Using Behavior Models for the Specification of Software based automotive Systems – Challenges and practical Experiences

Dr. Henning Kleinwechter, Carmeq GmbH
Dr. Andreas Leicher, Carmeq GmbH
Behavior Models for Software Specification

Agenda

› Challenges

› Model supported Specification

› Variant Management
Development of Electronic Control Units (ECU)

Challenges

- Increased requirement complexity in less time to market
  - Reduce failure rate and correction cycles during development
  - Provide precise requirement specification to ECU supplier
  - Description of either complex state based logic or control strategies, depending on the application

- Management of increasing functional variability
  - Product line / platform strategy
  - Hundreds of variation points in single ECUs software
  - Explicit variant modelling by using feature models
Behavior Models for Software Specification

Agenda

› Challenges

› **Model supported Specification**

› Variant Management
ECU Specification
Use of Models

Requirements Specifications have to be precise and consistent

- In order to get the correct functionality fast
- In order to get the same functionality from different suppliers
  (especially true for multi supplier strategy)

formalized text, informal models (more precise)

understandable? correct?

textual requirements (high degree of interpretation)
Use of Behavior Models
Model supported Specification

› Motivation
  › Early usage of models with defined semantic
  › Proof of correctness by simulation
  › Mainly Simulink/Stateflow used

formalized text, informal models

simulation model
defined semantic
proof of correctness through simulation

optional GUI
for demonstration purpose
Use of Behavior Models
Process View and Benefit

Early validation of functional requirements and detection of specification errors

OEM
Supplier

Specification
Default Implementation

Behavior Model

Further usage (optional)

Model
Simulation

Demonstrator

Test Oracle

Implementation Model

Specification

Implementation

Code generation
Model supported Specification
Role of models in development process

Informative Executable Specification
Informative part in addition to mandatory requirements specification

Mandatory Executable Specification
Reference for the implementation

Role of model in development process

Formal Specification
Models as part of the mandatory requirements specification
Role of Models
Informative Executable Specification

OEM

Requirements Engineer

Requirements Specification

Issues

Model Engineer

Tester

Behavior Model

GUI Simulation

mandatory

informative

Supplier

Requirements Specification

Requirements Engineer

GUI Simulation

Behavior Model
The improvement in the requirements specification (and the costs saved by this improvements) are usually compensated by the effort spend for modelling activities.

This strategy is profitable only, when models are further used for another purpose, e.g. rapid prototyping or series software implementation.
Role of Models
Formal Specification

OEM
Requirements + Model Engineer

Requirements Specification

mandatory

Issues

Model Engineer

Tester

optional

Behavior Model

GUI Simulation

optional

Supplier

Requirements Specification

GUI Simulation

Behavior Models for the Specification of Software based automotive Systems, V1.0
Requirements specification

Usually, only a part of the specification is replaced by models

Optional: GUI based Simulation
## Formal Specification
### Experiences

<table>
<thead>
<tr>
<th>advantage</th>
<th>disadvantage</th>
</tr>
</thead>
<tbody>
<tr>
<td>› Improvement in the requirements specification</td>
<td>› Still initially high effort, but this is scalable, depending on the degree of modelling</td>
</tr>
<tr>
<td>› Better understanding about functionalities on OEM and supplier side</td>
<td></td>
</tr>
<tr>
<td>› Low degree of interpretation</td>
<td></td>
</tr>
<tr>
<td>› Reduction in development time, since less requirements have to be</td>
<td></td>
</tr>
<tr>
<td>clarified</td>
<td></td>
</tr>
<tr>
<td>› Reduction of costs for CRs</td>
<td></td>
</tr>
</tbody>
</table>

- Improvement in the requirements specification with scalable effort
- Because only relevant parts of the specification are modeled, effort and cost for modelling activities are lower
Model supported Specification

Conclusions

- Model supported specification promises a variety of advantages
  - Less misinterpretation of requirements
  - Faster development cycles
  - Less communication effort
  - Better understanding of the underlying functionalities

- In order to obtain the most benefit
  - Models should play a mandatory role
  - Models should be exchanged with suppliers

- Initial high effort is notably justified in case of further usage of models
Behavior Models for Software Specification

Agenda

› Challenges

› Model supported Specification

› Variant Management
Variant Management
Motivation and Objectives

 Motivation

 Software functions have to be deployed for different platforms and vehicles
 Requirements differ slightly for each platform
 Set of relevant functions differ for each platform
 Software has to cope with these differences

 Objectives

 Explicit variability management for models and requirements
 Maintaining consistency between requirements and model variants
 Provide a single software model which is able to handle defined variants
 Management of similar software products (software product lines)
Variant Management

Feature Models

- Feature models explicitly describe variants in the software (Kang et.al., 1990)
- To reduce combinatorial multiplicity, relationships between features have to be defined
Overview Variant handling
Model Supported Specification

Variant Management

Configuration

DOORS Specification

Simulink Model

<<corresponds>>
Variant Management
DOORS Specification

Variants are maintained in feature models and are annotated to DOORS requirements. Feature selections are created by filters generated by the variability tool.
Variant Management
Simulink Model

Variant features are modeled in different subsystems. Correspondence to feature model is realized by special parameters. For each variation point in the feature model a separate parameter exists.
Variant Binding Time

Simulink Model

› Design Time
  › Variants are selected for a specific project, e.g. they are fixed for given project

› Compile Time
  › Variants are determined at compile time
  › Deselected variants are removed from production code via specific storage classes

› Run Time
  › Variants are expressed via parameters that can be changed during run time.

Compile Time and Run Time
Variants are modelled as „Enabled“ Subsystems
Variant Binding Time
Simulink Design Time Variants

- Are modelled as model references
- Variants are selected by variant objects
- Different implementations can be selected for specific projects.
Variant Management

Conclusions

› We use a variant model that coordinates requirements in DOORS as well as the Simulink model
  › Variability is explicitly defined for all stakeholders
  › Dependencies between variation points are defined
  › Less errors in ECU software configuration

› We handle all variants in a single Simulink model hierarchy, which includes constructs for all kinds of binding time variants

Further work

› Improvements in tool integration needed: feature modelling to requirements management and modelling tools
› Links between variant models of ECU functions and vehicle variant model model