

Efficient Testing Framework for Simulink Models with MTCD and Automated Test Assessments in the Context of ISO 26262

Tobias Schmidt, Shan Jin, Jens Rogalli, Thorsten Rogier,
Hartmut Pohlheim, and Ingo Stürmer
Model Engineering Solutions

ABSTRACT

Requirements-based functional testing of model-based embedded software is a crucial requirement of the ISO 26262 safety standard for passenger cars [1]. Test assessment of requirements-based test cases is a laborious task and checking test results manually is prone to error. The intent of this paper is as follows:

We introduce a method for requirements-based testing, which allows testing and automatic evaluation of single as well as several (grouped) requirements with one test sequence. Within a large-scale industrial project we have already shown that our new approach reduces testing expenditures and susceptibility to errors. Within this paper we shall present a method which facilitates the fulfillment of requirements traceability stipulated by ISO 26262. This method supports automated test case generation from test specifications, which then can be executed and assessed by a test tool automatically. The combination of these two methods in an efficient testing framework results in a significant reduction of testing expenditures and considerable increase in test coverage.

CITATION: Schmidt, T., Jin, S., Rogalli, J., Rogier, T. et al., "Efficient Testing Framework for Simulink Models with MTCD and Automated Test Assessments in the Context of ISO 26262," *SAE Int. J. Passeng. Cars – Electron. Electr. Syst.* 7(1):2014, doi:10.4271/2014-01-0306.

INTRODUCTION

Increasingly complex software combined with high quality standards makes efficient testing imperative. The ISO 26262 safety standard for passenger cars makes the development of efficient technologies for testing crucial. ISO 26262 requirements regarding test coverage (see ISO 26262 - 6, 9.4.5., Table 12 and ISO 26262 - 6, 10.4.6, Table 15) in relation to requirements tracing and recommendations for requirements-based testing (see ISO 26262 - 6, 9.4.3, Table 10 and ISO 26262 - 6, 10.4.3, Table 13) make a change in current test methods necessary.

Until now, only test oriented approaches were applied. These imply that a number of test sequences are created for each requirement, which are then examined and evaluated separately. Consequently, a great number of test sequences were necessary to reach a sufficient level of test coverage. In the following, we shall go into requirements-based testing, where the relation between test sequences and requirements is depicted as $n \times m$, instead of $n \times 1$. This means that m requirements are tested by a test sequence, instead of a single requirement. As a result, test

coverage increases substantially. With the automated evaluation of test results, the automated test assessment considerably reduces test expenditures.

Requirements traceability, another essential stipulation of ISO 26262, may also lead to additional costs. This prerequisite includes that each safety-critical requirement must be traceable within the software, from the software architecture, the safety plan, and the test specifications, to the test implementation. Therefore, modifications that typically occur during the development process have to be entered into various documents and artifacts, thus increasing complexity. With respect to this, we are going to elaborate a method regarding MTCD (Model Test Case Definition), which allows the automated creation of test sequences from tabular forms of test specifications. These are then performed by a test tool automatically. As a result, the maintenance of only a single table is necessary. This table includes the test specifications with the IDs of covered requirements, which are then automatically carried over into the

test, from the test implementation to the test report. Thus the complexity of synchronizing modifications can be significantly reduced.

The combination of automated test assessment and MTCD provides a testing framework for Simulink models by which ISO 26262 requirements can be met with substantially less effort.

METHODS

At this stage, it is necessary to analyze test model simulation and test sequence generation with MTCD (Model Test Case Definition). Subsequently, we shall consider requirement-based test evaluation and go into details over the Assessment Framework in this regard.

MTCD - Model Test Case Definition Language

Different functions can be defined within the test process: the creation of test specifications based on requirement specifications, the implementation of test cases within the test environment, their subsequent validation, their application to software checks, and finally the management of test cases with regard to later modifications.

In the case of comprehensive software projects, these functions are usually assigned to different developers. To avoid implementation errors, all parties involved need to have an equal understanding of the content of the test cases. Misunderstandings lead to additional communication cycles between participants and erroneous test results.

Therefore, definitions of test processes and test goals in the test specifications are decisive. They should be as precise as possible in order to eliminate misinterpretation, to cover all test scenarios, and to enable test simulations directly from test specifications. At the same time, the language should be intuitive and easily interpretable to avoid long learning curves and raise acceptance among users. MTCD was created to accomplish all of the criteria above.

MTCD was developed as a test specification language to enable automatic test implementation/compilation. The test process is efficient when using MTCD for test specification, because test implementation is automatic. Moreover, error sources in the test process are reduced, and revision tracking is more effective.

In this section, we start with an overview of test languages available in the automotive industry. Then we provide a brief overview of the MTCD-structure and its main elements with some application examples.

The State of the Art

There are many languages developed for test specification and implementation. The Testing and Test control Notation (TTCN-3), is the only world-wide ETSI European Telecommunications Standard of Test specification and implementation in the telecommunication industry [2]. The TTCN-3 is a high-level technology-independent language for abstract test specifications. It can be used for the definition of generic data types, the definition of components (test management component, test logging component platform adapter, etc.) and communication ports that are used to build various testing configurations, and the specification of the dynamic test system behavior. It is verified that TTCN-3 has limits in its application to automotive software [3]. Based on the existing TTCN-3 standard, Leicher [4] et al. developed MyTTCN for the special application in continue embedded-System test. Grossmann et. al. [5] presented an XML-based language TestML, which was developed as a language for exchanging test description between different test platforms and test systems in the context of model-based testing of embedded automotive software. CTE XL [6], is a graphical editor for classification trees that can also be used for test case design. Depending on the application domain, the levels of formalism used to define the expressions and statements in a test language are different. However, the specification of a long and complex test sequence in any of the above mentioned test languages can become overwhelming, and the review of such test specification becomes complicated.

The simple and formal syntax of MTCD is easy to use and a well-documented MTCD test specification offers a very good overview of test sequences summarized into a test-group.

MTCD Syntax Overview

MTCD test specification is in table form. For example MS Excel can be used to draft MTCD test specification. A well-defined MTCD test case is shown in [Figure 1](#).

Like other well-defined test specifications, the definition of a test case in MTCD consists of a number of basic attributes: a distinct test ID, a name, the description of the test case or test group, the requirements covered by this test case, the initialization or precondition of the test case, the test case action and an additional type component to specify the entry type of the actual table row. The components of an MTCD test case are summarized in [Table 1](#).

The signal and parameter initialization defined in a 'test-group' will be applied to all test cases in this group. A well-defined 'test-group' should contain an identification number, distinct name, description and signal initialization.

TestID	Name	Type	Requirements	Test Case Description	Initialization	Action
TG_AP_001	Autopilot Basic Test	Test Group			altitude=0; alt_ctrl=1; alt_des=0; alt_rate=200; gs_cmd=0; gs_signal=0; gs_theta_err=0; gs_dist=0;	
TS_AP_001	Climb Mode	Test	AP_req_1	Motivation: ... Testsequence: ... Expected Result: ...	altitude=10000; alt_des=10000; alt_rate=200;	[+80.5s] alt_des=12000; [+3s] alt_rate=800; [+116s]
TS_AP_002	Hold Mode	Test	AP_req_2	Motivation: ... Testsequence: ... Expected Result: ...	altitude=10000; alt_des=10000; alt_rate=1000;	[+19.5s] alt_des=12000; altitude = ramp(12000, 150s); [+200s]

Figure 1. Example of a MTCD test case specification

Table 1. Description of the components of an MTCD entry-line

Component Name	Description
TestID	A unique identifier for a row in table
Name	Name of a test-group, test-case, or even a section defined in a row
Type	Attribute which identifies the kind of content to be defined in this row. Three types are allowed in MTCD-syntax: 'Test Group', 'Test' and 'Caption'/'Information'
Requirements	ID of requirements covered by test case defined. This information is important for enabling traceability applications.
Test Case Description	Textual description of test group and test sequence.
Initialization	Formal signal initialization (test-group, test-case) in MTCD-Syntax
Action	Signal sequence definition (test-case) in MTCD-Syntax.

In the case of defining a test case it is necessary to enter the Test ID, Name, Requirements, Description, Initialization, Action, and expected result.

The entries about Initialization and Action follow the syntax described below.

Initialization

An initial value will be assigned to a signal or parameter. Keyword 'p' identifies the name of a parameter. A formal Initialization of a test-group or test-case is defined as:

// comments

SignalName = { value | math.expression }; //comments

p ParamName = { value | vector | matrix | math.expr. };

A signal can be assigned with a value or a simple mathematical expression. The value of a parameter can be value, vector, matrix or a mathematical expression. Comments are marked with '//'. Examples of valid variants of Initialization are summarized below.

Signal1 = 5.8;

Signal2 = 10 + 17.3;

p Parameter1 = 100.5;

p Parameter2 = [5 22 6 23; 1 54 2 16];

Action

A test sequence is defined step-by-step in the 'Action'-cell. An action-step is defined as:

Signal1 = {value|math.expr. |function(arg1,arg2,...)};

Signal2 = {value|math.expr. |function(arg1,arg2,...)};

....

[+TimeIntervalLength]

This step describes the actions within a time interval and is a step within a test sequence. A test sequence can be constituted of any number of test steps. The following applies to all time intervals of a test step:

- All assignments are processed within the time interval
- Jumps occur at the start of an interval

- A time interval can contain any number of assignments (or no assignments)
- Assignments that belong to a time interval stand before the time interval
- Time designations are relative and are indicated in ms, s or min, for example +150ms. Default is s.
- Assignments are terminated by semi-colons

Element functions are prepared for describing complex signal sequences. For example, a linear ramp can be defined with MTCD element function *ramp*, as described in Figure 2.

`signal = ramp (target_value,duration,time_offset);`

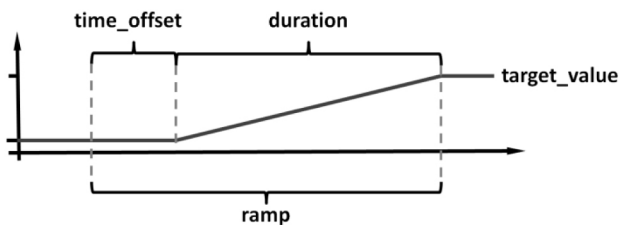


Figure 2. Illustration of parameters of the ramp function

The first parameter is the *target_value* that should change the signal. *Duration*, the optional second parameter, can be used to define the time length of the *ramp*. If the *duration* is not indicated, the length of the effective time interval is employed. The optional *time_offset* parameter can be used to delay the start of the *ramp*. The ramp begins with the effective time interval without *time_offset*. A minimal representation of the command *ramp* ensues:

`signal = ramp (target_value);`

A smooth transition between values can be specified formally with MTCD function *sinramp*, as described by Figure 3 and defined by:

`signal = sinramp(target_value,duration,time_offset);`

It is obligatory to provide a target value (*target_value*) in *sinramp*-transition specification. The assignment of *duration* and *time_offset* is optional.

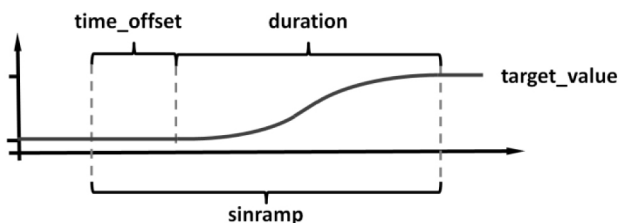


Figure 3. Illustration of parameters of the sinramp function

Available MTCD element functions include *sin*, *ramp*, *sinramp*, *gradient*, *set*, *impulse*. Further functions can be added, that can be optionally complex. It is also possible to combine functions.

A valid signal sequence can be defined as:

```
signal = ramp(10); // signal ramped to 10
[+100ms]
```

A noisy signal can be specified by combining a predefined MTCD-function with the noise function.

```
signal 1 = ramp(10) + noise_wg(1);
// noise_wg (white Gaussian noise)
[+100ms]
```

Key Features of MTCD-Syntax

A complex signal sequence can be easily, clearly and formally specified using MTCD-Syntax. Describing signal sequence with MTCD element-functions (*sin*, *ramp*, etc.) will ease the task of test specification further. In addition, it is possible to assign a signal with parameters defined in MTCD or in MATLAB-workspace. A noisy signal sequence can be defined in MTCD-Syntax as well.

Besides a clear definition of MTCD-Syntax, a parser was developed to verify and read test sequence specifications in MTCD format, and generate automated signal processes. This can be done in different output formats.

Multiple Signal Specification/Implementation with MTCD

A real maneuver for steering-control unit test is specified in MTCD-syntax below.

```
[+5s]
// ramping of gas pedal to 30% for 35s
acc_pedal = ramp(30,35s);

[+15s]
// start sinus of steer angle with amplitude 30
// period time 30s,duration 50s,phase off.0.5pi
// and mean offset of 0
st_angle = sin(30,10s,50s,0.5,0);

[+35s]
// slipping off the gas pedal,set acc_pedal to 0
acc_pedal = 0;

[+25s]
```

This MTCD specification includes following asynchronous actions:

1. After 5 seconds, the position of the accelerator-pedal will ramp to 30% within the next 35 seconds.
2. After 15 seconds (at the 20th second), the steering wheel begins to move. The change of the steering angle follows a sine waveform with an amplitude of 30°, a period of 10 seconds, and a phase-offset of 0.5 T. The change of steering angle takes 50 seconds.