Application of Model-Based Testing on a Railway Project

Keywords: Model-Based Testing; CBTC, testing process, train protections, track topology, automatic test script generation, automatic test cases generation.
Authors: Mathieu Steiner, Anthony Faucogney.

ABSTRACT

This paper discusses an application of Model-Based Testing for the test of a "Communication-Based Train Control" functionality. We will detail how MBT allows engineers to focus on the test campaign by modifying the testing process. A comparison between the old and the new MBT approaches will thus be realized and every stages of the process further explained.

MODEL-BASED TESTING

Model-Based Testing is the application of Model-Based Design for software testing; this concept has been adopted by many companies and integrated into both hardware and software systems testing. The process typically consumes functional requirements as inputs, and produces test cases. The test cases generation is done according to a model describing the required behavior of the system. The testing effort is made by test engineers during the design of the usage model. The effectiveness of Model-Based Testing is highly increased with automation, i.e. the capacity to automatically translate test cases into executable test material.

This paper deals with a concrete application of the MBT process in the railways industry. The MBT tool chosen is a tool from the market using Markov Test Logic and based on statistical usage testing. The tool allows the design of usage models and the generation of test cases according to several native generation algorithms.

We will first deal with the functionality under test and the old testing method. Then, we will get deeply into the new testing approach with MBT and the new testing process.

APPLICATION OF MBT ON A RAILWAY PROJECT

FUNCTIONALITY UNDER TEST

We recently have been putting the MBT concept into application in the railways industry in order to test part of a "Communication-Based Train Control" system. CBTC is an automated control system that ensures the safe operation of rail vehicles using data communication between various control entities that make up the system. The functionality under test was the "management of the protections surrounding trains" – aka M-function – which is handled by a software module running on a side-track hardware system. Each train on a line is monitored, and has a surrounding automatic protection that ensures the safe operation of the train. The tested M-function is in charge of creating, deleting and updating those Automatic Train Protections – aka as ATPs. The outputs produced by the function are mainly the type of the protection, and its coordinates.

Such functionality is complex. It actually handles different types of trains, and multiple types of protections. Additionally, the track topology possibilities are almost infinite. This altogether leads to a huge number of specific rules, and complex specifications that we had to deal with.

OLD TEST CAMPAIGNS AND EXISTING TESTING FACILITIES

Old test campaigns have been designed in a classical way, and test cases written by hand. These test cases are composed of moving trains, crossing track elements – e.g. blocks, sections, points... Testing of the M-function was usually performed by simulating the progression of a train on a known track, mostly inherited from real track topologies. Old test cases are thus a quick succession of single testing contexts – aka STCs – e.g. a train crossing an uncontrolled point. Test cases complexity is thus limited to the test Engineer imagination and the complexity of the chosen track topology. It means that modifications of the track topology are almost impossible because time-costly and interaction tests are not performed – not even formalized.

Testing facilities are fully custom. The hardware module that runs the software under test is connected through a network to simulators. Every hardware ECUs are simulated; onboard and side-track equipment. Simulators are piloted by scripts, sending messages to the module under test.

"WAR FOG" NEW APPROACH

Our objective was to propose a new approach that would allow:

- Easy modifications of test campaigns
- Easy updates of a test campaign consecutively to specifications modifications
- Track topology generation
- Enhanced reuse

Thus, a new approach, called "war fog" approach has been tried: we actually modeled a step-by-step progression of the train. This implies that the track topology is not fixed; it is different from one test case to another. Test cases are still a succession of smaller testing contexts, but they are not limited to a specific track topology. Each step of the train progression is a statistically drawn combination of a specific management rule and a specific track element. This clearly results in more complex test cases.

NEW TESTING PROCESS WORK FLOW

The testing process has been modified to take the new approach into account. The new work flow can be summarized as follow:

A. STC identification
B. STC review system Engineers
C. STC modeling
D. Model completion with script automation information
E. Test cases generation and execution on testing facilities
F. Analysis and testing coverage

We shall now dig deeper into each process step.

STC IDENTIFICATION

Our first task was to identify and characterize as many single testing contexts as possible from the system/software specifications:

- Identify and name STCs
- Identify which STCs can be chained up
- Determine system expected behavior for each STC
- Document all STCs

Here follow some examples of STC that we identified:

- A talkative train is moving and crosses a controlled point that becomes uncontrolled.
- A non-talkative train is moving and faces a talkative train
- A non-talkative train starts emitting position information – i.e. becomes talkative

This new work flow allows us to identify STCs that had never been tested in previous test campaigns. An additional interesting aspect of the way of working is that it allows work with a non-exhaustive list of STCs. STCs can actually be iteratively added later in the process.

STC REVIEW

The second part of the process is to review the STCs documentation with system Engineers; we especially want to check that the expected behavior is matching the system point of view. We actually have been able to diagnose some specifications issues at this point of the process.

STC MODELING

The modeling of the functionality has been realized with a tool called "MaTeLo", based on Markov chains. We do not concentrate on functionality design when writing such model, but on test cases design. The model consists in transitions and state – it actually looks like a state diagram.

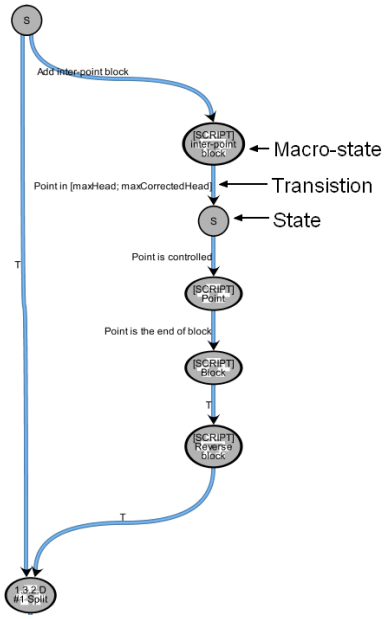

FIGURE 1: MATELO MODEL EXAMPLE

- Transitions wear stimulations to apply to the system, and SUT expected behaviors
- States can be seen as SUT stable test states

The model is designed in branches. Each branch contains at last one STC, in which we check ATPs properties. Some branches are possibly chained with others branches if the train can go on. Each branch contains track elements and train movements necessary for the STC. Such model can be easily reviewed by peer, even in the case they don't know the tool.

MaTeLo is based on statistical usage testing. Each transition has a specific frequency of being drawn – aka as probability or usage frequency. Additionally, each stimulation domain is statistically described with a repartition law. Such probabilities are not unique for a specific model; there can actually be as many probability layers as wanted. Test cases generation is done according to one of this layer, also called "profile".

Here follow some usage profiles:

- Only talkative trains (no non-talkative trains on the line)
- Track topology with only points (no other track elements)
- Any project specific profile in order to match with a real track topology

MODEL COMPLETION WITH SCRIPT AUTOMATION INFORMATION

After having designed the model, we can add associations on transitions to script test cases to any proprietary language. In our case, the test scripts do not follow the same flow that the test cases which is a restriction inherited from the test facilitie. So that the model is producing intermediate file:

- Track topology (e.g. sections, points, blocks…)
- Train moves
- Dynamic events (e.g. points state changes…)
- Test case log file

The model is used to output 9 intermediate files for this project.

TEST CASES GENERATION AND EXECUTION ON TESTING FACILITIES

MaTeLo test cases generator is able to generate test cases from the model according to several algorithms. Four algorithms are currently available; we mostly used the “Arc coverage” algorithm that allows the generation of the smallest test suite possible while covering the whole model. The test cases generator produces test cases and intermediate files that are parameterized concatenation of the script API. The resulting model intermediate files are then formatted by an external tool coded in python to match the test facility specific format. Test scripts re-formatted by the python executable are directly able to run on the test facility.

ANALYSIS AND TESTING COVERAGE

Analysis of failing test cases was done by hand. In regards to functionality, we are able to generate a test suite covering 100% of the model branches and thus 100% of the identified STCs. By adding some new profiles, we can generate more test cases and enhance the test coverage and reliability of the system.

CONCLUSION

The MBT method has modified the testing process. The process flow is more structured and allows the detection of specifications errors in its early stages. Additionally, it allows the identification of new test cases by requiring an increased exchange with the system specification team and allows the automatic generation of test scripts. By using probabilities profiles, the test designer is thus able to generate as many test cases as necessary to ensure an enhanced coverage and reliability of the tested functionality.