Using Topcased for Model-Based Testing

Frédéric Fondement[1], Pierre-Alain Müller[1], Brice Wittman[1], Jonathan Lasalle[2], Emilie Oudot[2],
Fabrice Ambert[2,3], Fabrice Bouquet[2,3], Bruno Legeard[2,3], Fabien Peureux[2,3], Marc Alter[4] and Claude Scherrer[4]

1: MIPS - UHA
12, rue des Frères Lumières
68093 Mulhouse Cedex

2: LIFC - UFC
16, route de Gray
25030 Besancon Cedex

3: Smartesting
18, rue Alain Savary
25000 Besancon

4: Clemessy
BP 2499
68057 Mulhouse Cedex 2

The Smartesting solution is a tooled testing approach to generate and manage functional tests from behavioural models specified with a subset of UML/OCL notation (called UML4MBT [BGL+07]). This test generation tool chain is based on Topcased modeller v3.3.0 (in addition to "Together Borland" and "IBM RSM/RSA") to specify the system under test, and the test engine Smartesting Test Designer™ for the tests cases generation and requirement traceability. The test generation with Test Designer™ allows covering automatically all behaviours of the model (transition coverage) and semi-automatically some executable sequences (scenario-based coverage). Within VETESS project [VET10], a dedicated Eclipse-based plug-in was developed to customize and extend Topcased UML and SysML modelling capabilities, and to connect it to Test Designer™ v3.4.2. The use of Topcased to develop this toolchain was motivated by the goal of the VETESS project, which was to address embedded system test generation issues from UML/SysML models [FMW+10].

A UML4MBT model has to contain one UML class diagram to represent the static view of the system (with classes, associations, enumerations, class attributes and operations) and one UML object diagram to list the concrete objects used to compute test cases and to define the initial state of the system (it must be an instantiation of the associated class diagram). The dynamic view can be modelled by OCL expressions on pre or postcondition of class operations and/or by a statemachine diagram (annotated with OCL constraints). The latest has also some restrictions; it cannot contain parallel states, historic states, fork and join states or trans-hierarchical transitions.

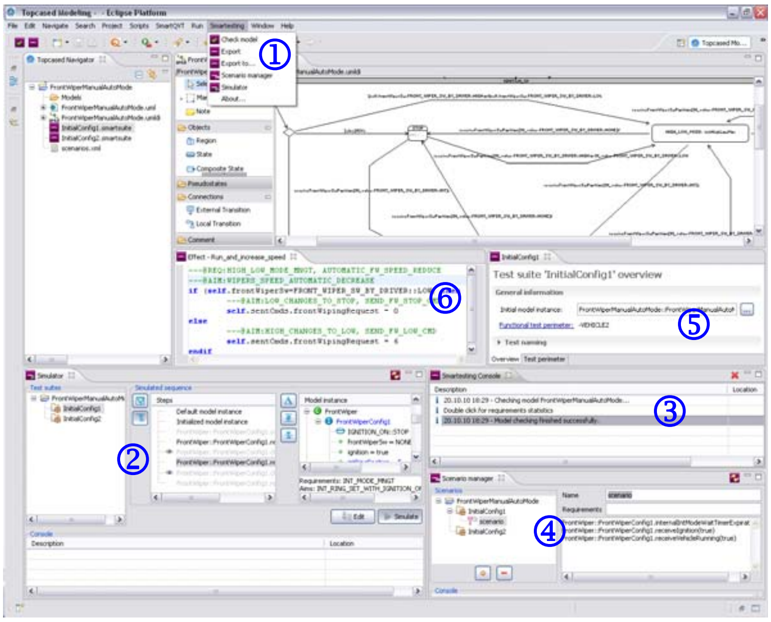
The Topcased plug-in functionalities can be separated in verification part and test generation part. The verification part allows checking the correctness of the models by the following functionalities:

1. Syntactic verification (item 1 in Fig. 1) that checks that the UML model satisfies UML4MBT restriction and if it contains syntactic errors.
2. Functional verification of the model is done by a simulator that animates the model to check manually the correctness of modelled behaviours (frame 2 in Fig. 1).

If the syntactic verification does not find any errors (frame 3 in Fig. 1), it is possible to use following functionalities addressing the test generation process:

1. The scenario manager that allows to create scenarios (sequences of operation calls) to define new test cases (frame 4 in Fig. 1).
2. Definition of tests suite to split tests targets (frame 5 in Fig. 1).
3. Export the test model to a file based on the metamodel of UML4MBT (which is the input interface of Test Designer™) (item 1 in Fig. 1).

Moreover, a proprietary OCL editor is embedded into the Topcased plug-in to help engineer specifying OCL constraints (syntactic colouring, auto-completion…) (frame 6 in Fig. 1).



```
---@REQ:HIGH_LOW_MODE_MNGT, AUTOMATIC_FW_SPEED_REDUCE
---@AIM:WIPERS_SPEED_AUTOMATIC_DECREASE
if (self.frontWiperSw=FRONT_WIPER_SW_BY_DRIVER::LOW) then
        ---@AIM:LOW_CHANGES_TO_STOP, SEND_FW_STOP_CMD
        self.sentCmds.frontWipingRequest = 0
else
        ---@AIM:HIGH_CHANGES_TO_LOW, SEND_FW_LOW_CMD
        self.sentCmds.frontWipingRequest = 6
endif
```

Test suite 'InitialConfig1' overview

General information

Initial model instance: FrontWiperManualAutoMode::FrontWiperManualAutoM

Functional test perimeter: -VEHICLE2

Test naming

Overview | Test perimeter

```
20.10.10 18:29 - Checking model FrontWiperManualAutoMode...
Double click for requirements statistics
20.10.10 18:29 - Model checking finished successfully.
```

Requirements: INT_MODE_MNGT
Aims: INT_RING_SET_WITH_IGNITION_ON

Name: scenario

```
FrontWiper::FrontWiperConfig1.internalIntModeWaitTimerExpirat
FrontWiper::FrontWiperConfig1.receiveIgnition(true)
FrontWiper::FrontWiperConfig1.receiveVehicleRunning(true)
```

Figure 1: screenshot of the Topcased modelling platform customized with VETESS plug-in.

During French project VETESS, this tool chain was adapted to SysML models in order to generate tests for embedded systems. The subset of SysML, called SysML4MBT [LBL+10], defines which elements are taken into account. A SysML model has to contain one block definition diagram to represent the static view of the system (with blocks, associations, compositions, enumerations, blocks attributes, operations, signal and flow ports), one internal block diagram to represent interconnections between blocks and one or more statemachine diagrams to represent the dynamic view of the system. SysML4MBT statemachine can contain much more elements than UML4MBT statemachine. Then, fork/join, historic and parallel states are allowed. About OCL, the circumflex (^) is now allowed in OCL constraints for SysML4MBT model. This element enables to send signal to an other block.

The plug-in was adapted for SysML4MBT models. The syntactic verification was transformed to check SysML4MBT models. The definition of tests suite is the same than in UML4MBT. The export of the model to a file-based on the SysML4MBT metamodel was developed. The input file of Test Designer™ is the UML4MBT metamodel-based file. Then a transformation from SysML4MBT interface file to UML4MBT interface file was developed and implemented into the export functionality of the plug-in (specific test generation strategy can require dedicated transformation from SysML to UML: during the export, a checkbox allows to select whether the strategy should be applied or not). Finally, the functional verification and the scenario manager are not yet transferred to this SysML plug-in (for lack of time during VETESS project).

Presented toolchain, hence part of the Smartesting commercial solution, has been successfully experimented on two real-life applications dealing with automotive front wiper and steering column managements. It is currently used and extended by Smartesting to achieve model-based testing issues within the VERDE project [VER10].

[BGL+07] F. Bouquet, C. Grandpierre, B. Legeard, F. Peureux, N.Vacelet and M. Utting. A subset of precise UML for model-based testing. In Proceedings of the 3rd International Workshop on Advances in Model Based Testing (A-MOST'07), pages 95-104, London, UK, July 2007. ACM Press.

[FMW+10] F. Fondement, P-A. Muller, B. Wittmann, F. Ambert, F. Bouquet, J. Lasalle, E. Oudot, F. Peureux, B. Legeard, M. Alter and C. Scherrer. VETESS : IDM, Test et SysML. Génie Logiciel, no 93, p. 43--48, Juin 2010. Selected paper from the 7th NEPTUNE Workshop.

[LBL+10] J. Lasalle, F. Bouquet, B. Legeard and F. Peureux. SysML to UML model transformation for test generation purpose. In Proceedings of the 3rd ICFEM International workshop on UML and Formal Methods (UML&FM'10), Shanghai, China, November 2010, ACM Press, to appear.

[VER10] The VERDE web site. http://www.itea-verde.org/, 2010.

[VET10] The VETESS web site. http://lifc.univ-fcomte.fr/VETESS/, 2010.