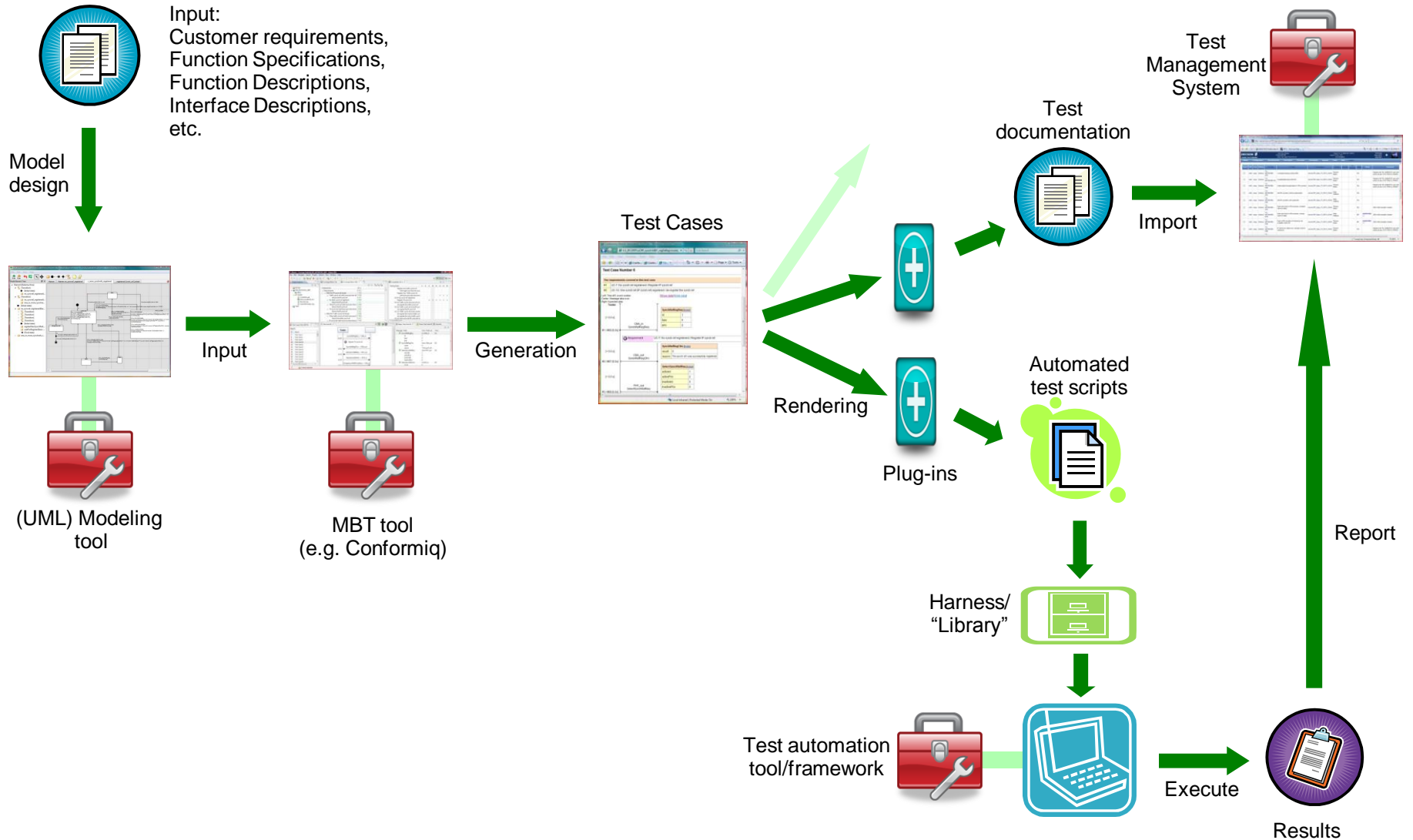
[hakan.fredriksson@ericsson.com](mailto:hakan.fredriksson@ericsson.com)

# Contents

---

1. Introduction
2. Modeling
3. Test Case Generation
4. Conclusions

# MBT - Introduction



# MBT - Introduction

---

Where we were before MBT

- A fairly stable test organization
- A fairly mature product to test
- Fairly stable test environment and tools (partly self developed)
- Main part (90%) of the Test Cases automated
- A huge amount of Test Cases/scripts – which had started to become expensive to maintain

The perfect background for trying out new tools and methods!

# MBT - Introduction

---

Why we tried it out

- Striving for Operational Excellence
- Reduced costs and shortened lead-times, by
  - generation of test documentation
  - generation of Test Cases
  - generation of automated test scripts
- Increase agility
- Improve test coverage
- “Better” (more complex) Test Cases
- Improve Test Case and test script quality
- Inspirational challenge for the testers
- Highlighting the tester’s role in the organization
- ...and so on

# MBT - Introduction

---

## How we introduced MBT

- Conformiq were invited to perform prototyping with an already verified function
- The prototyping took less than two weeks – and we were really impressed by the outcome
- We decided to use MBT and Conformiq for real in a “live” project
- A team was put together, and some training within modeling and the Conformiq tool took place
- A suitable new function was selected
- We just did it...

# MBT - Introduction

---

## Considerations when deploying MBT

- Requires stability and maturity
  - the SUT
  - the organization
  - methods used
  - tools and test framework
- Select an object/function that is suitable for MBT  
*Recommendation:* Use MBT for Functional Testing first, avoid non-functional testing, such as performance test, in the beginning
- The substantial gains will be received when:
  - You have a need for (or benefit of) automated test suites – for example: you run regression tests on a regular basis
  - All testing can be fully automated – and no manual intervention is needed to test your functionality

# MBT - Modeling

---

## Modeling tools

### With Conformiq:

- the Conformiq modeler
- Rational's tool kit
- IBM/Telelogic Rhapsody

Conformiq action language: "QML"/extended Java

Other MBT tools provide other possibilities, for example Smartesting, that supports Borland's Together as well as Rational's tools, and has OCL as action language.



# MBT - Modeling

---

*Recommendation:* Do not use the same model to both generate code *and* generate Test Cases!

Differ between the two different model types:

## Development Model

- Main purpose: generate code
- Technical model
- Often contains implementation information

## Test Model

- Main purpose: generate Test Cases
- Functional model – describing the system behavior of a certain system function (or part of a system function)
- Contains no implementation information
- Black box model

# MBT - Modeling

---

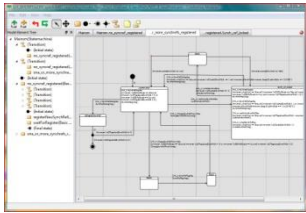
When designing your MBT model you can have two different approaches:

- Design your model from scratch – independently from other existing models
- Reuse and adapt other existing development models:
  - Customize the model to your needs
  - Simplify the model
  - Strip all irrelevant information from the model
  - Make sure that it is black box, and describes the system behavior in a proper way

# MBT - Modeling

One possible approach:

High level system model

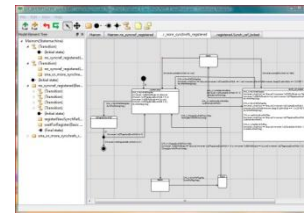


Black box

Responsible:  
System department

Model  
adaptations

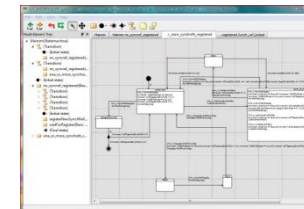
Test Model



Stripped model  
Black box  
Responsible:  
Test department

Further model  
development

Development  
Model



Detailed model with  
implementation  
information  
“White box”  
Responsible:  
Design department

# MBT - Modeling

---

Conformiq Requirements/Modeling Requirements

- a key concept, and a very useful concept to us

A ModelingRequirement is a requirement on the MBT tool, that a certain "event" or "situation" must be covered in (at least) one Test Case. It does not necessarily refer to the functionality provided by the SUT.

A ModelingRequirement is introduced in the model as a text string, where the event/situation has been covered.

Example: *"Synch ref type 1 is active / Register new type 2 synch ref / with lower priority than the active synch ref"*

The Modeling Requirements can also be used for traceability to the "real" (customer) requirements, e.g. with simple tagging.

"Requirement coverage" was for us also the best indication of the quality of the outcome from the Test Case generation.

We found it very beneficial to define all the Modeling Requirements early on - before you start designing the actual model.

# MBT - Modeling

---

## Select Suitable Functionality

### Technical criteria to consider

- There is a need for/benefit of having automated test suites
- It is possible to fully automate the testing of the functionality covered by the model (and to do so in a practical way)
- No manual interaction will be required during test execution
- A proper abstraction of the functionality should result in a fairly small model
- The test scope should not be too small
- The individual Model Requirements should not take too long time to verify (automatically)

Other criteria must be considered as well, e.g. strategic, project wise and practical criteria.

# MBT - Modeling

---

Considerations regarding competence needed

- Compared to traditional test methods, MBT is a complete paradigm shift
- New competences are required, and training must be taken care of
- New roles must be established within the test organization, especially the model designer/"test architect"
- The testers must learn to *not* think as testers. When designing the model, the tester shall not think in terms of Test Cases – the tester should, ultimately, only consider the system behavior
- The tester must have a thorough understanding of the functionality, and be involved in (and contribute to!) the development project already in the early stages

Working with MBT has increased the motivation among the testers. Most testers consider MBT as challenging, inspiring, and Fun.

# MBT - Modeling

---

Considerations regarding the MBT Way of Working

- MBT is well suited for pair modeling/programming
- It is recommended that the model is designed in small increments
- Existing methods for reviews and inspections might not be suited for MBT – new, and hopefully more efficient, methods must then be invented
- Keep it simple

Modeling is all about abstraction!

# MBT – Test Case Generation

---

So far, Conformiq is the only MBT tool we have been using. A thesis work to study other MBT tools was carried out, and a couple of possible alternatives to Conformiq were pointed out, for example Smartesting and Elvior MOTES. However, no detailed case studies were carried out, and evaluation licenses were hard to come by when we needed them.

The first couple of years we only applied MBT for functional testing. Lately have seen possibilities to use MBT also for non-functional functional testing, such as performance tests. We have developed a method for doing this, that we have now started to deploy.



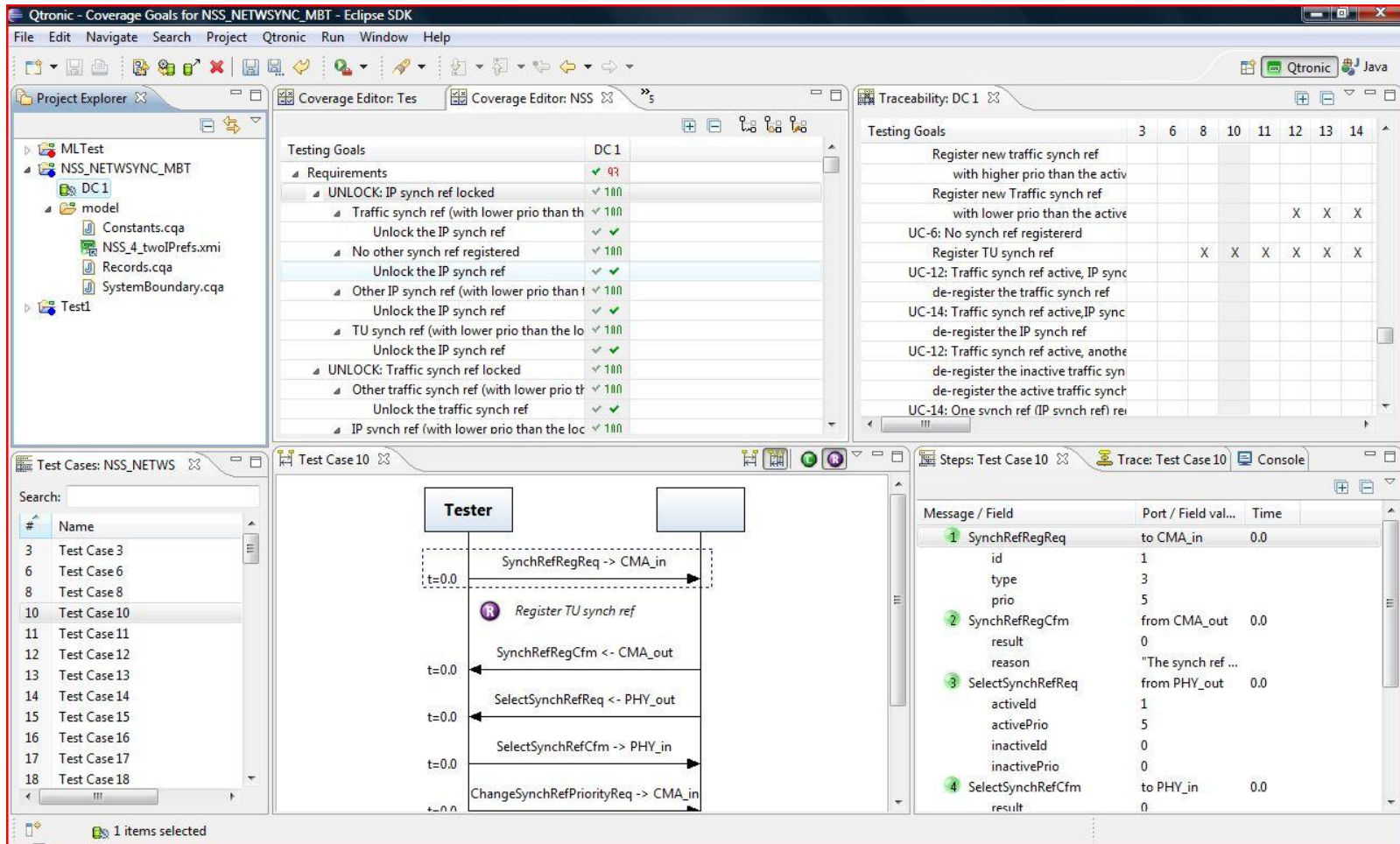
# MBT – Test Case Generation

---

## Specifics for Conformiq

- Runs on either Linux or Windows (and we have tried out both – no major differences in Conformiq's performance have been observed)
- Integrated with Eclipse, but also available as stand-alone product
- General perception: Conformiq is easy to use, and the support from the provider has been very good

# MBT – Test Case Generation



The screenshot displays the Qtronic Eclipse SDK interface for MBT test case generation. It is divided into several panes:

- Project Explorer:** Shows the project structure for 'MLTest' and 'NSS\_NETWSYNC\_MBT', including a 'model' folder with files like 'Constants.cqa', 'NSS\_4\_twoIPRefs.xmi', 'Records.cqa', and 'SystemBoundary.cqa'.
- Coverage Editor:** Displays 'Testing Goals' for 'DC1' with a list of requirements and their coverage status (e.g., 'UNLOCK: IP synch ref locked' is 100% covered).
- Traceability:** A table showing the mapping between testing goals and test cases across various scenarios.
- Test Cases:** A list of generated test cases, with 'Test Case 10' selected.
- Sequence Diagram:** A UML sequence diagram for 'Test Case 10' showing interactions between a 'Tester' and an external component. The diagram includes messages like 'SynchRefRegReq -> CMA\_in', 'SynchRefRegCfm <- CMA\_out', 'SelectSynchRefReq <- PHY\_out', 'SelectSynchRefCfm -> PHY\_in', and 'ChangeSynchRefPriorityReq -> CMA\_in'. A specific goal 'Register TU synch ref' is highlighted.
- Steps / Trace:** A table showing the execution steps of the test case, including message details like 'SynchRefRegReq' with fields 'id', 'type', and 'prio'.

# MBT – Test Case Generation

---

## First Experiences

- The model must be good if the output shall be good!
- The user documentation could be better, especially considering Design Rules and Best Practices
- As an inexperienced modeler it is easy to introduce faults in your model – and the debugging support available in the Conformiq tool kit was not very extensive
- **The generation times can be long**, and sometimes extremely long
- It is very difficult to predict how your model will affect the generation times
- There are clear benefits by optimizing the model from a generation time point of view

# MBT – Test Case Generation

---

Some hints on how to design your model to get the best possible output and shorter generation times

- Design the model in small increments, and generate often
- Learn how the tool works by working with it, and by experimenting (e.g. with the settings) and share the knowledge you gain
- Review your model often, and pay close attention to logical faults as well as pure modeling faults, such as “leakage” and “model holes”
- Also review the model by reviewing the generated Test Cases

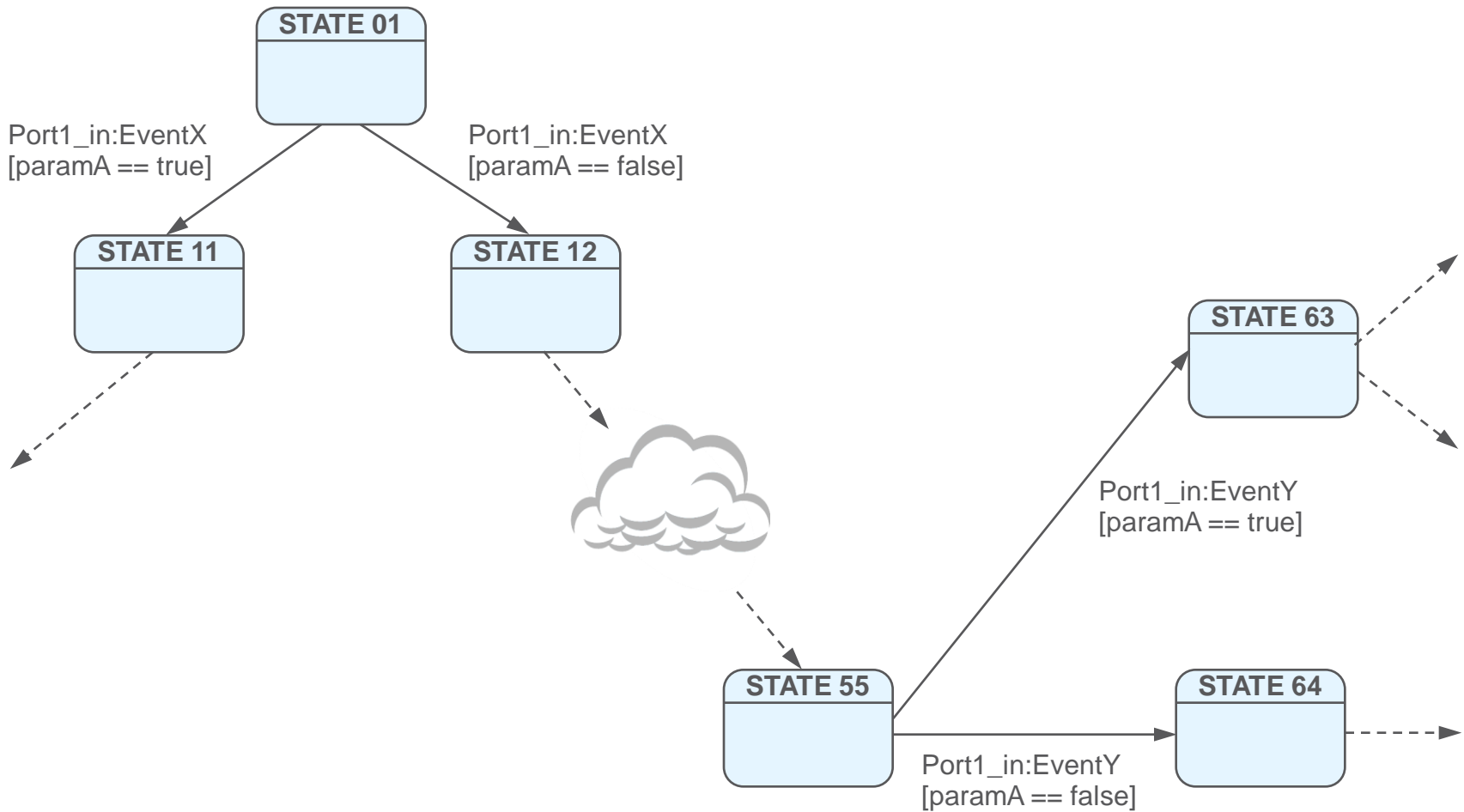
# MBT – Test Case Generation

---

Some hints on how to design your model (continued)

- Do the tool settings carefully, especially the coverage criteria and the “look-ahead depth”
- Always start generating with the lowest possible look-ahead depth, and do not increase it to anything higher than absolutely necessary
- **Avoid parameter combination explosion!**
- **Avoid extreme model depth!**

# MBT – Test Case Generation



# MBT – Test Case Generation

---

Note! There are methods to improve the generation times (e.g. by decreasing the model depth or reducing the number of parameter combinations) but these are not properly documented.

During the time that we have been using Conformiq, the tool in general, and the generation times in particular, have been constantly improved.

Conformiq have given swift and professional support, and listened to our feed-back.

With later versions of Conformiq we got the possibility to distribute the calculations over multiple CPUs and hosts, which will reduce the generation times considerably.

The debugging support has also improved somewhat.

# MBT – Test Case Generation

---

The output from the Test Case generation

- The generated test suite consists of a number of Test Cases, which in turn contain a number of test steps
- The Test Cases are easy to read, follow and understand
- One Test Case can cover more than one Modeling Requirement
- One Modeling Requirement can be covered in several different Test Cases - this is sometimes a drawback, since even “not so important” requirements, or requirements that take long time to verify, can be included many times in one test suite

A way to prioritize the Modeling Requirements would be very useful



# MBT – Test Case Generation

---

The output from the Test Case generation (continued)

- Progress reporting regarding how many Test Cases that have been passed, is no longer relevant - instead we have chosen progress reporting based on the number of verified Modeling Requirements, which works at least as well
- We get “better” Test Cases, in terms of being longer and more complex
- We have found several faults in the SUT that we would not have found using our traditional methods

# MBT – Test Case Generation

---

## Rendering the output

- Conformiq provides some plug-ins for rendering the generated Test Cases into, for example, automated test scripts, html documents and test specifications in word
- For script generation Conformiq currently provides plug-ins for TCL, TTCN-3, Java (“JCAT”) and Perl  
You can also design your own plug-ins if you have other needs
- You also need to design your own “test harness” (or “glue” or “library”) to be able to execute the generated scripts in your own test framework  
The size of this task can vary and depends partly on the model and the test framework  
For our first models the task to design this library grew much bigger than originally planned

# MBT – Test Case Generation

---

The generated test scripts

- The generated test scripts have a good structure, and are also easy to read, follow and understand, and they are easy to edit, which is good for troubleshooting purposes
- Furthermore you can easily build your own Test Cases using the structure from the generated test suite

# MBT – Conclusions

---

**MBT has been a Success Story for our organization and is now our main Way of Working!**

# MBT – Conclusions

---

Why did we succeed?

- We had a *vision*
- We were *willing to take the risk*
- We were *committed*

# MBT – Conclusions

---

## Hard experiences

### ➤ Cost

Estimated gains of total test project lead time: 20-30% when new model is created

### ➤ Re-usability

Models, and parts of models, can be re-used to a much higher extent than originally anticipated

When possible to re-use model parts, the gains can be much higher than 20-30%

### ➤ Coverage

Faults found that we would not have found with traditional test design methods

# MBT – Conclusions

---

## Hard experiences (continued)

### ➤ Quality

No decrease in the quality of the tested product has been observed

## Soft experiences

➤ Most testers think of MBT as a very interesting way of working, and are eager to learn

➤ High motivation among the testers that have been working with it – and everyone think is Fun (most of the time)

➤ Not all testers are suited for working with modeling

# MBT – Conclusions

---

Final recommendations

If you are a test organization,  
and if you benefit from having automated test suites:

**Try out the MBT way of working!**

But

- Don't go for full deployment from the start
- Start with a smaller, well defined, well encapsulated, area/functionality
- Do it yourself

And

- In the beginning: Stay away from functionality where you suspect you cannot avoid models with parameter explosion or great depth





**ERICSSON**