Towards Verification-based Development of In-Vehicle Safety Critical Software: A Case Study
– Workshop CARS –

Martijn van den Heuvel - Reinder J. Bril - Peter van de Velde - Johan J. Lukkien

Systems Architecture and Networking (SAN)
Department of Mathematics and Computer Science
Eindhoven University of Technology
the Netherlands

27 April 2010
Outline

1. Introduction
2. Case Study
3. Tool Integration
4. Conclusions
Context and Background

Overall goals:
Show that the integration of Flexray-based by-wire technologies is functional, safe, robust and cost-effective.

Problem Description

- Model Based Control Design:

- Correct system behaviour:

- Automated platform mapping:

- Requires additional run-time libraries
**Problem Description**

- Model Based Control Design:
  - Automated platform mapping:
    - Requires additional run-time libraries

- Correct system behaviour:
  - How to integrate the ASD tooling with Matlab-Simulink while maximizing the benefits from both?

---

Martijn van den Heuvel (TU/e, SAN) Towards Verification-based Software 27 April 2010
A Case Study

Functionality:
- Control motor speed proportional to user input
- Detect sensor failure
- Stop on sensor failure
Design Steps:

1. Interface Specifications
2. Design Models
Design Steps:

1. Interface Specifications
2. Design Models

- All specified interface behaviour must be implemented by the system design
- Verified by Modelchecker
Model Integration Approaches

Legend: --- (Currently a) Manual Step -- Automated Step

1. Code Integration
2. Transform Design Models to Stateflow Models
   - Similar approach as:
     
     Sequence-based specification of feedback control systems in simulink®.
     
Embed ASD code in custom Simulink block

Wrapper passes Simulink I/O values to/from the ASD component
Conclusions

Given:
1. the model-driven development flow of Matlab Simulink
2. ASD’s specification tooling giving formal guarantees

How to integrate both development flows?
Conclusions

Given:
1. the model-driven development flow of Matlab Simulink
2. ASD’s specification tooling giving formal guarantees

How to integrate both development flows?

- Two proposals:
  1. Model transformation
  2. Code Integration

- Both alternatives need automatization
Conclusions

Given:

1. the model-driven development flow of Matlab Simulink
2. ASD’s specification tooling giving formal guarantees

How to integrate both development flows?

- Two proposals:
  1. Model transformation
  2. Code Integration

- Both alternatives need automatization
- Risk to add complexity to wrapper-code instead of model refinement
- Matlab’s code generator no longer preserves operational semantics between
  1. specification
  2. verification
  3. executable code