© QTronic

Continuous, Platform-independent Testing of Functional Requirements

Thousands of kilometres are driven in simulation and with real car prototypes to ensure that all functional, safety and quality requirements are met with enough coverage of the various operating conditions and driving situations. In this paper, Daimler and QTronic present their approach to build an efficient requirement assessment framework which can be reused for the various test platforms and use cases that contribute to a comprehensive and continuous quality assessment. This approach is currently being reused by several powertrain development projects in the Daimler Group.

AUTHORS

Holger Brückmann
is Development Engineer at Daimler AG in Stuttgart (Germany).

Alexander Waiss
is Development Engineer at Daimler AG in Stuttgart (Germany).

Markus Stix
is Team Leader at QTronic GmbH in Stuttgart (Germany).

Dr. Ingo Matheis
is Team Leader at QTronic GmbH in Stuttgart (Germany).

CHALLENGES

Most often, due to limitations of the representation or of the implementation, the requirement models initially developed for tests in MiL/SiL/HiL simulation, cannot be reused later for all use cases that contribute to a continuous quality assessment. For instance, the test drivers often must rely on pen-and-paper checklists to make sure that they performed all the maneuvers that are planned for a test drive. Moreover, without an immediate feedback from the maneuver assessment, they cannot be sure that the car always reached the operating conditions required by the test. This reduces the completeness, respectively increases the costs of the test drives which are iteratively performed during the development. A further important source of data for the safety and quality assessment under real operating conditions are the measurements recorded by data loggers, either from test benches or from test drives. Requirements tested by classical test automation scripts, being designed to test the system reaction only in the context of a predefined test stimulus, cannot be applied for this case either. Multiple redundant requirement assessment models, hard to maintain in-sync with the development, are often a consequence. That causes additional costs and hinders a continuous and efficient requirement assessment during the development process.

The aim is to cover the following relevant use cases with a unique representation and interpretation of requirements:

- online assessment of manual and automated tests performed in simulation
- offline assessment of measurements from data loggers
- online assessment during test drives. This includes the real-time feedback and guidance for the driver to cover the required tests and maneuvers.

The following usability objectives are important as well:

- Requirement models should be easy to read and write.
- Maintenance, debugging and sharing of models should be easy.
- Multiple product variants should be supported.

FORMAL SPECIFICATION OF REQUIREMENTS

A first step toward platform independent testing is to realize that a requirement model should not depend on a particular test stimulus. The requirement models should detect the necessary operating conditions when the associated system reactions are expected to occur and trigger the assessment in all situations when the pre-conditions are met. This way, the test stimulus is separated from the requirement model and the requirement can be evaluated for virtually any stimulus.

The “QTronic Requirement Modelling Language (RML)”, which is part of TestWeaver, provides an easy to use, yet unambiguous specification language that can be translated automatically into executable code, **FIGURE 1**. The language encodes logical and temporal relations which, depending on the statement, either wait for triggering conditions or check expected reactions. This is done for all requirements simultaneously.

FIGURE 2 shows a code snippet of a simple RML Watcher. Note the message string used as a hint for a driver that should test the requirement in the car. RML models can be written with any common text editor. TestWeaver provides, however, also a specialized RML editor with syntax highlighting, syntax checking and code completion. The plain text representation is compatible with any diff-tool or repository tool for version control. Meta information like author, description, requirement ID and the assumed dependence on configuration parameters can be provided as well. For a better overview, the watchers can be structured in several RML files and directories, for instance as functionally related clusters/chapters.

PLATFORM INDEPENDENT TESTING

Three types of setups are currently used for powertrain tests at Daimler:

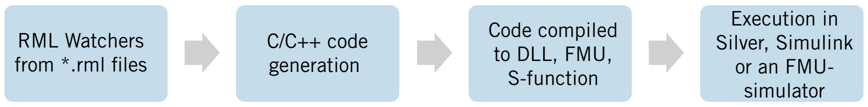
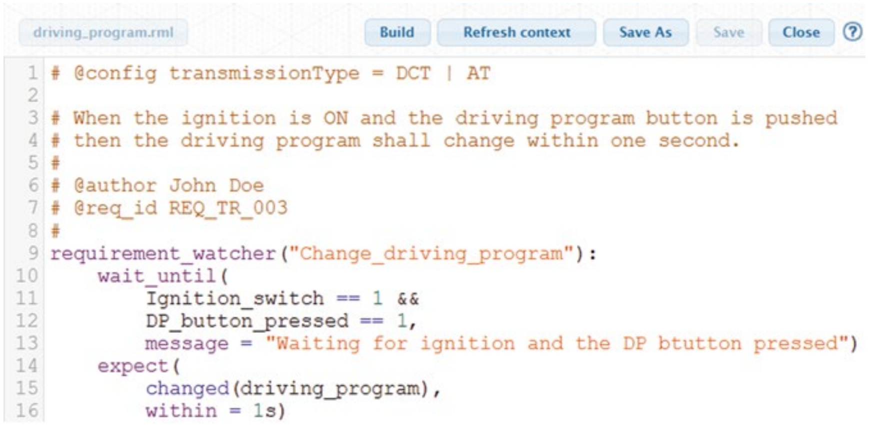

RML Watchers from *.rml files
C/C++ code generation
Code compiled to DLL, FMU, S-function
Execution in Silver, Simulink or an FMU-simulator

FIGURE 1 Compile process of an RML Watcher (© QTronic)



```
# @config transmissionType = DCT | AT

# When the ignition is ON and the driving program button is pushed
# then the driving program shall change within one second.
#
# @author John Doe
# @req_id REQ_TR_003
#
requirement_watcher("Change_driving_program"):
    wait_until(
        Ignition_switch == 1 &&
        DP_button_pressed == 1,
        message = "Waiting for ignition and the DP btutton pressed")
    expect(
        changed(driving_program),
        within = 1s)
```

FIGURE 2 Code snippet of a simple RML Watcher (© QTronic)

- Online tests performed in simulation with single or multiple virtual ECUs: Silver is used to build virtual ECUs and to simulate networks of virtual ECUs on a PC in closed loop with plant models. The compiled RML modules are used for virtual system tests. These are executed automatically after integrating new software versions. The stimuli for these tests originate from a variety of sources: predefined test scripts, recorded test drives, as well as computer-generated stimuli for automatically increasing the test coverage with TestWeaver [1, 2, 3].
- Offline measurement evaluation: Huge amounts of data recorded by loggers or measurement, calibration and diagnostics (MCD) tools can be checked for requirement compliance offline. This can be performed with Silver with the same compiled RML modules. The measurement evaluation usually runs several times faster than real-time on a PC.
- Real-time assistance and assessment of acceptance tests performed in the car: Silver running on a laptop can be connected to the real car communication networks (CAN, Lin, Flexray, Ethernet) via a hardware device. The execution of the compiled RML files is performed in real-time for this use case. Feedback about the executed and the remaining tests is provided to the driver on the laptop, **FIGURE 3**.

All three use cases benefit from the concurrent evaluation of all requirements and the resulting test density. The first two use cases have been in use since a longer time at Daimler AG [1, 2, 3]. The online assessment of acceptance tests performed in the car was the last testing task that was not possible to complete with the same requirement models and assessment tools until now. Therefore, some more details about this setup are given in the following.

An Excel-based frontend documents in real-time the test results and displays hints about the maneuvers expected to be executed in order to complete the untested requirements, **FIGURE 4**. During a certain maneuver all watchers with matching preconditions are evaluated and can report a success or a failure. It is possible that a requirement, that initially reported a successful evaluation, reports a failure at a later evaluation, for instance under differing operating conditions. The success state of a requirement evaluation can thus turn into a failed result later (but not the other way around). Measurement snippets in MDF format are automatically recorded for each requirement watcher that reaches a conclusive evaluation state. The measurements can be inspected later for debugging the identified problems. To support department comprehensive development of requirements, the document is structured in chapters. One chapter holds the requirements of one RML file and is listed in a separated worksheet as well as in the overview. With the "file per chapter" approach, the editorial work can be independently administrated by the responsible test engineers. The acceptance tests are developed and debugged first in simulation on PC and only afterwards used for the car drives.

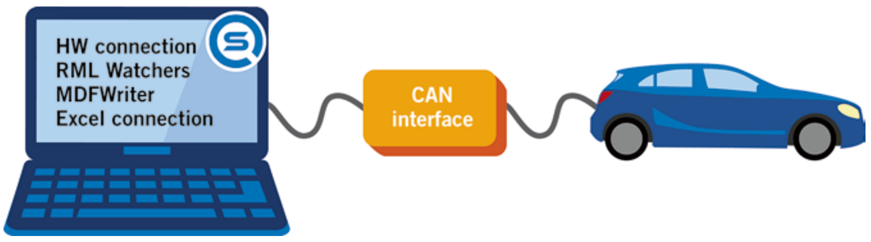
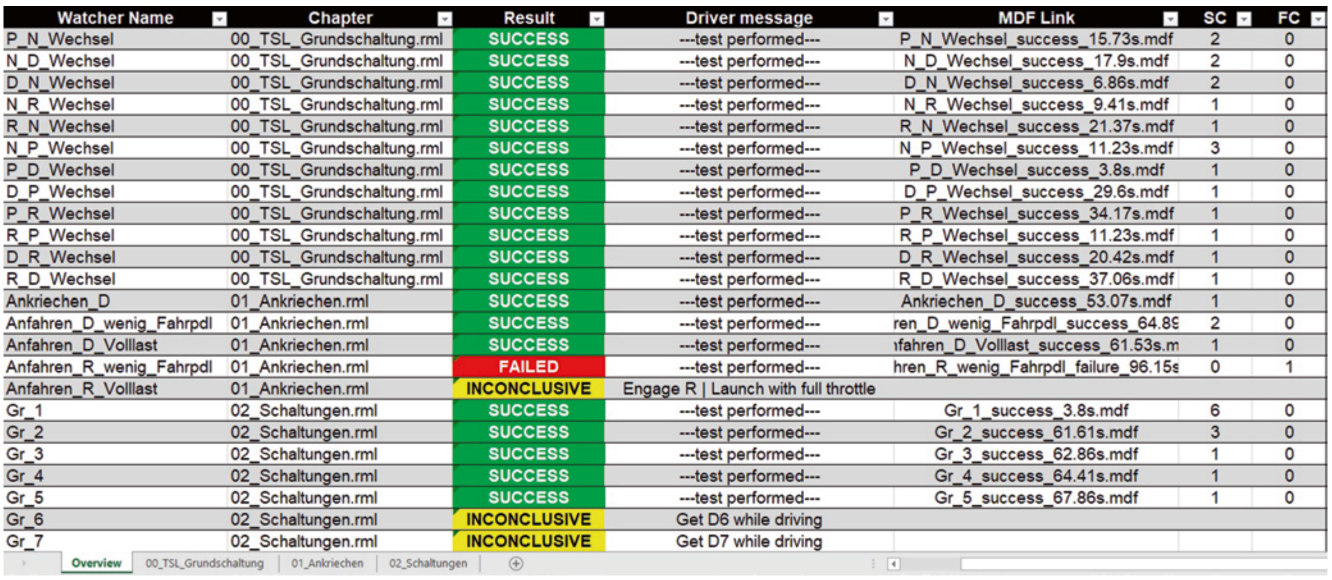


FIGURE 3 Real-time execution of acceptance tests in the car (© QTronic)



Watcher Name	Chapter	Result	Driver message	MDF Link	SC	FC
P_N_Wechsel	00_TSL_Grundschaltung.rml	SUCCESS	---test performed---	P_N_Wechsel_success_15.73s.mdf	2	0
N_D_Wechsel	00_TSL_Grundschaltung.rml	SUCCESS	---test performed---	N_D_Wechsel_success_17.9s.mdf	2	0
D_N_Wechsel	00_TSL_Grundschaltung.rml	SUCCESS	---test performed---	D_N_Wechsel_success_6.86s.mdf	2	0
N_R_Wechsel	00_TSL_Grundschaltung.rml	SUCCESS	---test performed---	N_R_Wechsel_success_9.41s.mdf	1	0
R_N_Wechsel	00_TSL_Grundschaltung.rml	SUCCESS	---test performed---	R_N_Wechsel_success_21.37s.mdf	1	0
N_P_Wechsel	00_TSL_Grundschaltung.rml	SUCCESS	---test performed---	N_P_Wechsel_success_11.23s.mdf	3	0
P_D_Wechsel	00_TSL_Grundschaltung.rml	SUCCESS	---test performed---	P_D_Wechsel_success_3.8s.mdf	1	0
D_P_Wechsel	00_TSL_Grundschaltung.rml	SUCCESS	---test performed---	D_P_Wechsel_success_29.6s.mdf	1	0
P_R_Wechsel	00_TSL_Grundschaltung.rml	SUCCESS	---test performed---	P_R_Wechsel_success_34.17s.mdf	1	0
R_P_Wechsel	00_TSL_Grundschaltung.rml	SUCCESS	---test performed---	R_P_Wechsel_success_11.23s.mdf	1	0
D_R_Wechsel	00_TSL_Grundschaltung.rml	SUCCESS	---test performed---	D_R_Wechsel_success_20.42s.mdf	1	0
R_D_Wechsel	00_TSL_Grundschaltung.rml	SUCCESS	---test performed---	R_D_Wechsel_success_37.06s.mdf	1	0
Ankriechen_D	01_Ankriechen.rml	SUCCESS	---test performed---	Ankriechen_D_success_53.07s.mdf	1	0
Anfahren_D_wenig_Fahrpdl	01_Ankriechen.rml	SUCCESS	---test performed---	ren_D_wenig_Fahrpdl_success_64.8ɛ	2	0
Anfahren_D_Volllast	01_Ankriechen.rml	SUCCESS	---test performed---	ıfahren_D_Volllast_success_61.53s.m	1	0
Anfahren_R_wenig_Fahrpdl	01_Ankriechen.rml	FAILED	---test performed---	hren_R_wenig_Fahrpdl_failure_96.15ɛ	0	1
Anfahren_R_Volllast	01_Ankriechen.rml	INCONCLUSIVE	Engage R | Launch with full throttle			
Gr_1	02_Schaltungen.rml	SUCCESS	---test performed---	Gr_1_success_3.8s.mdf	6	0
Gr_2	02_Schaltungen.rml	SUCCESS	---test performed---	Gr_2_success_61.61s.mdf	3	0
Gr_3	02_Schaltungen.rml	SUCCESS	---test performed---	Gr_3_success_62.86s.mdf	1	0
Gr_4	02_Schaltungen.rml	SUCCESS	---test performed---	Gr_4_success_64.41s.mdf	1	0
Gr_5	02_Schaltungen.rml	SUCCESS	---test performed---	Gr_5_success_67.86s.mdf	1	0
Gr_6	02_Schaltungen.rml	INCONCLUSIVE	Get D6 while driving			
Gr_7	02_Schaltungen.rml	INCONCLUSIVE	Get D7 while driving			

FIGURE 4 Excel frontend for acceptance tests (© QTronic)

SUPPORT FOR VARIANT DEPENDENT REQUIREMENTS

The Daimler AG builds dozens of different variants for one car model [2]. The differing configurations sometimes expose sets of optional features or differing implementations of the same functions, with differing operational characteristics. Of course, this must be reflected in the set of requirements that are checked with a given system configuration. The requirements associated with a system must be composable and configurable in the same way in which the system itself is composable and configurable.

RML supports this with the declaration of system configuration constraints. This declaration is part of the meta information provided by the "@config section of an RML file." For instance, the watcher Change_driving_program from **FIGURE 2** will be considered only when testing a system that has an automatic transmission with type AT or DCT. This configuration mechanism can also be used for (de)activating platform dependent requirements, since some evaluations are only possible with special measurement hardware.

PRACTICAL EXPERIENCE

An already existing acceptance test catalogue with about 120 requirements was used as specification in the pilot phase of the first application project. It took a test engineer located off-site about two weeks to implement and test the associated RML watchers with the virtual ECU simulation implemented with Silver. Because the access to car prototypes is quite limited, the validation of the watchers in simulation was very helpful.

Before the introduction of this new testing solution, performing systematic tests in the car was very time consuming. Way too often tests were missing or having inconclusive results. Due to the high costs, the routine tests performed in the car focused mostly on new features or changed functions. Now, checking if old features still work properly, after new ones are introduced, is performed with virtually no added cost: the RML-setups collect the old and the new watchers, they are all concurrently checked during the entire drive.

Today, every new virtual ECU is also automatically tested against the basic acceptance test catalogue after new software was integrated on the server (nightly continuous integration). This takes about 10 m for a DCT transmission system and is done before the software is flashed on real control units. In case of failed tests, this prevents blocking expensive resources, such as cars and test rigs and lots of engineers from debugging failed setups. More intense automated tests are performed as well, either periodically or at certain quality gates. The Excel document and the measurement snippets serve as long term documentation, too. They can be archived with conventional repository tools and help to ensure statutory testing activities.

Future activities aim at improving the driver interaction for real-time validation, as well as the cooperation between departments. The latter is inevitable, since many functions are distributed amongst several ECUs, as for instance the start-stop operation.

CONCLUSION

Platform independent testing is a highly relevant task, essential for increasing the product safety and quality with affordable costs. Requirement models, that are formal, reusable, and at the same time, easy to develop and to test, proved to be essential in order to achieve a thorough system testing with sufficient coverage. This approach applies to simulation, to measurement evaluation and finally to the acceptance tests performed on hardware-driven test systems.

REFERENCES

[1] Brückmann, H.; Strenkert, J.; Keller, U.: Model-based Development of a Dual-Clutch Transmission using Rapid Prototyping and SiL, Intern. VDI Congress Transmissions in Vehicles, Germany, 2009

[2] Gloss, S.; Slezak, M.; Patzer, A.: Systematic Validation of over 200 Transmission Variants. In: ATZelektronik worldwide 8 (2013), No. 4, S. 34-39

[3] Rink, A.; Chrisofakis, E.; Tatar, M.: Automating Test of Control Software – Method for Automatic Test Generation. In: ATZelektronik worldwide 3/2009, No. 3, pp. 52-57