Safety Analysis of Automotive Software: From Functional Safety to Component Testing

Wojciech Mostowski and Mohammad Mousavi
Centre for Research on Embedded Systems
Halmstad University
Elevator Pitch

• Current practice of safety analysis (a la ISO26262) lacks support for systematic (de)composition

• Combination of techniques for model-based testing, learning, and model-based component mocking can provide such support mechanisms
Nomenclature (simplified)

• Safety: Absence of risk

• Risk: Combination of probability, severity, and controllability

• Controllability: Avoidance of injury or damage
ISO 26262

- Item Definition
- Hazards and Risks
- Functional Safety Concept
- Technical Safety: Goal and Case
- System Development
- Component Development
- Component Development
- Component Development

Refinement / Verification
Main Assumption

Models for system-level technical safety requirements

Model-Based Testing

Model-Based Testing of *Autosar basic software* is the main scope of the AUTO-CAAS project
Goal

• Decompositional testing: decomposing system-level technical safety requirements into tests on element / component

• Compositional safety validation: composing safety case from the test results
Challenges

• Decomposing the (technical) safety requirements:
  – decompositional model-based testing

• Coming up with models of components / elements / items:
  – automata learning

• Compositional safety validation:
  – mocked components, fault injection
Model Based Testing Ecosystem

Test-Case Generator
(UPPAAL Yggdrasil, Sikuli, SpecExplorer, TorXakis, QuickCheck)

Test Result:
pass or fail (+ counterexample)

Traceability Info.
Coverage Metrics
Impl. Under Test
Test DB

Model
Goal

- Decompositional testing: decomposing system-level technical safety requirements into tests on element / component

- Compositional safety validation: composing safety case from the test results
 Decompositional Model-Based Testing
Decompositional Model-Based Testing

for all c, 
(c || e) conforms S iff c conforms S(e)

Decomposability in Input Output Conformance Testing. MBT 2013.
Decompositional Model-Based Testing

Challenge: How to find a model $S(e)$ for $e$?
Automata Learning
Decompositional Model-Based Testing

Diagram:
- c
- Mocked Environment
- MBT Tool
- S(e)
  - a
  - b
  - c
Decompositional Model-Based Testing

QuickCheck

Model-Based Mock: Intercept, log, and forward

Simulated Environment (Environment Components)
Decompositional Model-Based Testing

Benjamin Vedder,
Testing Safety-Critical Systems Using Fault Injection and Property-Based Testing,
Conclusions

• Compositional trajectory for safety validation:
  – starting from system-level requirements
  – learning environments models
  – decomposing the requirements into component requirements
  – using mock models to intercept and forward calls and inject faults
Thank You Very Much!

Wojciech Mostowski
wojciech.mostowskii@hh.se