

Model-based Quality Assurance of Automotive Software

Jan Jürjens¹, Daniel Reiss², David Trachtenherz³

¹ Open University (GB) and Microsoft Research (Cambridge)

² Elektrobit (Germany)

³ TU Munich (Germany)



J.Jurjens@open.ac.uk



The Problem (Meta-level M3)

- Research in Software Engineering is largely “hype-driven”.
- Research activities largely consist of solution development [cf Wieringa].
- Very little independent scientific validation, as would be expected from a scientific discipline (e.g. controlled and repeatable experiments, preferably independently from solution developers).
- This paper tries to contribute (a bit) towards improving this situation.

The Problem (Meta-level M2)

- Model-based development using UML is one of the current “hypes”: strongly promoted by “gurus” in industry, actively researched in academia [cf previous slide].
- But does it really pay ? When / under which conditions / to what degree / which techniques exactly ... etc ?
- Very few independent, controlled and repeatable experiments regarding this question.
- This paper tries to contribute to improving this situation wrt. model-based quality assurance, with an emphasis on automotive / embedded software.

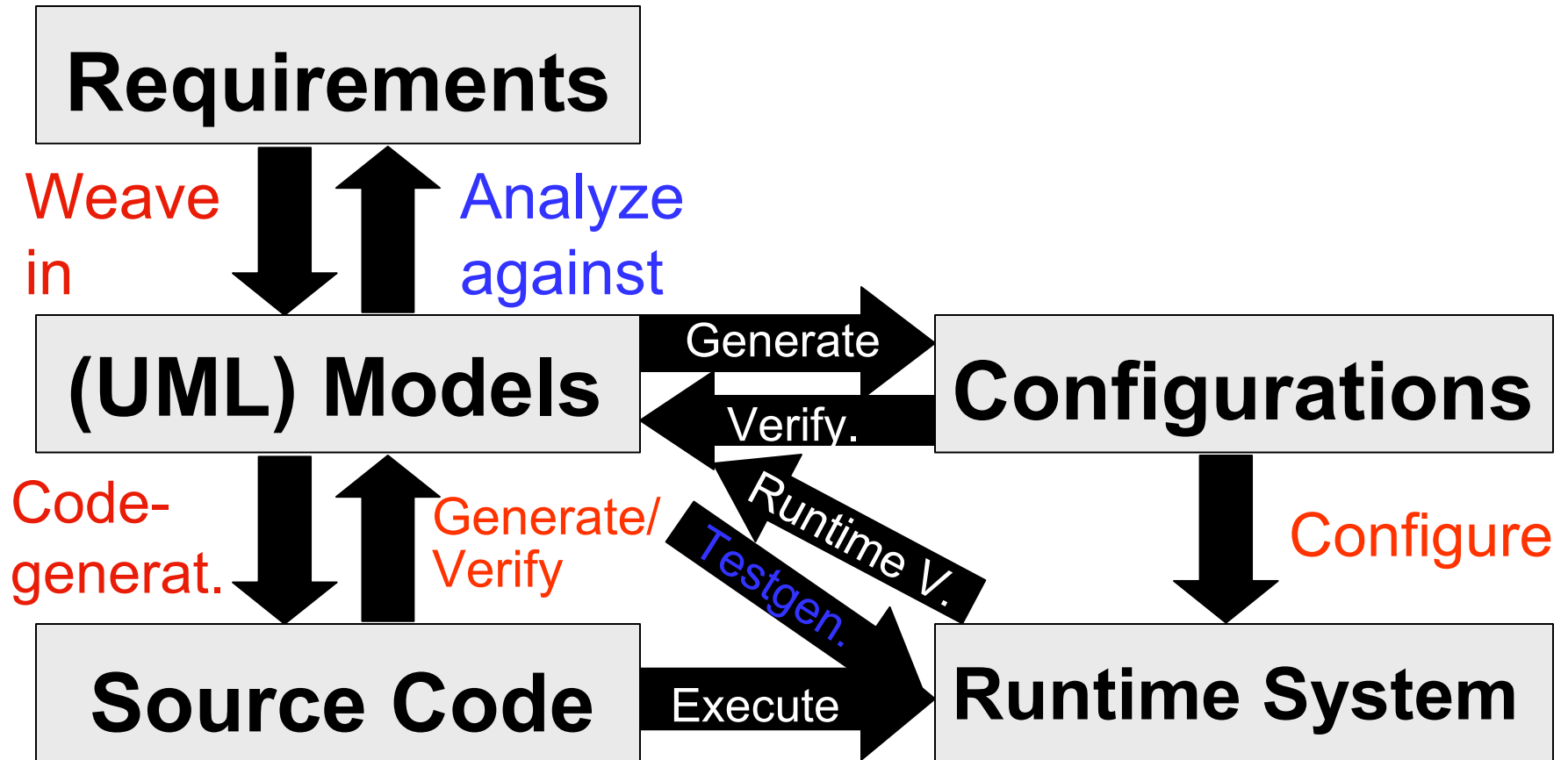
The Problem (Meta-level M1)

- Quality assurance of software consumes significant resources.
- There are high levels of assurance expected especially in safety-critical systems.
- The QA process should as far as possible be controllable (to measure degree of assurance) and repeatable (also to account for software changes).
- Model-based quality assurance seems to offer the potential to address these requirements due to a high degree of automation.
- Investigate based on a practical experiment to which extent this may be true wrt. to different QA techniques in the context of model-based development, in a comparative approach.

Automotive Software

- High safety requirements for some of the embedded software.
- Increasing complexity of the software.
- High sales numbers (compared to e.g. airplanes).
- Incentive for quality assurance as opposed to fault-tolerance by replication of functions.
- Relatively high uptake of model-based development techniques and tools.

Model-based System Assurance

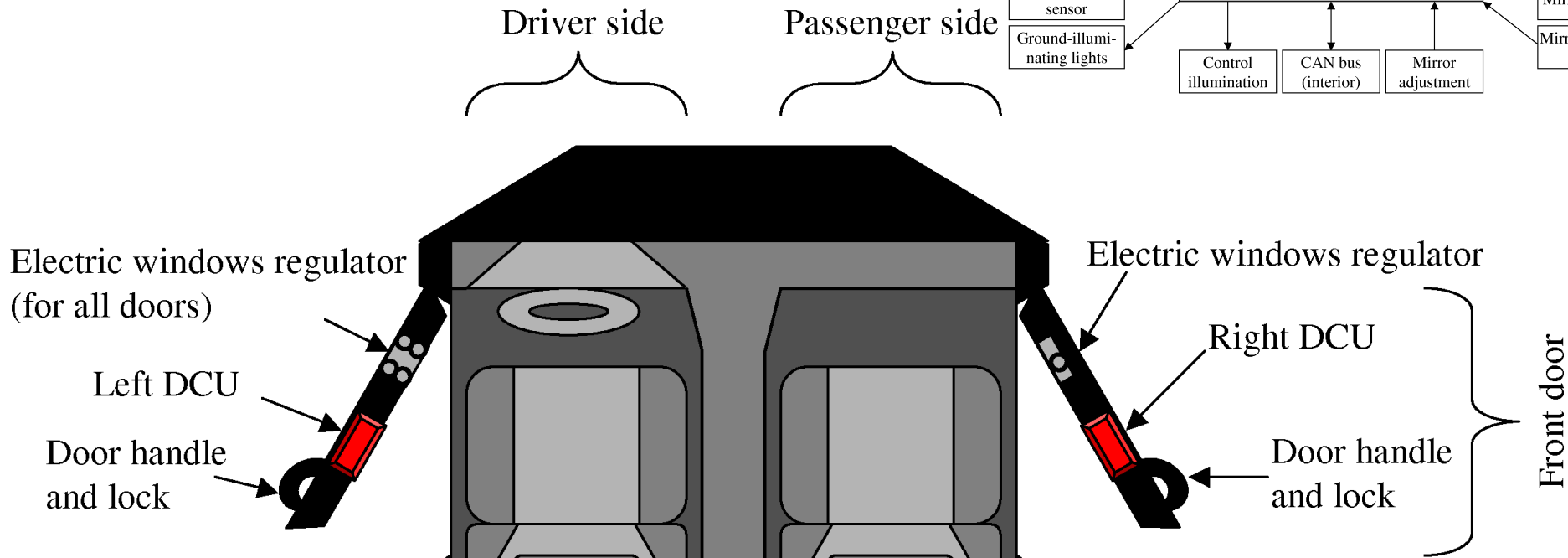
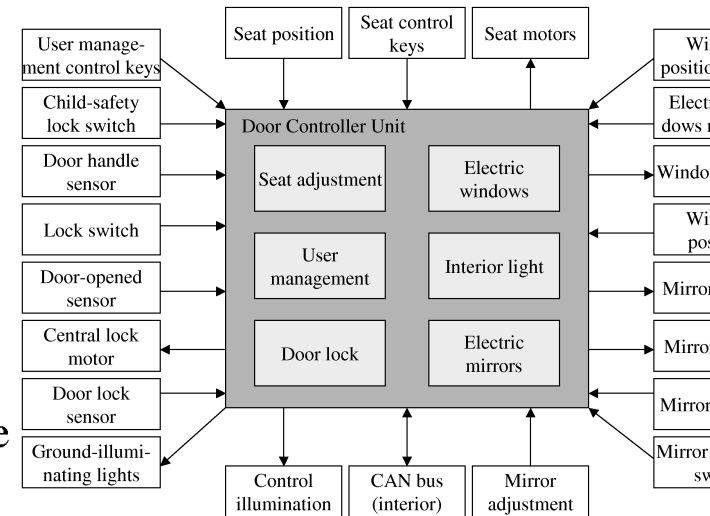


Case study: Door controller (M0)

Industrial specification [Paech et al, Fraunhofer 2002]. Here:

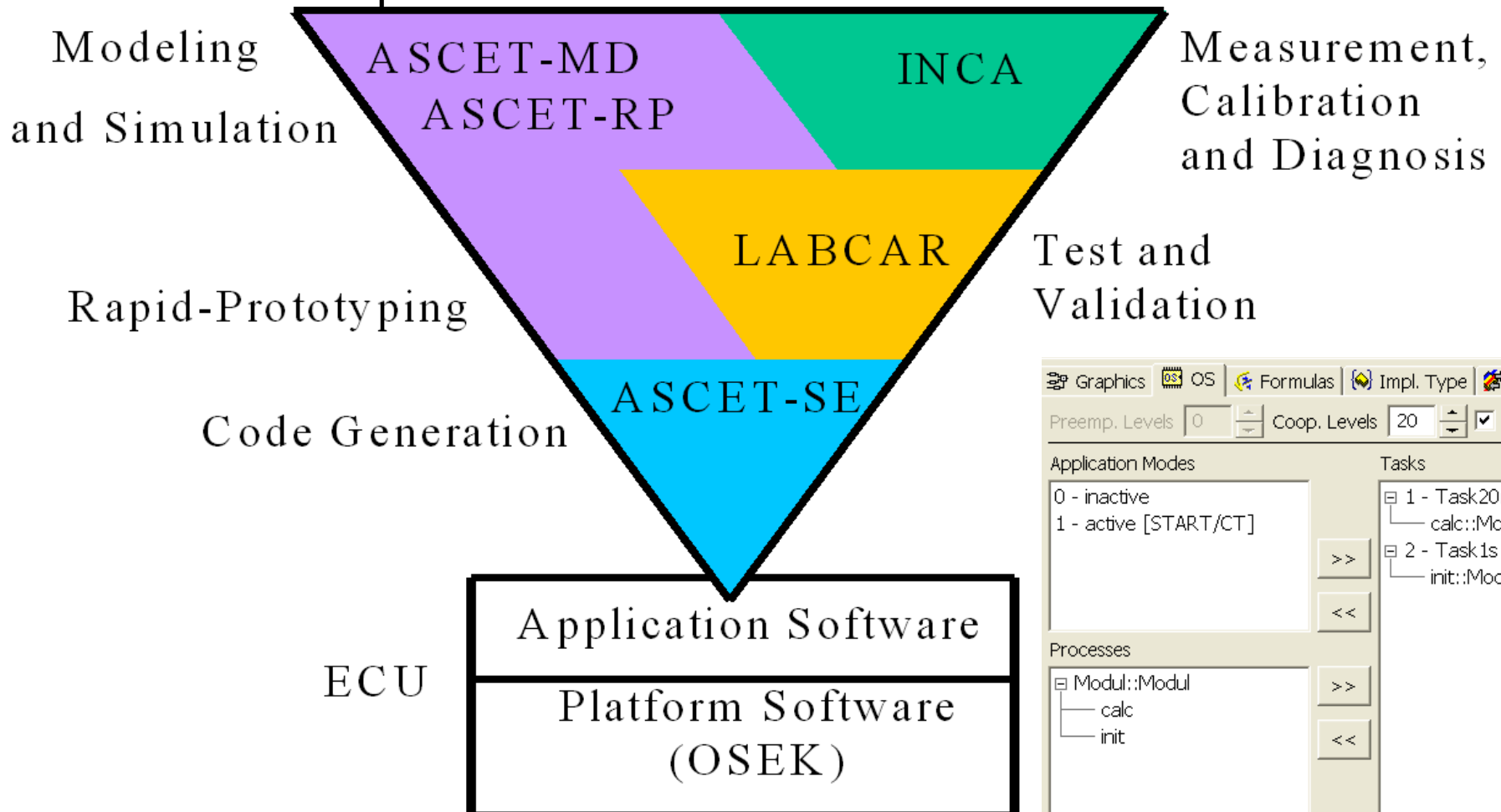
- Window lifter (including crush guard)
- Door locking/unlocking

Two door controllers communicating via CAN bus

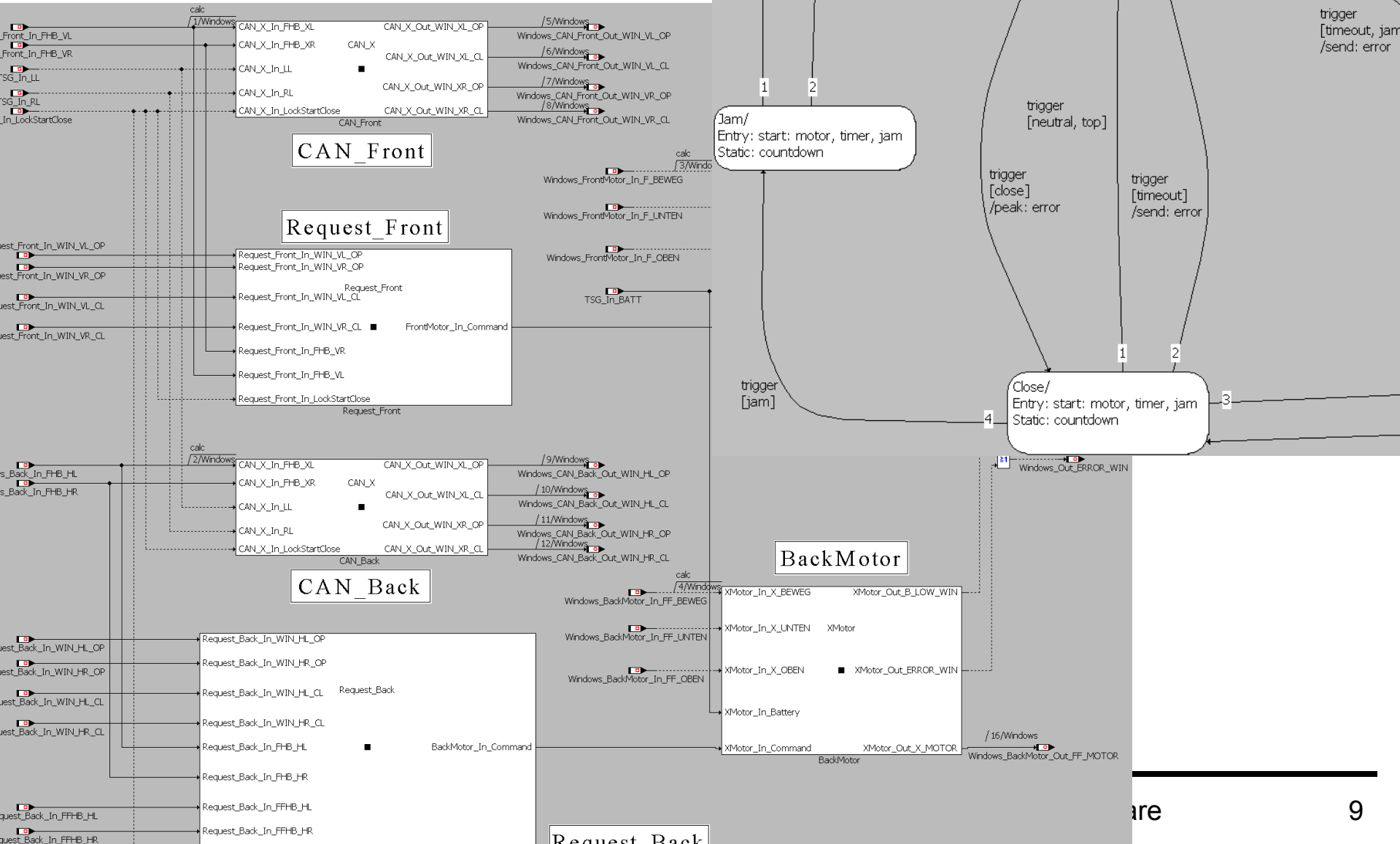


ASCET

- Commercial CASE tool by ETAS
- Used in automotive industry
- Event-driven operational model

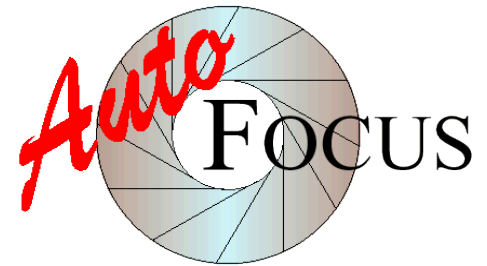


ASCET Modeling



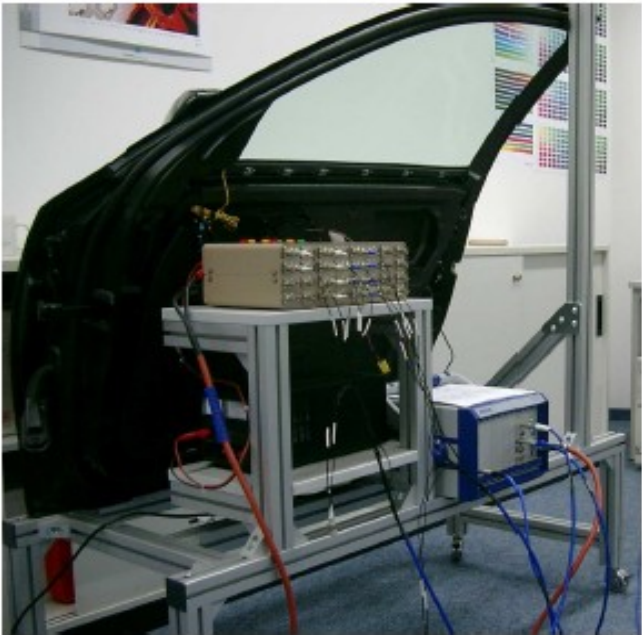
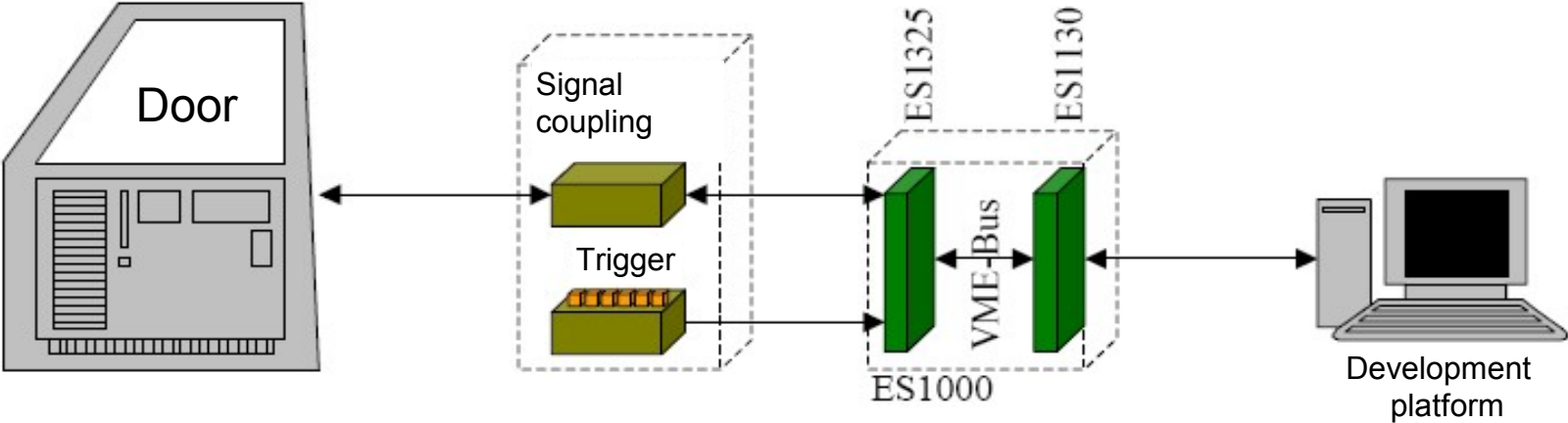
AutoFOCUS

Academic CASE tool for model-based development with UML-like notation
(<http://autofocus.informatik.tu-muenchen.de>)



- Discrete-time operational semantics
- Simulation
- Validation (Consistency, Testing, Model Checking)
- Code Generation (e.g. Java, C, Ada)
- Connection to Matlab

ASCET: Rapid Prototyping / Simulation



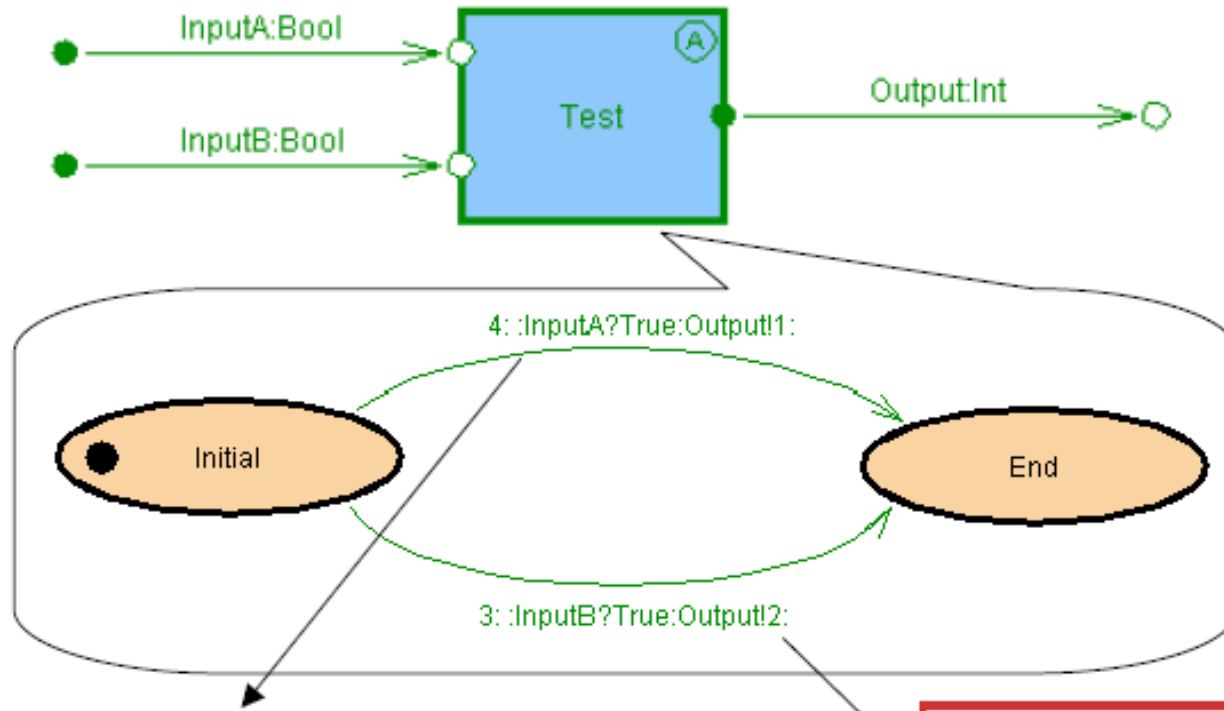
Testing in ASCET

The screenshot displays the ASCET software interface, which is used for configuring and executing tests. The interface is divided into several sections:

- Properties Panel (Left):** Contains fields for 'Standard', 'Name', 'ID =', 'Description', and 'Text'.
- Test Sequence List (Bottom Left):** Shows a sequence of 12 test steps with their respective durations in seconds:

Teststep	Zeit [Sekunde]
Teststep 1	0
Teststep 2	0,2
Teststep 3	0,4
Teststep 4	1,0
Teststep 5	1,2
Teststep 6	1,8
Teststep 7	2,0
Teststep 8	2,2
Teststep 9	2,4
Teststep 10	2,6
Teststep 11	2,8
Teststep 12	3,0
- Environment Diagram (Top Right):** A hierarchical tree structure showing the test environment. The root is 'Environment', which branches into 'CTE_Windows_In_LockStartClose' and 'CTE_Windows_Request_Back_In_FFHB_HL'. Below these are various parameters: 'BATT_gt105le180', 'false', 'true', 'FHB_Neutral', 'FHB_DownAuto', 'FHB_DownMan', 'FHB_UpMan', 'FHB_UpAuto', and 'FHB_Neutral'.
- Timing Diagram (Bottom Right):** A grid-based diagram showing the execution of the test sequence over time. The vertical axis represents time in seconds, and the horizontal axis represents the test steps. Black dots indicate the start and end of each test step.

Model Checking (AutoFOCUS)



```
state==Initial
&& next(state)==End
&& is_Msg(InputA)
&& getVal(InputA)==True
&& getVal(next(Ouput))==1
```



```
not( state==Initial
&& is_Msg(InputA)
&& getVal(InputA)==True )
```

```
&& state==Initial
&& next(state)==End
&& is_Msg(InputB)
&& getVal(InputB)==True
&& getVal(next(Ouput))==2
```

Evaluation: Modeling Effort

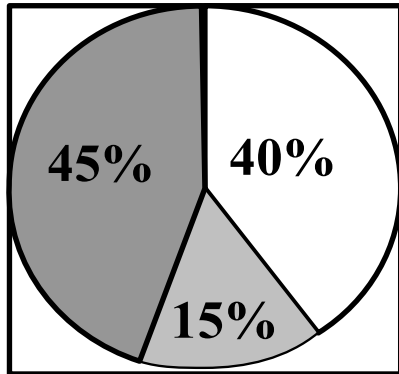
Modeling	ASCET	AutoFocus (Including Specification)
Training	1 week	1 day
Door lock	1.5 weeks	3 weeks
Window lifter	1.5 weeks	3 weeks
Interior light	0.5 weeks	1 week

Evaluation: QA Effort and Results

Method	Time effort (days)	Error count
Simulation (ASCET)	3	10
White box testing (ASCET)	7	5
Simulation (AutoFocus)	3	5
Model Checking (AutoFocus)	10	5

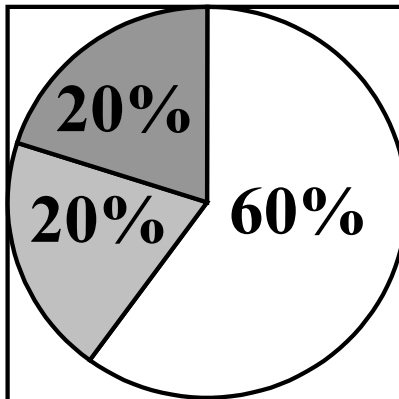
Evaluation: Effort Distribution

- Testing



- Create test cases
- Execute test cases
- Interpret test cases

- Model Checking



- Create properties
- Execute modelchecker
- Interpret counter examples

Modelchecking: Experiences

State explosion problem

- compositional modelchecking

Modelling abstractions:

- execution timer
- equivalence classes for values

➔ compromise between abstraction and verification efficiency

Classes of Bugs

Simulation (ASCET [10] / AutoFOCUS [5]):

- wrong priority definition
- wrong value communicated
- logical error at branchings
- wrong execution sequence (ASCET)

Coverage analysis / rapid prototyping (ASCET [7])

- same bugs as in simulation
- unreachable code
- wrong assumptions on hardware

Modelchecking (AutoFOCUS [5])

- synchronization error for concurrent components
- wrong evaluation of logical expressions

➔ Testing takes real hardware into account;
modelchecking finds spurious / obscure bugs

➔ combination brings synergies



Evaluation: Model vs Code QA

Model:

- + earlier (**less expensive** to fix flaws)
- + more abstract → **more efficient** (→ higher coverage, but at higher abstraction level)
- more abstract → may **miss flaws**
- **programmers** may **introduce flaws**
- even **code generators**, if not formally verified

Code:

- + „the real thing“ (which is executed)
- **Do both where feasible.**



Evaluation: General Comparison

Modelchecking

Examines an abstract model

Cheap and early verification
(without setting up complex in-the-loop-test environments)

Proof of correctness of properties possible

Uses selected user specific properties

Testing

Examines a physical or concrete system

In-the-loop-tests take place in an environment near to the real one

No proof of correctness of properties possible

Uses often many, superficial test cases



Evaluation (M3)

- Semi-independent: researchers in model-based development, from AutoFocus group
- Repeatability: experimental data available from <http://mcs.open.ac.uk/jj2924/publications/experiments/autoqa> (ref 10 in paper)
- Comparative SE: use same or different developers ?
- Qualitative study, so no claim to statistical significance.

Related Work

Practical experiments on model-based QA in:

- automotive: Pretschner et al. (ICSE 2005: model-based testing with AutoFocus); Kropf (CAV 2007)
- security: Best, Jurjens, Nuseibeh (ICSE 2007; information systems); Jurjens Schreck, Bartmann (ICSE 2008; mobile systems); Jurjens, Rumm (M.Med.Inf 2008; e-health-card)
- general: Halling, Biffi, Grunbacher (METRICS 2003; requirements analysis); Brat, Drusinsky, Giannakopoulou et al. (FMSD 2004; Martian Rover); Cheng et al. (Models 2005; model analysis); Bradbury, Cordy, Dingel (PASTE 2005; testing vs formal analysis); Denney, Fischer, Schumann (IJAIT 2006; ATPs); Mouchawrab, Briand, Labiche (ESEM 2007; model-based testing)

Conclusions

Model-based QA of automotive software:

- Model-checking and model-based testing complementary.
- Model-based testing quickly excludes large classes of flaws.
- Model-checking exhaustively checks user-defined sophisticated property.

Ongoing work with Microsoft Research Cambridge:
assurance for cryptoprotocol implementations.

ADVERTISEMENT: Postdoc / PhD positions in model-based security !



Questions?

More information
(papers, slides,
tool etc.):

<http://www.jurjens.de/jan>

J.Jurjens@open.ac.uk