The MATE Approach: Enhanced Simulink® and Stateflow® Model Transformation

Ingo Stürmer (Model Engineering Solutions)
Ingo Kreuz (DaimlerChrysler AG)
Wilhelm Schäfer (University of Paderborn)
Andy Schürr (TU Darmstadt)
MATE Project Partners

DAIMLERCHRYSLER
Research & Technology
(Ingo Kreuz)

Real-Time Systems Lab
(Andy Schürr)

Software Engineering Group
(Wilhelm Schäfer)
MATE Objectives

Functionality:

- Tool support for model-based development on the basis of MATLAB®, Simulink®, and Stateflow® models
- Developer support in verifying and observing even the most complex modeling guidelines
- Automated (partially interactive) model transformation with regard to conformity with modeling guidelines
- Advanced editing functions (beautifier, pattern generator, etc.)

User interface(s):

- Deep integration in the MATLAB, Simulink, and Stateflow environment (Model Advisor)
- Web server solution (add-on for e-Guidelines Server)
Presentation Outline

1. Motivation

2. Modeling guidelines in practice

3. Advanced model (dataflow) analysis

4. Automatic/interactive model repair (model refactoring)

5. Enhanced editor operations

6. Conclusions and future work
Motivation for Model-Based Development

- Seamless use of executable models in all software development stages
  - Specification → design → implementation → testing

- (Some) significant advantages:
  - Early testing: begins on model level even before implementation is available!!!
  - Model-based code generation: productivity enhancement through the use of model-based code generators (automatic translation of controller models into highly-efficient production code)
The increasing complexity of electronic systems and associated software in vehicles is resulting in increasingly complex models.

Modeling tools offer only limited mechanisms to master this complexity and support developers.

This is particularly evident from:

- Hitherto insufficient automated support in checking and correcting models with regard to modeling guidelines.
Modeling Guidelines at DaimlerChrysler

- At present: approx. 200 rules and patterns
  - Available as text documents or so-called “e-Guidelines” on commercial web server (e-Guidelines Server)
  - Guidelines available online or stand-alone
  - Comprehensive search and filter options are provided

1. General modeling conventions
   - Naming conventions
   - Model structuring
2. Modeling patterns and rules
   - Simulink
   - Stateflow
   - Mixed Simulink and Stateflow
3. Autocode intent guidelines
   - Tool-independent rules
   - TargetLink
   - Real-Time Workshop®
4. Model-based testing guidelines
5. …

www.e-guidelines.de
Experiences of Using Tools for the Statistical Analysis of MATLAB®, Simulink®, and Stateflow® Models
Case Study: Static Model Analysis with MINT

- Checking a complex MATLAB, Simulink, Stateflow model
Potentials for Model Refactoring Using MATE

- Immediate repair: 900; 45%
- Interactive repair: 870; 43%
- Manual change necessary: 90; 4%
- Unsettled: 170; 8%

More than 80% bug fixes with semi-automatic model refactoring operations
Conclusion

- Modeling guidelines are an important and appropriate means of guaranteeing model quality and the resulting code.

- The high number of rules to be controlled makes manual checking (of compliance with guidelines) time-intensive and error-prone.

- Even the use of a static analysis tool will not make the modeler’s task in performing revisions to the model any easier.

- Model transformation means a considerable work burden is taken away from the modeler (approx. 70% time-saving).

- High-level checks with graph transformation tools allow additional model analyses, which were difficult and complicated to realize with previously implemented means (M-Scripts).
Advanced Model Analysis: Dataflow Analysis

- Analysis of fixed-point scaling information
- Calculation of maximum quantization (scaling) errors
- Minimal and maximal range calculation
- Suggestions for improvements (in future version)

\[ X = A \times ((C-B)-(A-B)) \]

<table>
<thead>
<tr>
<th>Data type:</th>
<th>Scaling:</th>
</tr>
</thead>
<tbody>
<tr>
<td>sfixed(16)</td>
<td>$2^{-11}$</td>
</tr>
<tr>
<td>sfixed(16)</td>
<td>$2^{-9}$</td>
</tr>
<tr>
<td>sfixed(16)</td>
<td>$2^{-11}$</td>
</tr>
<tr>
<td>sfixed(32)</td>
<td>$2^{-16}$</td>
</tr>
<tr>
<td>sfixed(32)</td>
<td>$2^{-16}$</td>
</tr>
<tr>
<td>sfixed(16)</td>
<td>$2^{-11}$</td>
</tr>
<tr>
<td>sfixed(16)</td>
<td>$2^{-11}$</td>
</tr>
<tr>
<td>sfixed(16)</td>
<td>$2^{-11}$</td>
</tr>
<tr>
<td>sfixed(16)</td>
<td>$2^{-11}$</td>
</tr>
<tr>
<td>sfixed(16)</td>
<td>$2^{-11}$</td>
</tr>
</tbody>
</table>

**Dataflow 1**
- Output Error max: 2.44140625E-4
- Output min: -5.498948899375
- Output max: -5.499437180625

**Dataflow 2**
- Output Error max: 0.0011238956451416016
- Output min: 2.049329624296875
- Output max: 2.0502628157031255

**MATE X**
- Output Error max: 2.44140625E-4
- Output min: -5.498948899375
- Output max: -5.499437180625

**MATE Y**
- Output Error max: 0.0011238956451416016
- Output min: 2.049329624296875
- Output max: 2.0502628157031255
Interactive Model Analyses, Repairs and More

Analysis
(no. of product block inputs < 3)

Interactive repair
Limitations with Regard to Operand Numbers for the Product Block (tl_0009)

Description
If a fixed-point data type is specified for the product block, the number of input signals (or element numeric type) is specified for the output of the Product block.

Remark
The generation of proper fixed-point code requires that the number of operands and intermediate results be less than or equal to two. TargetLink therefore imposes a limit on the number of variables to two for fixed-point code generation. TargetLink allows for the fixed-point code generation limit on the number of variables to be increased using the TargetLink Interface Parameters.

Guideline Violation:
Found a product block with more than 2 input values.

Highlight the product block

Applicable Transformations: Cascade

Integration in MathWorks Model Advisor

visit www.model-engineers.com for live demo
An agreement to follow certain modeling guidelines is important to:

- Increase the comprehensibility (readability) of the model
- Facilitate maintenance, ease testing, reuse and extensibility,
- Simplify the exchange of models among OEMs and suppliers.

MATE supports instantiation of agreed-to modeling patterns.

visit www.model-engineers.com for live demo
Use of MATE for model analysis and repairs promises a significant increase in productivity when revising models in respect to their conformity with guidelines.

Currently MATE is the only available tool for MATLAB, Simulink, and Stateflow models, that in addition to model analyses, also has the following functionalities:

- Automated model repairs through graph replacement
- High-level editing functions (beautifier, pattern generator)
- Extended model analysis on graph level (e.g., dataflow analysis)

Two approaches (use cases) are currently being pursued with MATE:

- Direct call-up from MATLAB, Simulink, and Stateflow environment (Model Advisor)
- Batch operation on external server (e-Guidelines Server)
MATE Features NOT Shown during this Presentation

- Automatic generation of model check from so-called “ANTI Patterns”
- Calculation of model metrics
- Offline (Batch) transformations
- Report generation
Outlook

- Provision of analysis rules to check MAAB modeling guidelines

- Extension to TargetLink blockset

- Add-on for e-Guidelines Server
  - No local installation on PC necessary
  - Rule-checking guidelines are maintained centrally

- Provision of individual functionalities as a toolbox
  - Beautifier operations

- Extended search for industry partners
  - Implementation in (complex) real-life development projects
  - Extension of rules set
Contact

Model Engineering Solutions
Friedrichstraße 50
10117 Berlin
Tel.: +49 (0)30 20 65 9 173
Fax: +49 (0)30 20 65 9 200
stuermann@model-engineers.com
www.model-engineers.com

Dr.-Ing. Ingo Stürmer
Managing Director

http://www.model-engineers.com