Using Model Based Testing for Robustness Tests

Håkan Fredriksson
Ericsson AB
hakan.fredriksson@ericsson.com

Agenda

1. Introduction
2. Background
3. System Under Test
4. Needs
5. Execution
6. Outcome
7. Evaluation

Introduction

Nothing revolutionary

Not very innovative

Actually no big deal at all

But still…

Introduction

We managed to

- reduce project lead time, and
- perform a more thorough testing of the robustness than we normally do

by utilizing existing tools and competence in a way that was new to us

Evolution…

Background

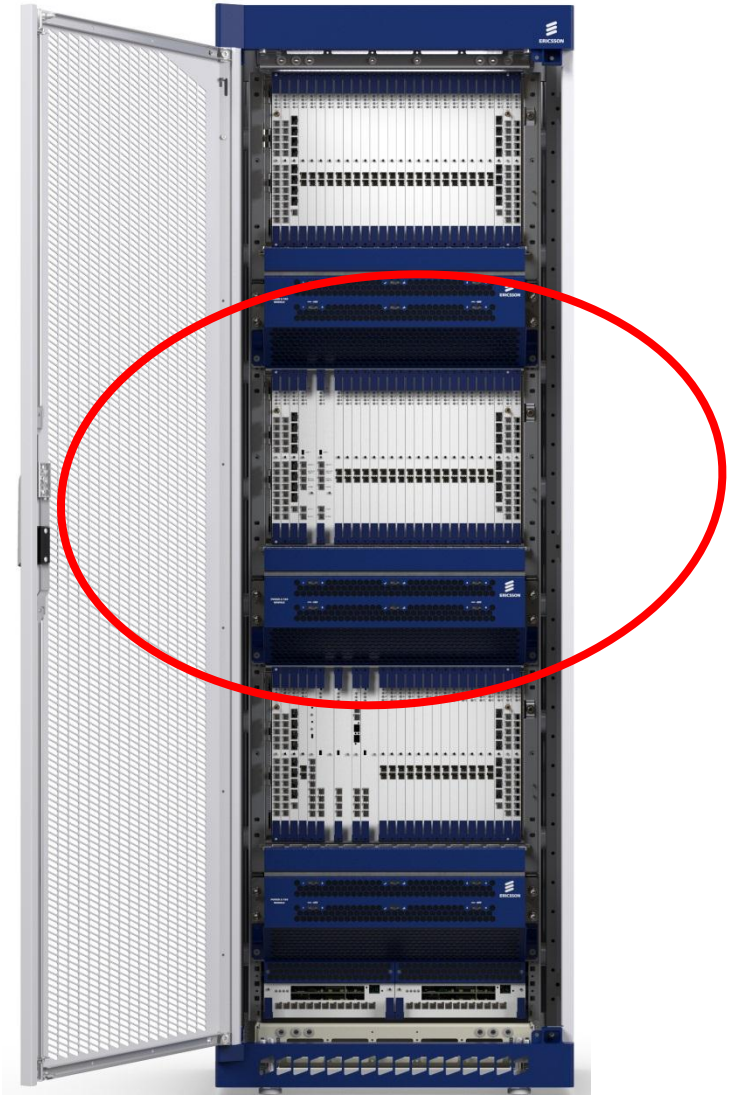
- A test organization that started using MBT 2007, and deployed it shortly afterwards, for functional testing
- Have procured licenses for commercial MBT tools
- The MBT way of working is now fully integrated in our processes and our test framework
- Main purpose of usage, so far:
  To generate *optimized* automated test suites to be used during the development project as well as in future regression testing activities

Background

- Lately the usage of MBT in the development projects has declined a bit, probably due to the introduction of more agile ways of working (e.g. scrum)
- The licenses we have are quite expensive, and the capacity is far from fully utilized
- We have a wish to extend the usage of MBT
- We have started to use MBT also for “lower level testing” (i.e. Component Testing, Multi Component Testing, etc.)
- We have also worked out methods and processes to use MBT for verification of some non-functional requirements

System Under Test

- We are testing the platform parts of a system that consists of HW as well as SW
- In this particular project the SUT consisted of just one magazine/subrack, with a number of different boards



Needs

- In this particular project a lot of new products were introduced, HW as well as SW
- Therefore we also had a need to run extensive robustness testing of the system
- One typical aspect of robustness: for the system to handle restarts of one or more boards, and become operational and stable again after the restarts

Needs

- Our aim: to run a fully automated test suite with loads of test cases, where each test case is different from the others
- Each test case shall contain at least two consecutive restarts, with a delta-time in between
- Parameters that varies:
  - which boards that restarts
  - the type of the restart
  - the time between the restarts
  - number of consecutive restarts

Needs

- But to write such a test suite manually would take long time (and be quite e tedious task)
- And since we already have an infrastructure to generate executable test suites, i.e. our MBT solution, why not use that…?

- So we did

Execution

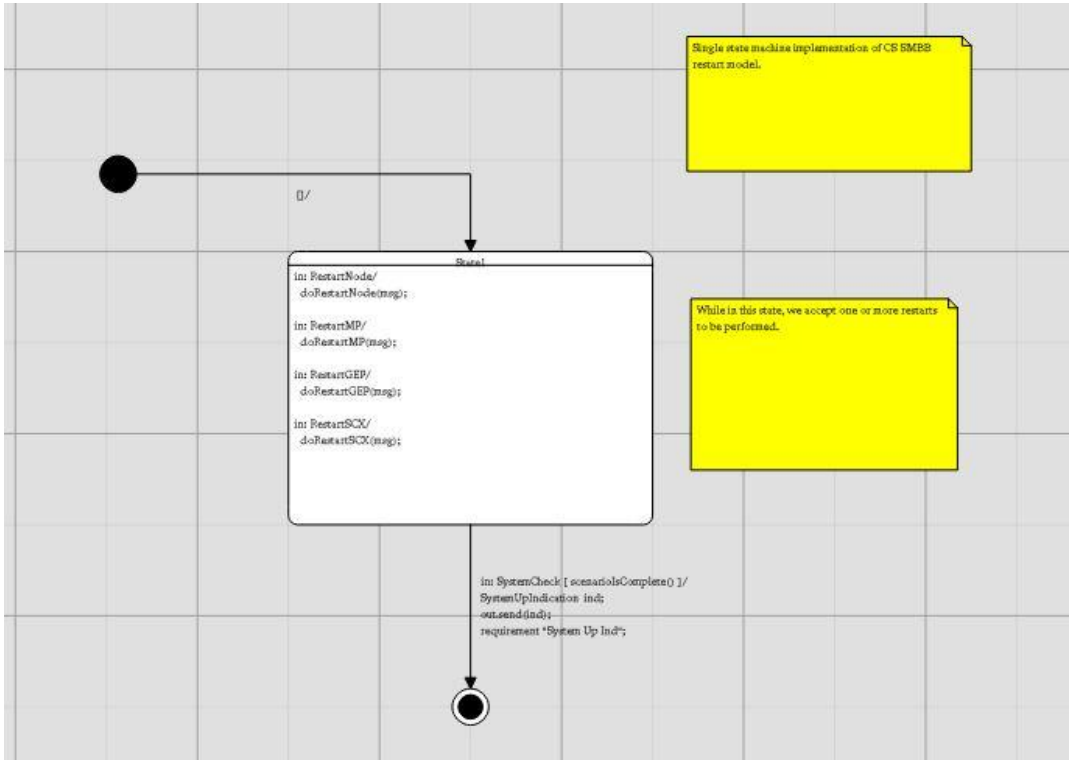
- Two very experienced modelers went ahead to produce one model each
- Main differences from our previous MBT approach:
  - the test suite does not need to be optimized
  - the test suite will only be executed a few times during the development project, and *not* used for future regression testing
- The solution also need to be flexible, with clear and accessible “settings”

Execution

- The resulting models were both very simple, easy to understand and work with and fulfilled our initial needs
- Each modeler spent less than a day to produce the model
- The two modelers used the same MBT tools (Conformiq) but concentrated on different modelling features

Outcome

Model 1 – Requirement driven



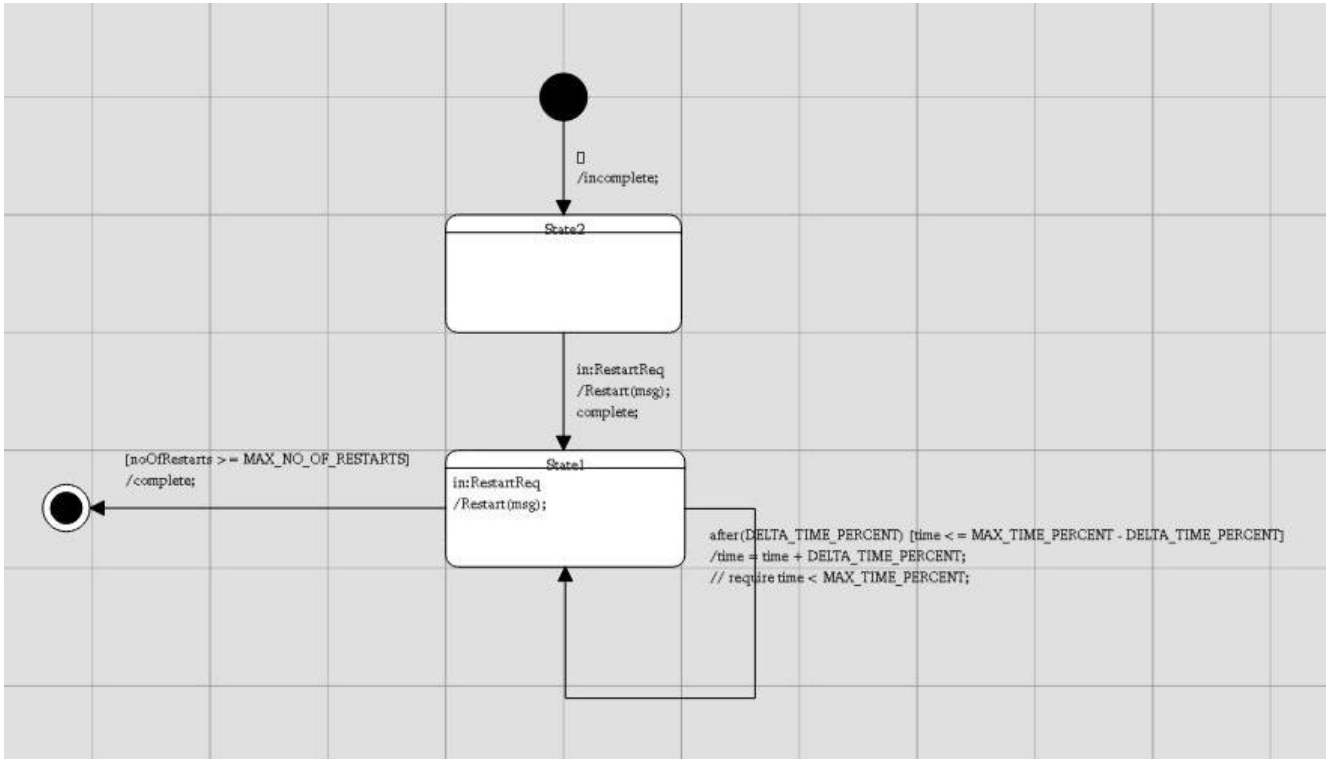
Outcome

Model 1

- Includes requirements of the most important scenarios
- 15 requirements, and basic settings with no more than 2 restarts per test case => 23 generated test cases
- Not much variation in the delta time combinations
- It is easy to change the settings, and also to generate test cases with more than 2 restarts, but it will not lead to many more test cases than in the basic case

Outcome

Model 2 – Combinatorial approach



Outcome

Model 2:

- Utilizing the MBT tool feature to generate with “all combinations”
- With basic settings and no more than 2 restarts per test case we got 500-700 generated test cases (which would still be feasible to execute over a weekend)
- Increasing the number of consecutive restarts to more than 2 led to impractical long test suites
- Removal of scenarios that are not relevant needs to be taken care of (preferably in the model itself)

Outcome

- Reflection regarding model 2:

  Do we really need an MBT tool to generate all combinations?

  Could we not just write a script that does that instead?

  Probably.

  But.

  The model files do give us visibility, flexibility and control of what is happening, and it is convenient to base the test design on the existing infrastructure for test automation.

Outcome

Other differences between the two modelling approaches:

- Model 1 has one unique restart event for each board, while model 2 has one generic restart event with the board as a parameter
- Model 1 has absolute values for the delta times, while model 2 has relative
- Waiting for the delta time timeout is taken care of by an MBT tool built-in construct for model 2, while delta time is an event parameter in model 1

Outcome

- For both models the generation time varies, and depends primarily on the number of chosen consecutive restarts and the settings (both tools settings and model settings)
- The generation times were never considered as inconveniently long (except in the cases where we got parameter combinations explosion…)

Outcome

- Model 1 was chosen, for a start
- Adaptions to the test framework were written
- Executable test cases were generated and executed
- The result of the test execution was evaluated
- When suddenly…

Outcome

…the current project, and the used technical solution, was terminated…!

…but a couple of months later a new project, with a new technical solution, rose from the ashes

From a test point of view the preconditions for the new technical solution were not that different from the original project

Outcome

- We thus decided to use the same approach regarding robustness testing for the new project
- And it turned out that it was very easy to adapt the existing models to the new circumstances

- And that's just about where we stand today

- But have we found many faults then, so far?

  Not really, but we feel that we have verified our SUT's robustness

Evaluation

- Our needs were fulfilled
- If you have an existing infrastructure for MBT, it is fully possible, as well as beneficial, to use it also to generate this type of test suites for robustness testing

Evaluation

- On the plus side:
  - It is fast
  - It is easy
  - Only a thin harness to the existing automation framework is needed
  - It is flexible
  - It is reusable
- But beware:
  - Competence within the MBT area is required
  - When working with combinations, there is always a risk for “explosions”

Way Forward

- Expand the usage for the current project, and also cover interference between system functions and restarts of individual processes/programs
- Possible further expansions: to use this approach for other types of robustness testing as well, not just restarts, for example link failures, HW faults, etc.
- Possible elaboration of current model: instead of delta times, introduce “trigger points” (e.g. specific states during the restart)

Way Forward

- And of course we aim to use this approach in any future project with similar needs and prerequisites
- And it would also be interesting to perform trials with other MBT tools...

ERICSSON