HALMSTAD UNIVERSITY

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

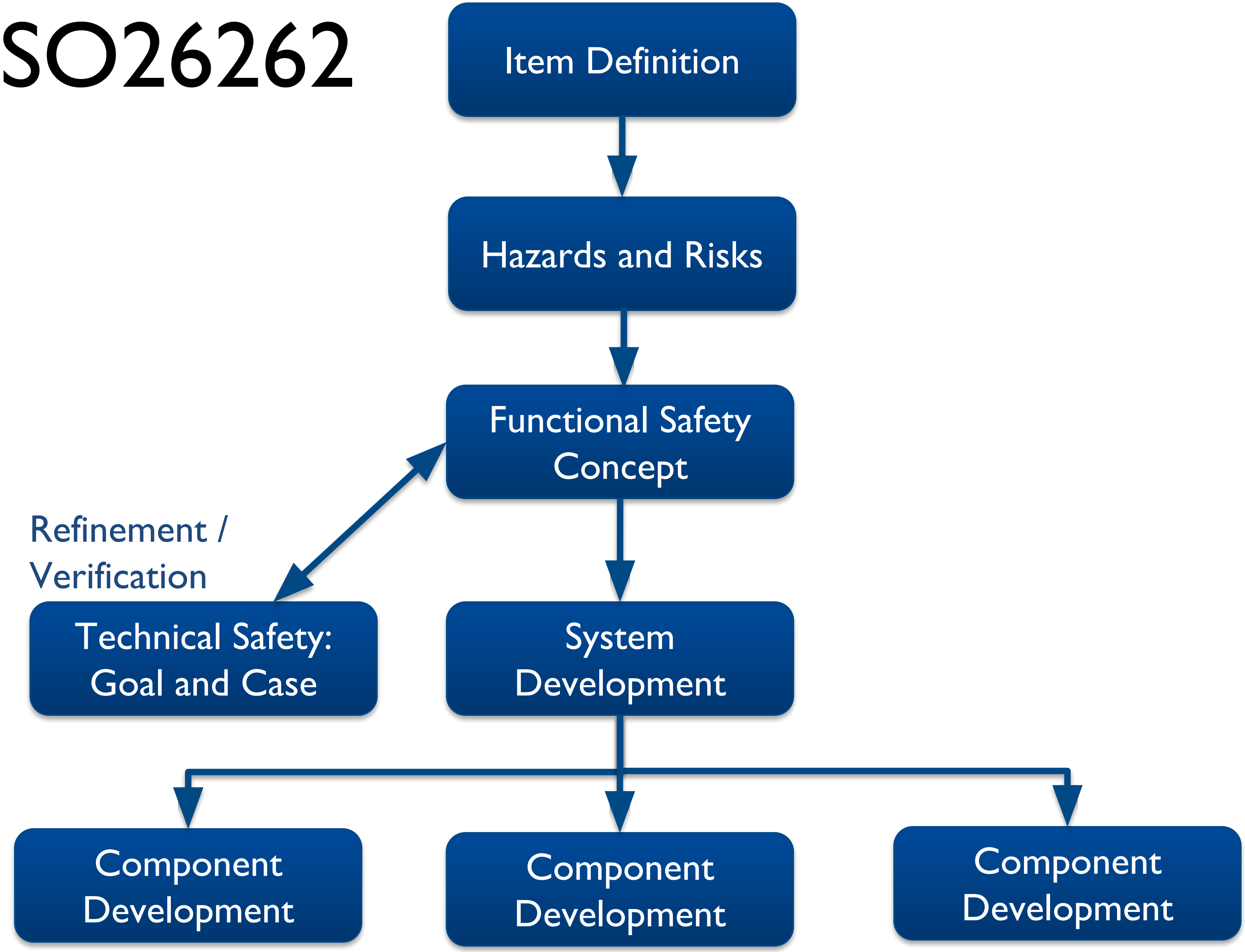
- Current practice of safety analysis (a la ISO26262) lacks support for systematic (de)composition

- 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

ISO26262



Item Definition

Hazards and Risks

Functional Safety Concept

Refinement / Verification

Technical Safety: Goal and Case

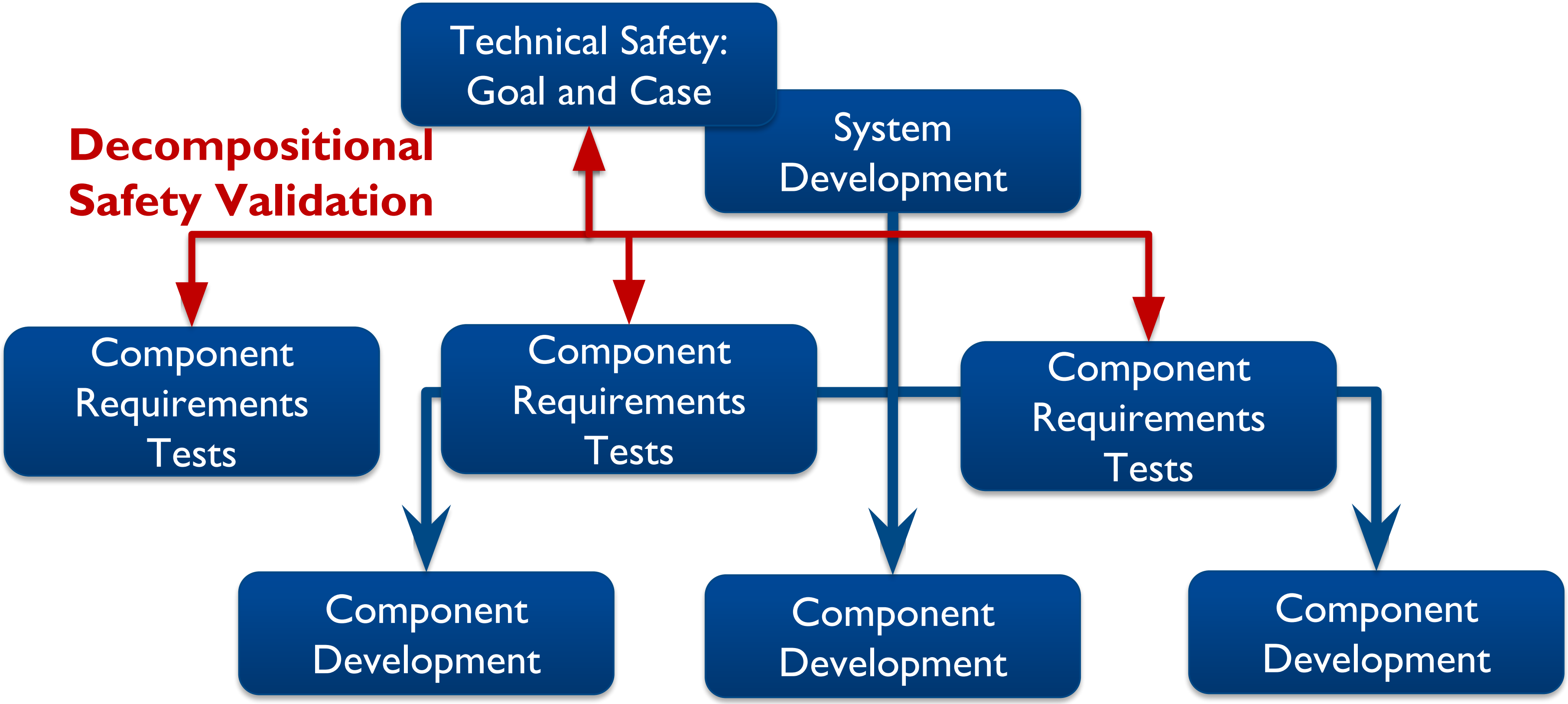
System Development

Component Development

Component Development

Component Development

ISO26262



Technical Safety: Goal and Case

System Development

Decompositional Safety Validation

Component Requirements Tests

Component Requirements Tests

Component Requirements Tests

Component Development

Component Development

Component Development

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

QuviQ ARC CORE

Goal

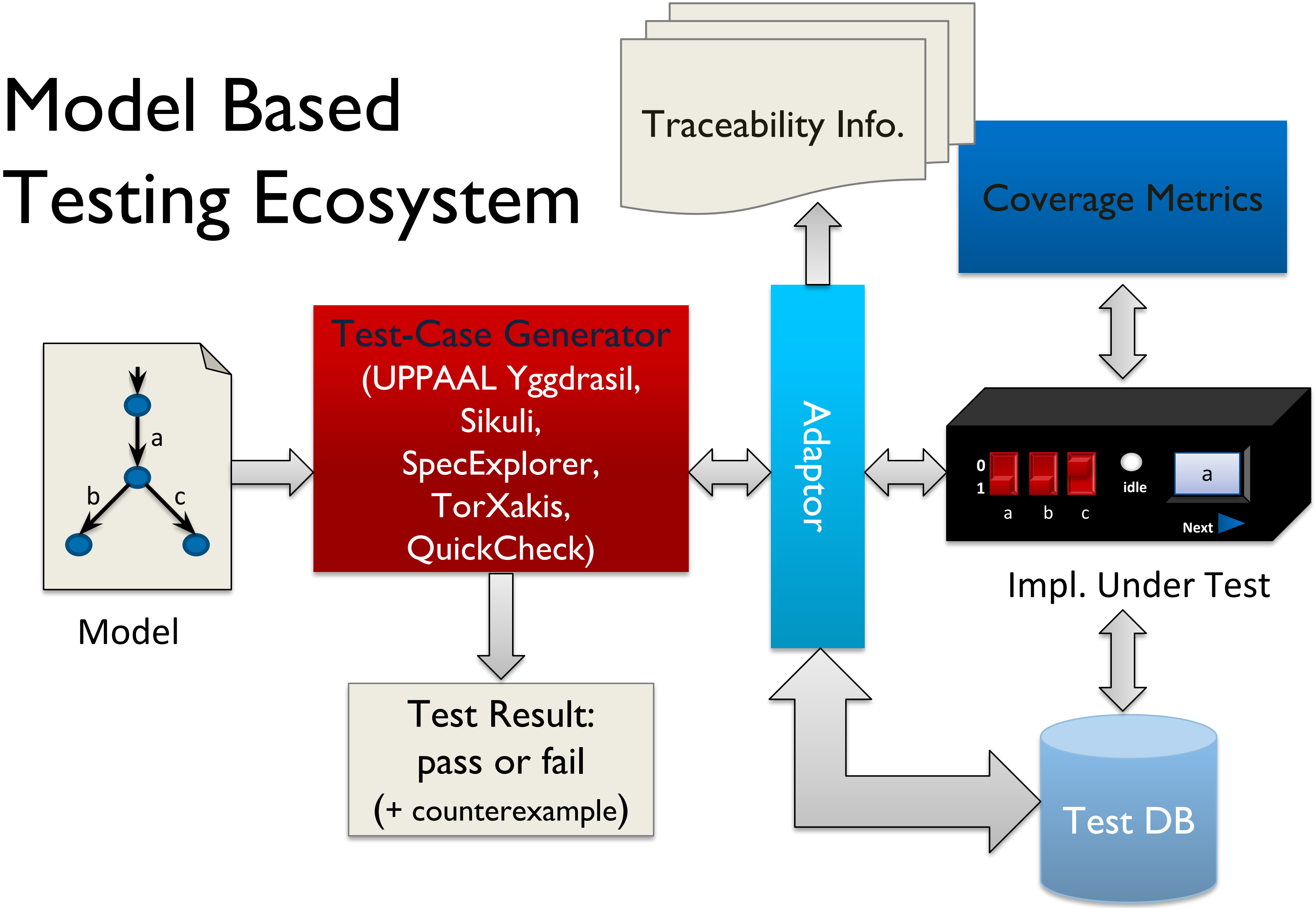
- Decompositional testing: decomposing system-level technical safety requirements into tests on element / component

- 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



Traceability Info.

Coverage Metrics

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

Adaptor

Model

Impl. Under Test

Test Result: pass or fail (+ counterexample)

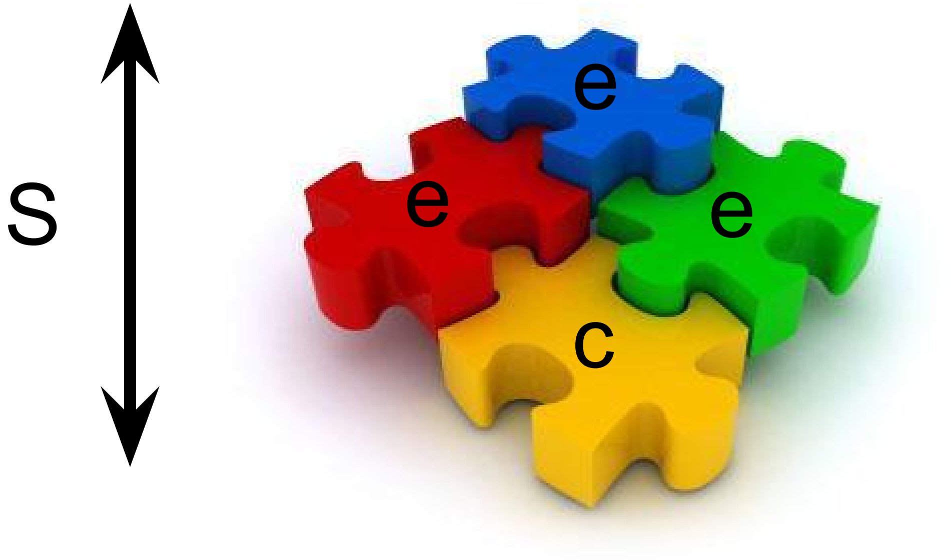
Test DB

Goal

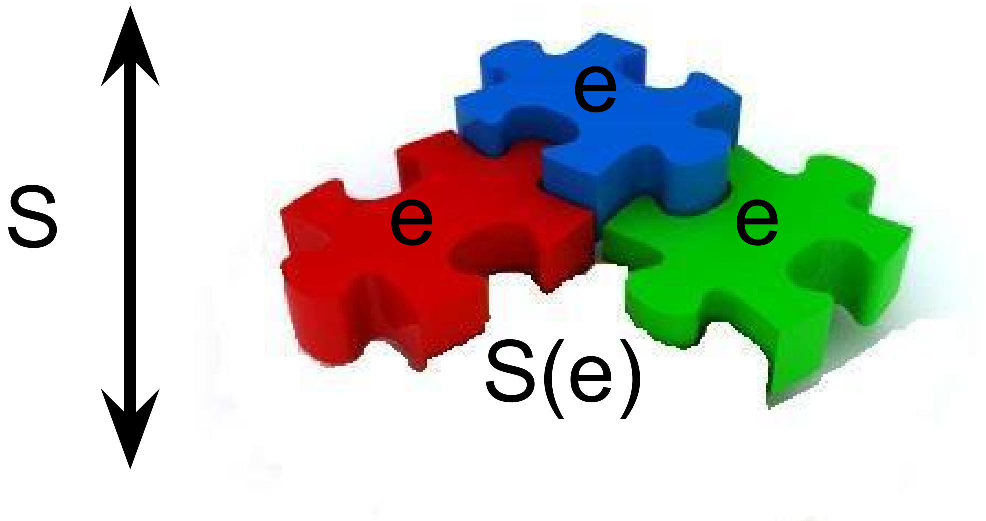
- Decompositional testing: decomposing system-level technical safety requirements into tests on element / component

- Compositional safety validation: composing safety case from the test results

Decompositional Model-Based Testing



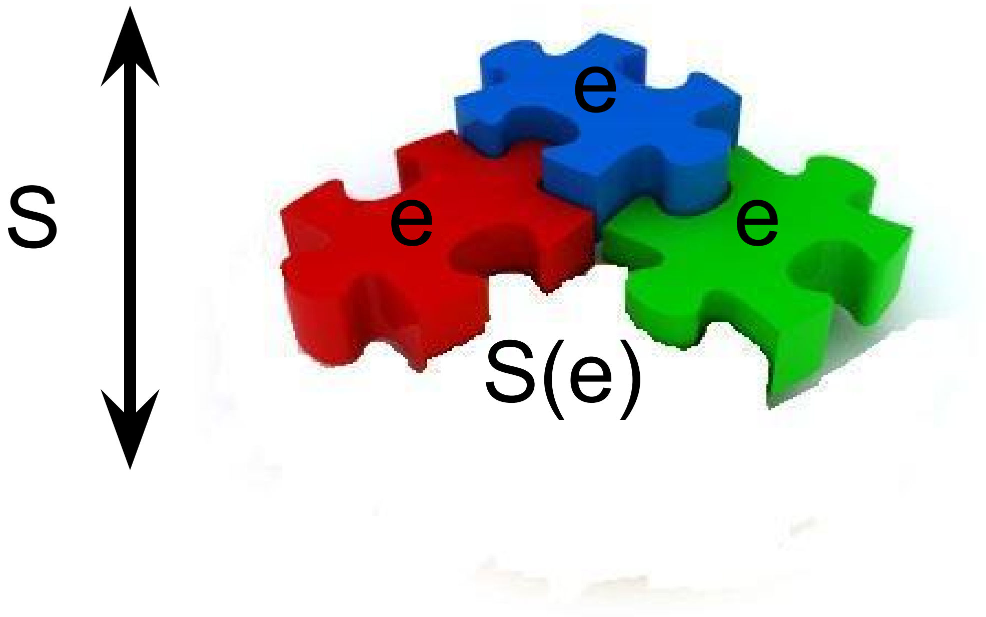
Decompositional Model-Based Testing



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

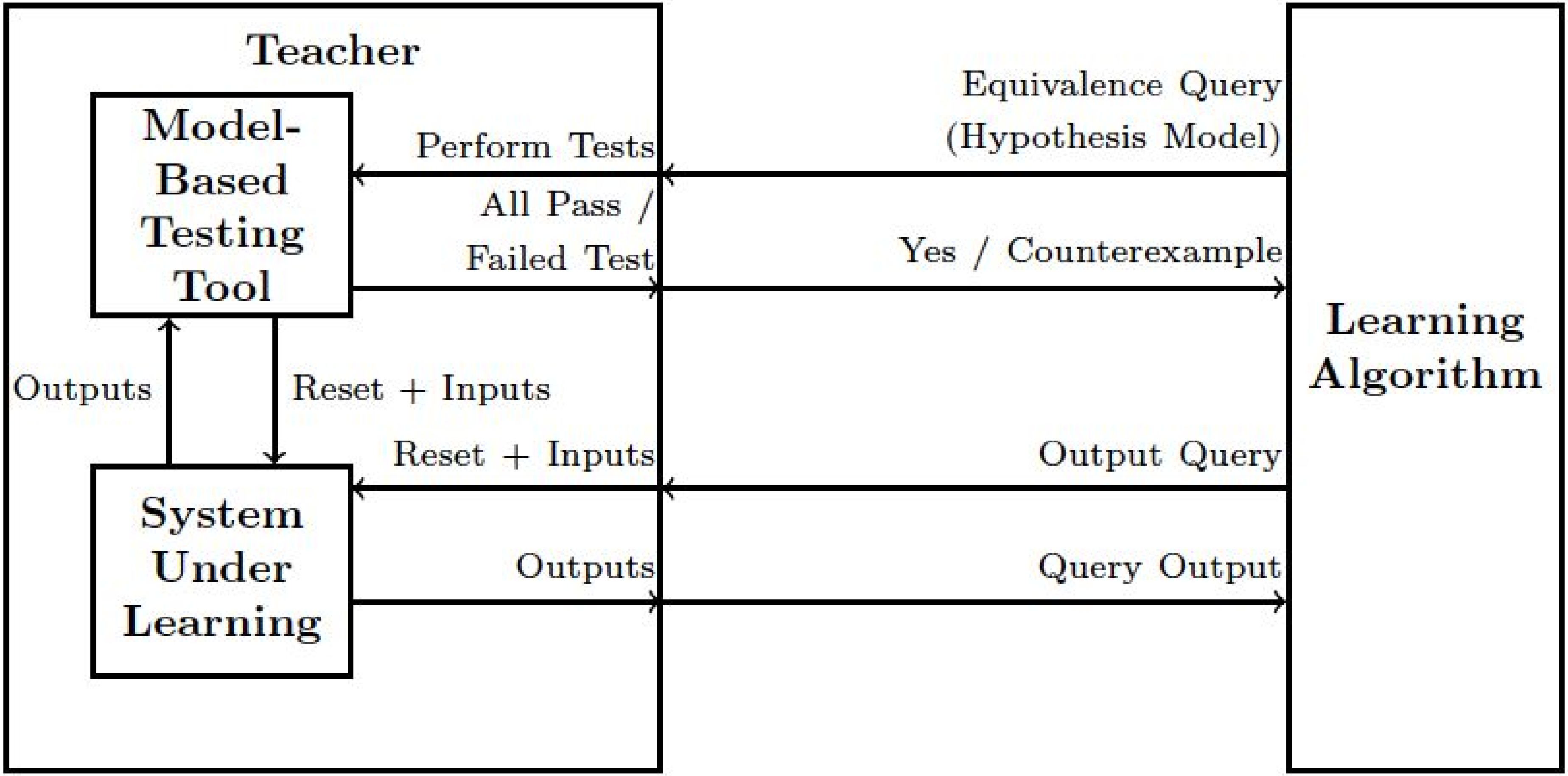
N. Noroozi, M.R. Mousavi, and T.A.C. Willemse.
Decomposability in Input Output Conformance Testing. MBT 2013.

Decompositional Model-Based Testing

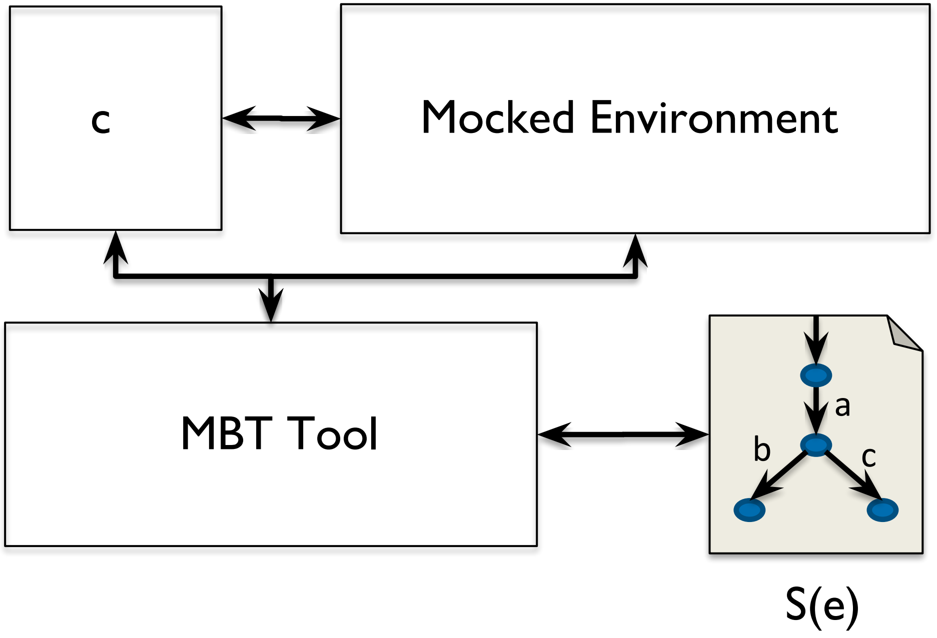


Challenge: How to find a model S(e) for e?

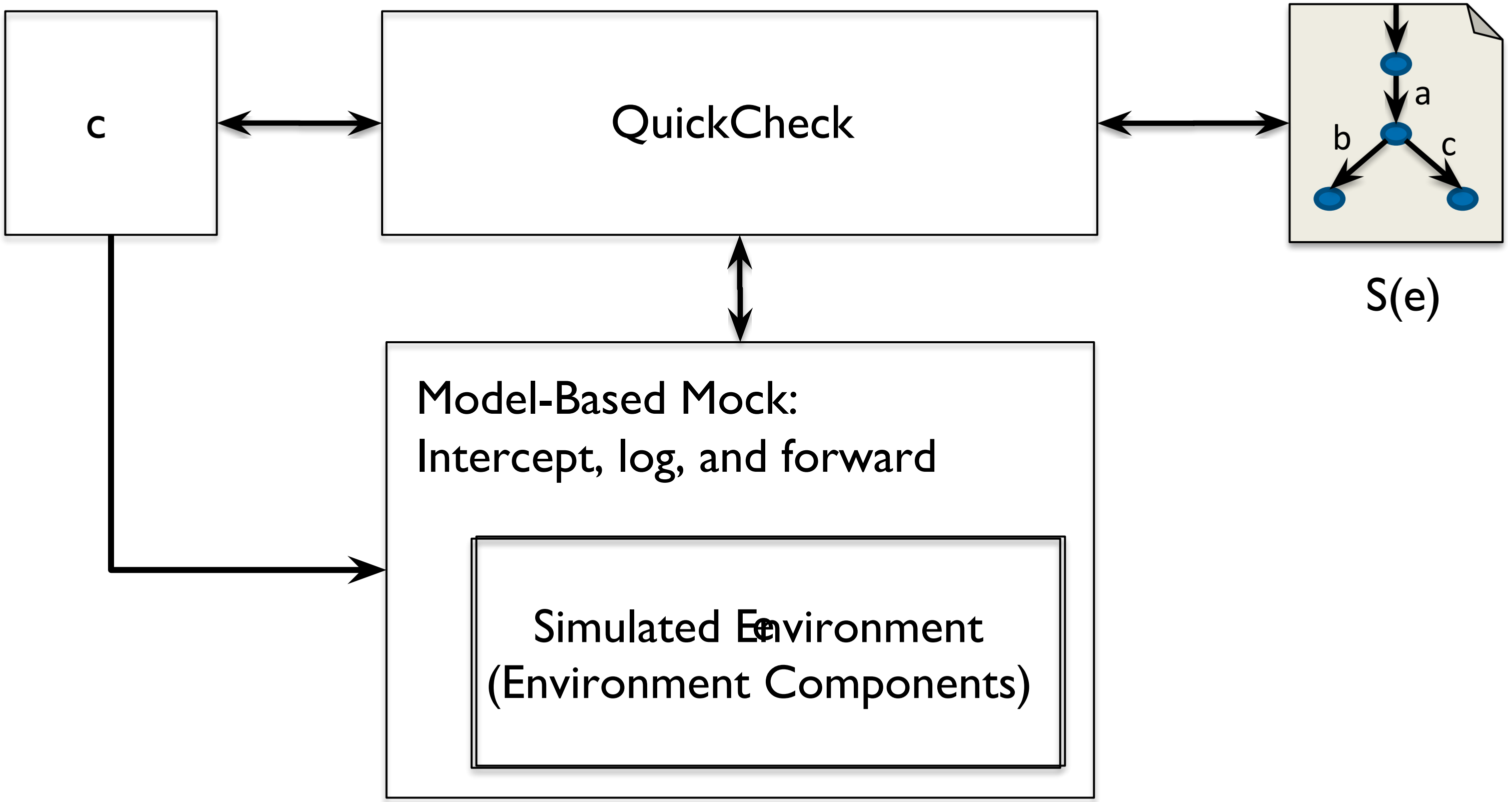
Automata Learning



Decompositional Model-Based Testing



Decompositional Model-Based Testing



c

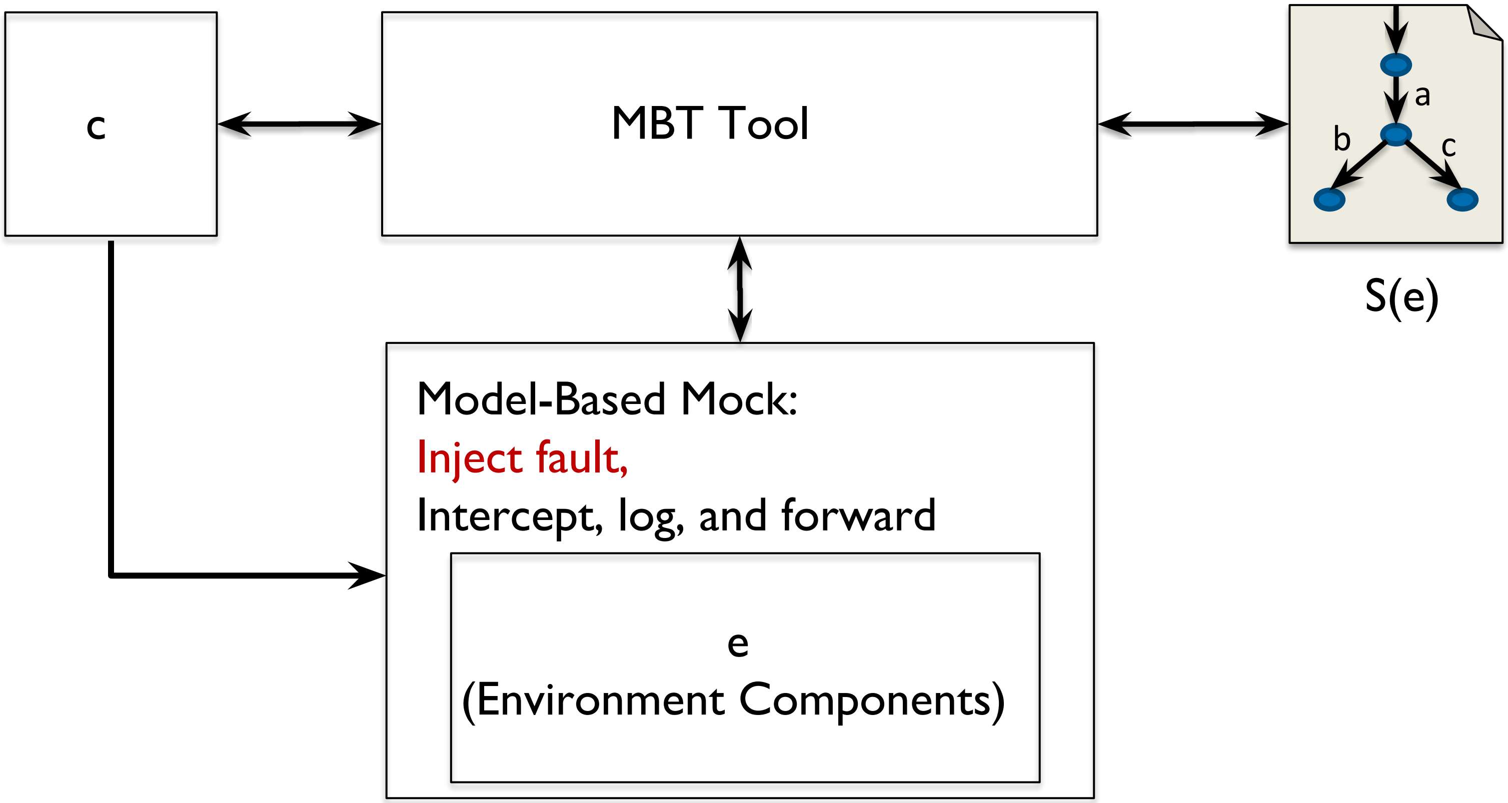
QuickCheck

S(e)

Model-Based Mock:
Intercept, log, and forward

Simulated Environment
(Environment Components)

Decompositional Model-Based Testing


c

MBT Tool

S(e)

Model-Based Mock:

Inject fault,

Intercept, log, and forward

e

(Environment Components)

Benjamin Vedder,
Testing Safety-Critical Systems Using Fault Injection and Property-Based Testing,
Licentiate Thesis, Halmstad University, 2015.

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.mostowski@hh.se