TestML
An Exchange Notation for Model-Based Testing

Jürgen Großmann, Mirko Conrad, Ines Fey, Alexander Krupp
Klaus Lamberg, Christian Wewetzer et al.
Contents

- TestML Overview
  - background information and
  - motivation

- TestML Basics
  - XML schema,
  - data types and
  - stimulus behavior

- TestML Examples for
  - continuous stimuli
  - reactive test sequences
  - measured data

- TestML Current Status and Outlook
**TestML Background**

**Test Markup Language (TestML)**

- A XML-based test exchange language for model-based development.
- Exchange of test descriptions and results
- Tailored towards the needs of the automotive industry.
- Result of the German Ministry of Education and Research (BMBF) project IMMOS (Integrated Methodology for Model-based ECU Development)
- Project participants are *DaimlerChrysler, dSPACE, IT Power, Fraunhofer FIRST, FZI Karlsruhe and the University of Paderborn*
TestML Motivation

Different Sources for Tests

- Specified during different phases of development
- Made up from different sources/methods
  - manually developed test specifications
  - automatic test generations from models and codes
  - reused specification and/or legacy specifications
TestML Motivation

Heterogeneous Test Specifications

- Methods and notations are not integrated yet.
- Platforms are not integrated yet
- To support reuse and test exchange create a platform for test exchange
- TestML is a major building block of this platform
TestML Basics

Language Basics

- TestML is a test exchange language
- Independent from tools and methods.
- Exchange of test stimulation, recording, evaluation and results.
  - supports discrete and continuous (analogue) stimuli
  - provides a concept of time to describe time-dependent events
  - supports reactive tests
  - management of measured data as test input or as reference for comparative tests
- Provides a set of expressions for
  - primitive continuous signal definitions
  - management of control flow during stimulation
  - test evaluation regarding the analysis of discrete and continuous signals
SUT represents the system to be tested. Mainly relevant is the test interface.

The stimulation unit is responsible for the generation of test stimuli.

The capture unit records the system reactions and/or the system reactions together with the test stimuli.

The evaluation unit is responsible for the evaluation of test cases. It accesses all data recorded by the capture unit.
<testML/> forms the root element for each TestML description and represents a large amount of test cases.

<testSequence/> and <rtTestSequence/> constitute test sequences that describe an operating scenario to be tested.

- RTTestSequence has a distinct relation to time. We know about duration of operation and time distances between all events.
- TestSequence has no distinct relation to time. We know about order of event occurrence and the time distance between certain events (use of timer).
TestML Basics

Structural Elements (Inside Test Sequences)

- `<testInterface/>` serves to describe the interface to the SUT.
- `<stimulate/>`, `<capture/>` and `<evaluate/>` describe stimulation, recording, and test analysis within one test sequence.
- `<behavior/>` is used for the specification of stimulation behavior in a generic form.
TestML Basics

Data Types

- TestML types have the following meanings:
  - boolean, integer double are types to describe scalar values
  - signal is a complex type to represent a set of values over time
  - time is a complex type representing a duration or a fixed point into time

```xml
<signal ID="ID5" type="double">
  <time><unit>s</unit>
    <double><value>10</value></double>
  </time>
  <ramp>
    <start><double><value>0</value></double><start>
      <end><double><value>100</value></double></end>
    </ramp>
  </signal>
```
TestML Basics

Describing Stimulation Behavior with Automatons

- Specification of continuous signal parts through primitive signal expressions
- Specification of overall control flow through automatons
- For modularization purposes, automatons can be nested within other automatons
- Automaton semantics based on Mapping TestML to a widely accepted automaton notation (e.g. Stateflow statecharts)
TestML Examples

Cruise Control Example

- Tests the “Pedal Interpretation”, a cruise control subsystem.

- Test Interface:

<table>
<thead>
<tr>
<th>port</th>
<th>type</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>phi_Gas</td>
<td>double</td>
<td>angle of the gas pedal</td>
</tr>
<tr>
<td>phi_Brake</td>
<td>double</td>
<td>angle of the brake</td>
</tr>
<tr>
<td>v_target:</td>
<td>double</td>
<td>target speed</td>
</tr>
<tr>
<td>v_act:</td>
<td>double</td>
<td>vehicle speed</td>
</tr>
<tr>
<td>LeverPos:</td>
<td>integer</td>
<td>lever position</td>
</tr>
</tbody>
</table>
TestML Examples

**MTest: Continuous Behavior**

Tests the hysteresis for pedal interpretation

- Vehicle speed is kept constantly at -10 m/s. and after a second it is set to -5 m/s for another second.
- The accelerator pedal is raised from 0% to 100% and then lowered linearly from 100% to 0%.
- The brake pedal is linearly lowered from 100% to 0% and then raised from 0% to 100%.
**Tests activation of cruise control**

- Set target speed \( v_{\text{target}} \) at 18 m/s.
- Use gas pedal until vehicle speed is greater \( v_{\text{target}} + \text{tol}_1 \) (e.g. 25 m/s).
- Switch on cruise control.
- Wait until vehicle speed is less \( v_{\text{target}} + \text{tol}_2 \) (e.g. 19 m/s).
TestML Examples

Measured Data

For Measured Data

- We ensure that exactly one signal is defined inside each step
- Each step has the same duration $\Delta t$ (here 0.2 s)
  - For time control, a timer is started in every state.
  - If the timer reaches the defined limit the next state is entered
TestML Examples

Exchange between CTE and Automation Desk

CTE Test Specification

Automation Desk Specification

#*** begin of signaldescription ***
TT(0)  # time tag
phi_acc.RAMP(0, 100, 100)
phi_brake.RAMP(100, -100, 0)
v_acc.CONST(-10)

TT(1)  # time tag
phi_acc.RAMP(100, -100, 0)
phi_brake.RAMP(0, 100, 100)
v_acc.CONST(-5)

TT(2)  # time tag
STOP()

#*** end of signaldescription ***

Step 1
- write(phi_Brake,ramp(100,0,1s))
- write(phi_Gas,ramp(0,100,1s))
- write(v_act,const(-10,1s))

Step 2
- write(phi_Brake,ramp(0,100,1s))
- write(phi_Gas,ramp(100,0,1s))
- write(v_act,const(-5,1s))
TestML Status and Outlook

Status and Outlook

- TestML-Schema Version 1.0.2 available
  - Supports exchange of stimulus and evaluation expression

- Next Steps
  - Refining semantics by mapping TestML to a widely accepted automaton notation (e.g. Stateflow statecharts)
  - Define conformance levels to support different kinds of language compatibility
  - Enhance TestML to support:
    - more tools and platforms by writing TestML adapters
    - more languages and concepts (e.g. temporal logic expressions)
    - method-specific and platform-specific annotations.
Thank You

Jürgen Großmann
DaimlerChrysler AG
Method and Tools (REI/SM)
Email: Juergen.Grossmann@DaimlerChrysler.com

IMMOS Project
Integrated Methodology for Model-based ECU Development
http://www.immos-project.de