The Thousands Shades of Model-Based Testing: Pitfalls, Challenges, and Guidelines

University of Luxembourg
Interdisciplinary Centre for Security, Reliability and Trust
Software Verification and Validation Lab (www.svv.lu)

October 23rd, 2013
UCAAT, Paris, France

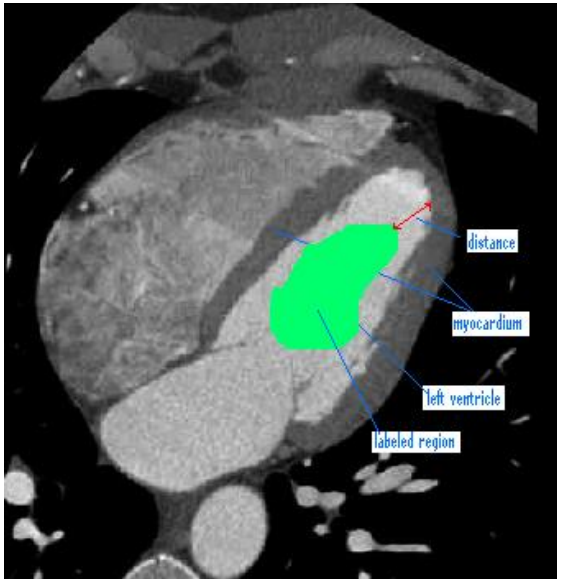
Lionel Briand, IEEE Fellow
FNR PEARL Chair

Diversity


5
4
3
2
1


Carrier
3:56 PM
Partager
de Guichet
de Guichet
Grand-Duché de Luxembourg
Citoyens
Entreprises
Bürger
LE GOUVERNEMENT DU GRAND-DUCHÉ DE LUXEMBOURG
Classic version | Helpdesk
Organismes
de Guichet
Actualités
Autres


ENTER AMOUNT


distance
myocardium
left ventricle
labeled region

Objectives

- Report on 20 years of experience in automated software testing & verification research and innovation
- Explain why and how model-based testing is in many situations—though not always—the best solution to test automation
- Show why and how the tailoring of the MBT process and technology to context is a key success factor
- Present representative project examples
- Identify challenges and lessons learned
- Guidelines

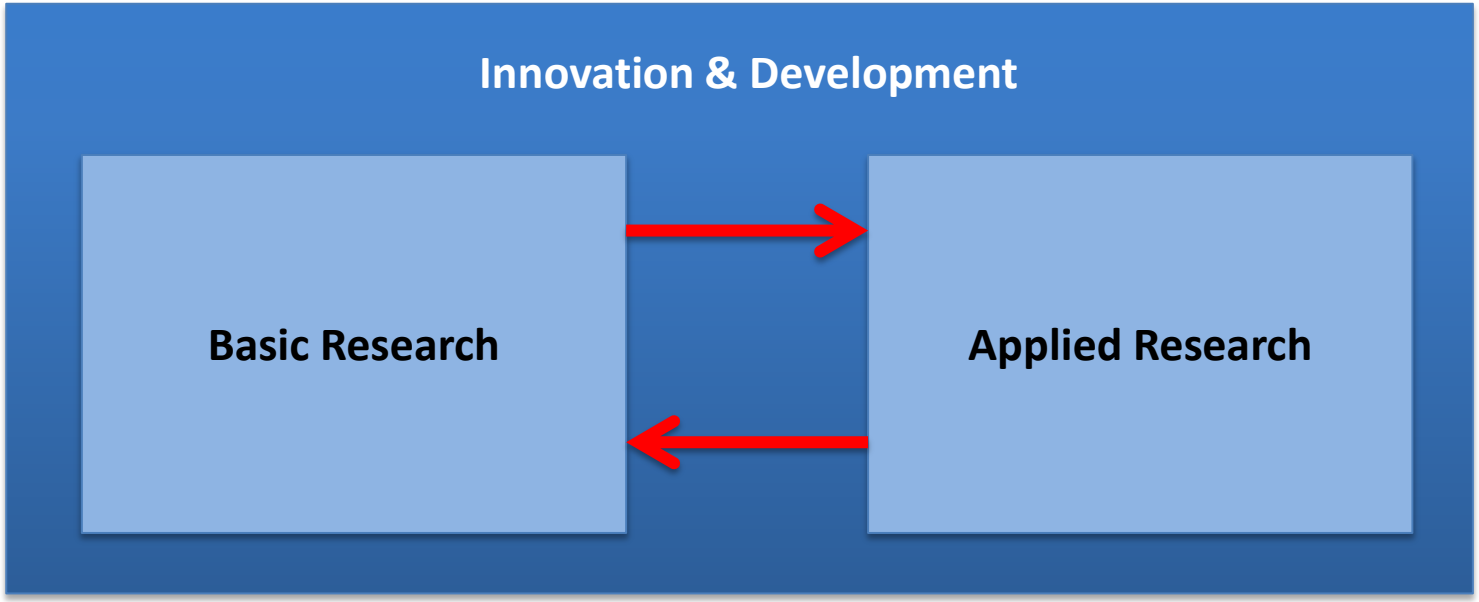
Testing vs. Verification

- Testing: The process of executing software with the intent of finding and correcting defects
- Verification: The process of analyzing a representation or model of the system specification or design in order to reason about its properties
- Verification takes place in earlier phases than testing: feasibility of requirements, design decisions …
- Will focus mostly on testing here, but there are many common challenges, technologies, and lessons learned

SnT Software Verification and Validation Lab

- SnT centre, Est. 2009: Interdisciplinary, ICT security-reliability-trust
- 200 scientists and Ph.D. candidates, 20 industry partners
- SVV Lab: Established January 2012, www.svv.lu
- 20 scientists (Research scientists, associates, and PhD candidates)
- Industry-relevant research on system dependability: security, safety, reliability
- Six partners: Cetrel, CTIE, Delphi, SES, IEE, Hitec …

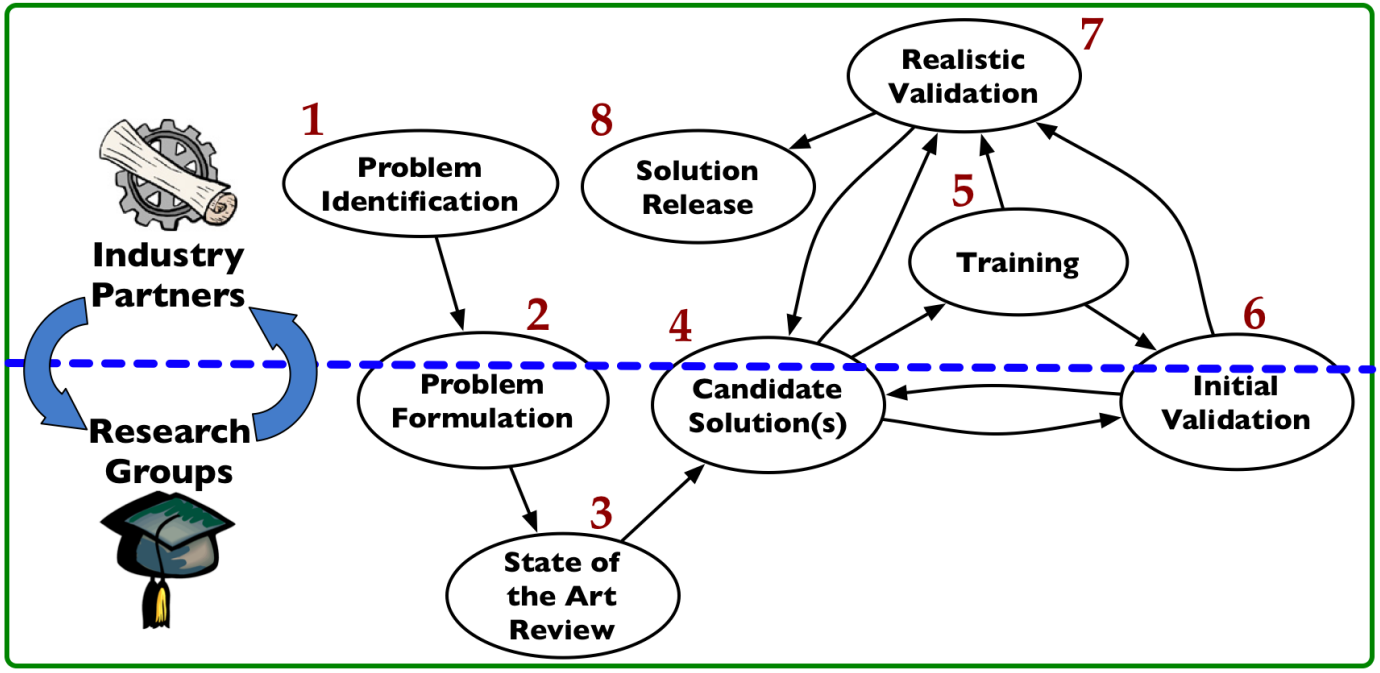
An Effective, Collaborative Model of Research and Innovation



- Basic and applied research take place in a rich context
- Basic Research is also driven by problems raised by applied research
- Main motivation for SnT's partnership program

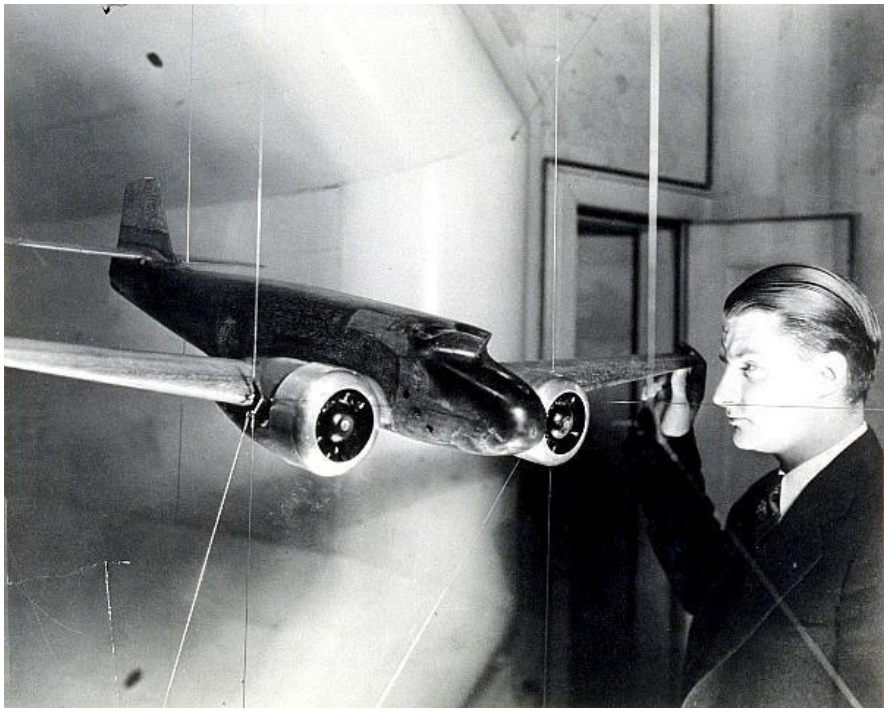
Collaboration in Practice

- Research informed by practice
- Well-defined problems in context
- Realistic evaluation
- Long term industrial collaborations



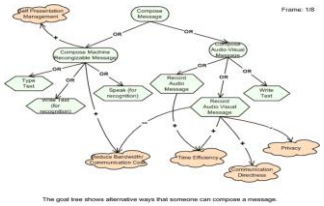
“Model-based”?

- All engineering disciplines rely on abstraction and therefore models
- In many cases, it is the only way to effectively automate testing or verification => scalability
- Models have many other purposes: Communication, support requirements and design
- There are many ways to model systems and their environment
- In a given context, this choice is driven by the application domain, standards and practices, objectives, and skills

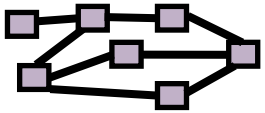


Models in Software Engineering

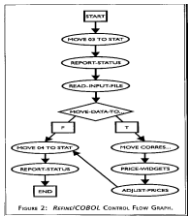
- **Model:** An abstract and analyzable description of software artifacts, created for a purpose



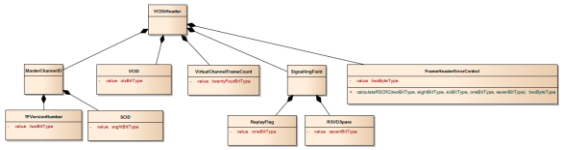
Requirements models



Architecture models



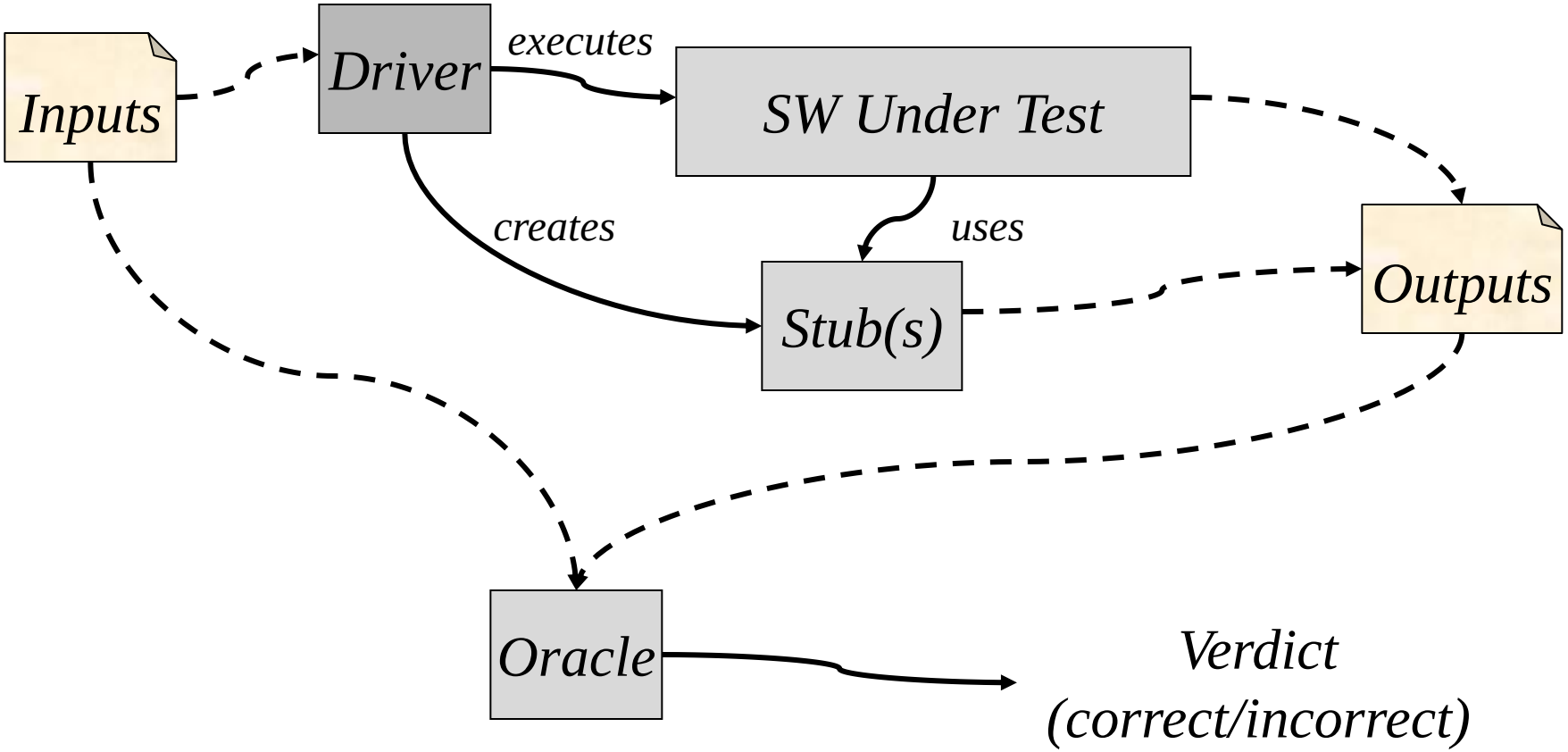
Behavioural models



Test models

- **Abstract:** Details are omitted. Partial representation. Much smaller and simpler than the artifact being modeled.
- **Analyzable:** Leads to task automation
- **Standards:** UML, SysML, MARTE, BPMN, …

Test Automation Problem Decomposition



Automation Needs

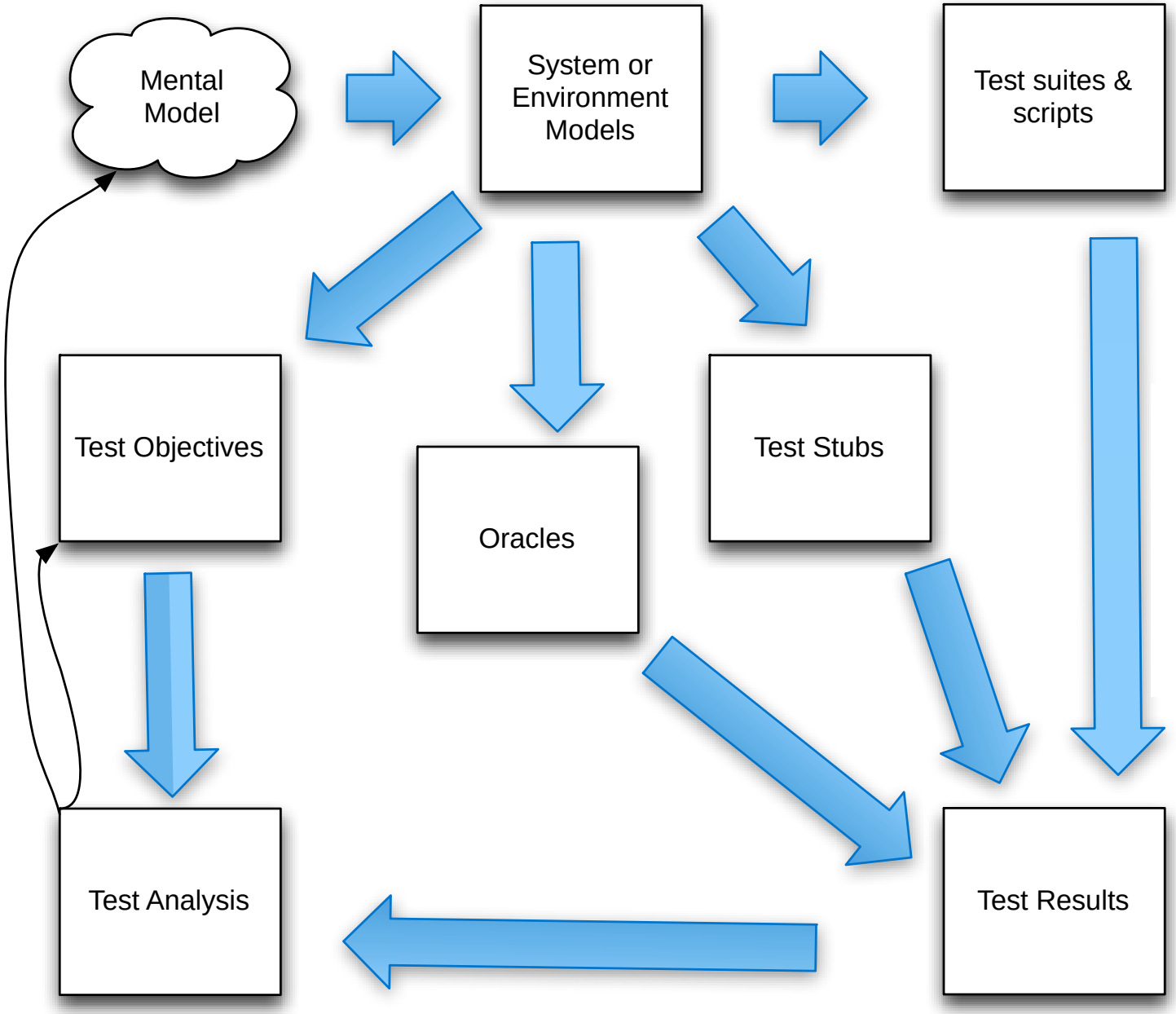
- Test case generation
- Test oracle (verdict) generation
- Test stubs generation
- Test driver generation (test execution)
- Logging and analysis of test results
- Test suite evolution, e.g., requirements changes

Common Testing Technology

- Test case generation
- Test oracle (verdict) generation
- Test stubs generation
- Test driver generation (test execution)
- Logging and analysis of test results
- Test suite evolution, e.g., requirements changes

Consequences

- Test automation not scalable
- Expensive (generation and evolution), though costs often hidden
- Error-prone
- Not systematic
- Not predictable


Mental Model
System or Environment Models
Test suites & scripts
Test Objectives
Oracles
Test Stubs
Test Analysis
Test Results

MBT Process

Testing Driven by Environment Modeling

Context

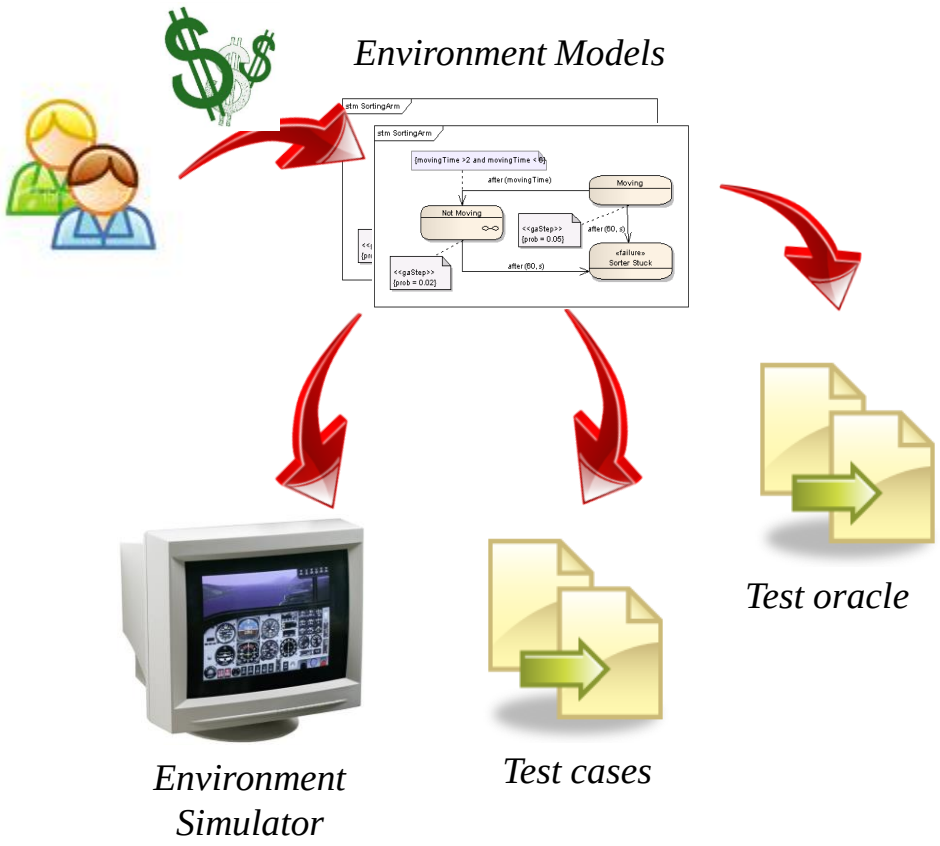
- Three-year project with two industry partners
  - Soft real-time systems: deadlines in order of hundreds of milliseconds
    - Jitter of few milliseconds acceptable
  - Automation of test cases and oracle generation, environment simulation

Tomra – Bottle Recycling Machine

WesternGeco – Marine Seismic Acquisition System

Environment Modeling and Simulation

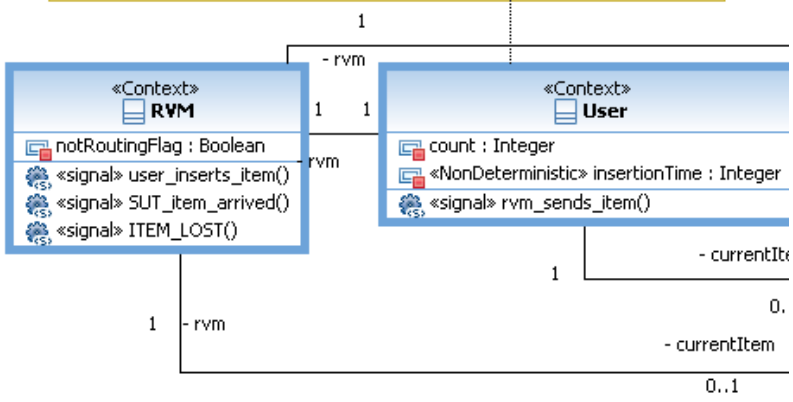
- Independent
  - Black-box
- Behavior driven by environment
  - Environment model
- Test Engineers: Software engineers
- No use of Matlab/Simulink
- One model for
  - Environment simulator
  - Test cases and oracles
- UML profile (+ limited use of MARTE)

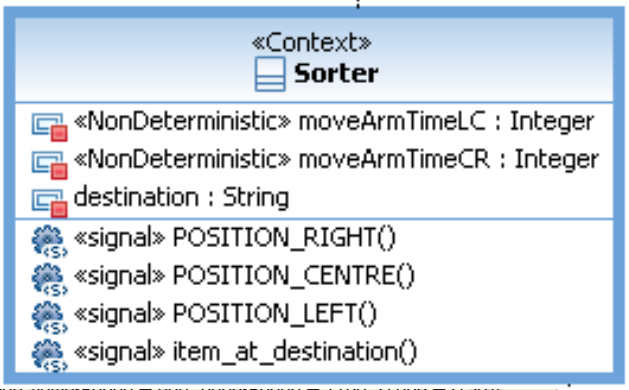


Domain Model

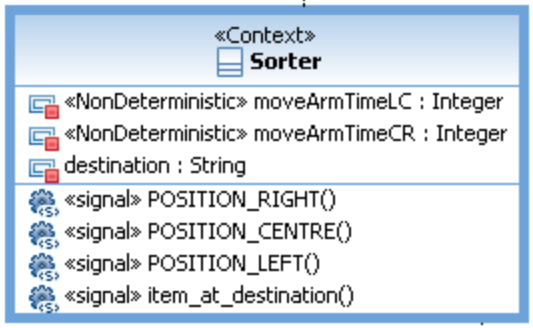
«NonDeterministic»
User::insertionTime {lowerBound = 1, upperBound = 10000, scope = state}

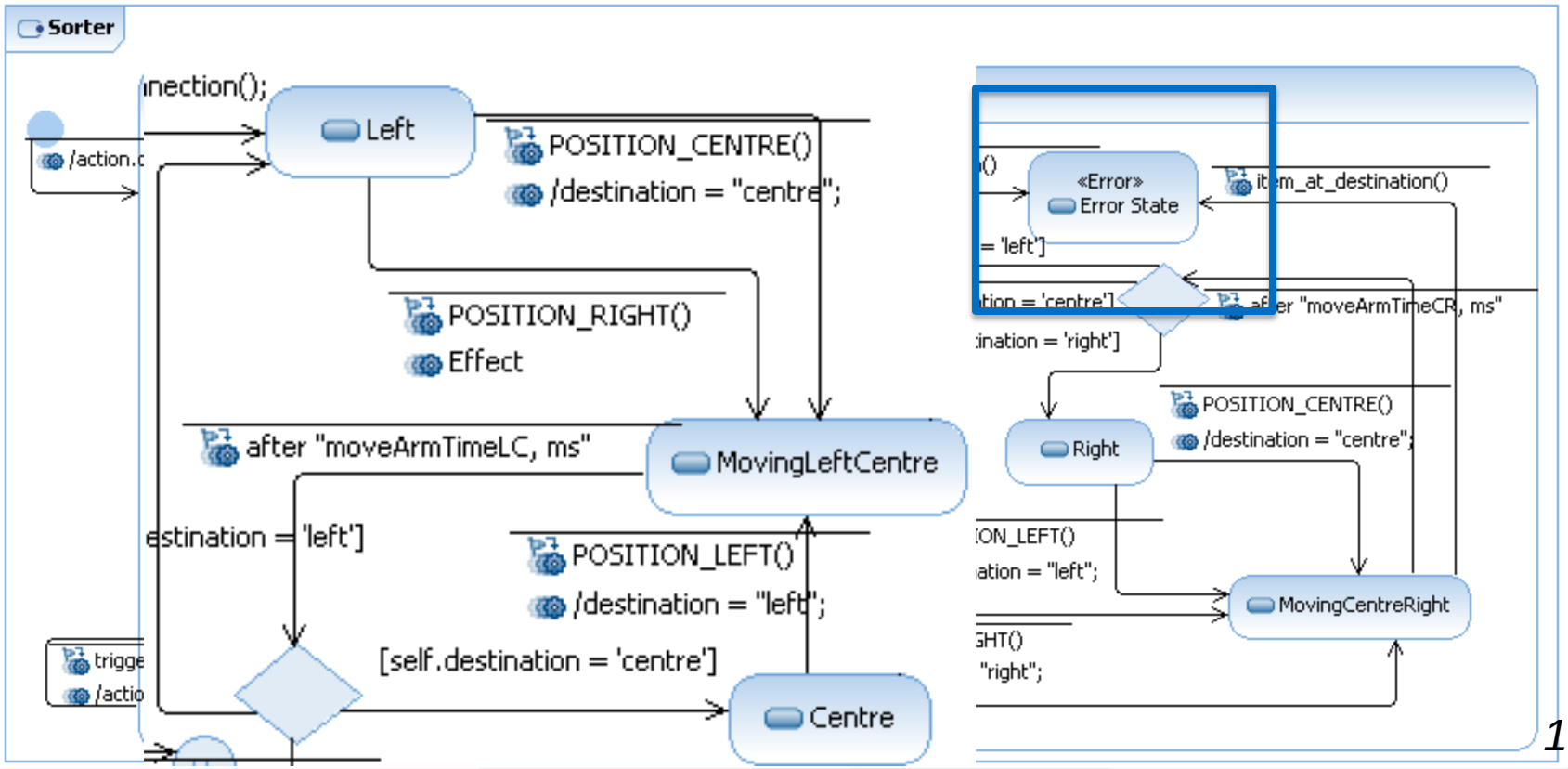
«NonDeterministic»
Sorter::moveArmTimeLC {lowerBound = 280, upperBound = 320, scope = state}
Sorter::moveArmTimeCR {lowerBound = 280, upperBound = 320, scope = state}





Behavior Model





Automated Test Case Generation and Oracles

- Test cases are defined by
  - Simulation configuration
  - Environment configuration
- Environment Configuration
  - Number of instances to be created for each component in the domain model (e.g., the number of sensors)
- Simulator Configuration
  - Setting of non-deterministic attribute values
- Bring the system state to an error state by searching for appropriate values for non-deterministic environment attributes
- Search metaheuristics to search the test case space
- Test oracle: Environment model error states (state invariants)

Testing Closed Loop Controllers

Complexity and amount of software used on vehicles' Electronic Control Units (ECUs) grow rapidly

securityandtrust.lu

Comfort and variety

More functions

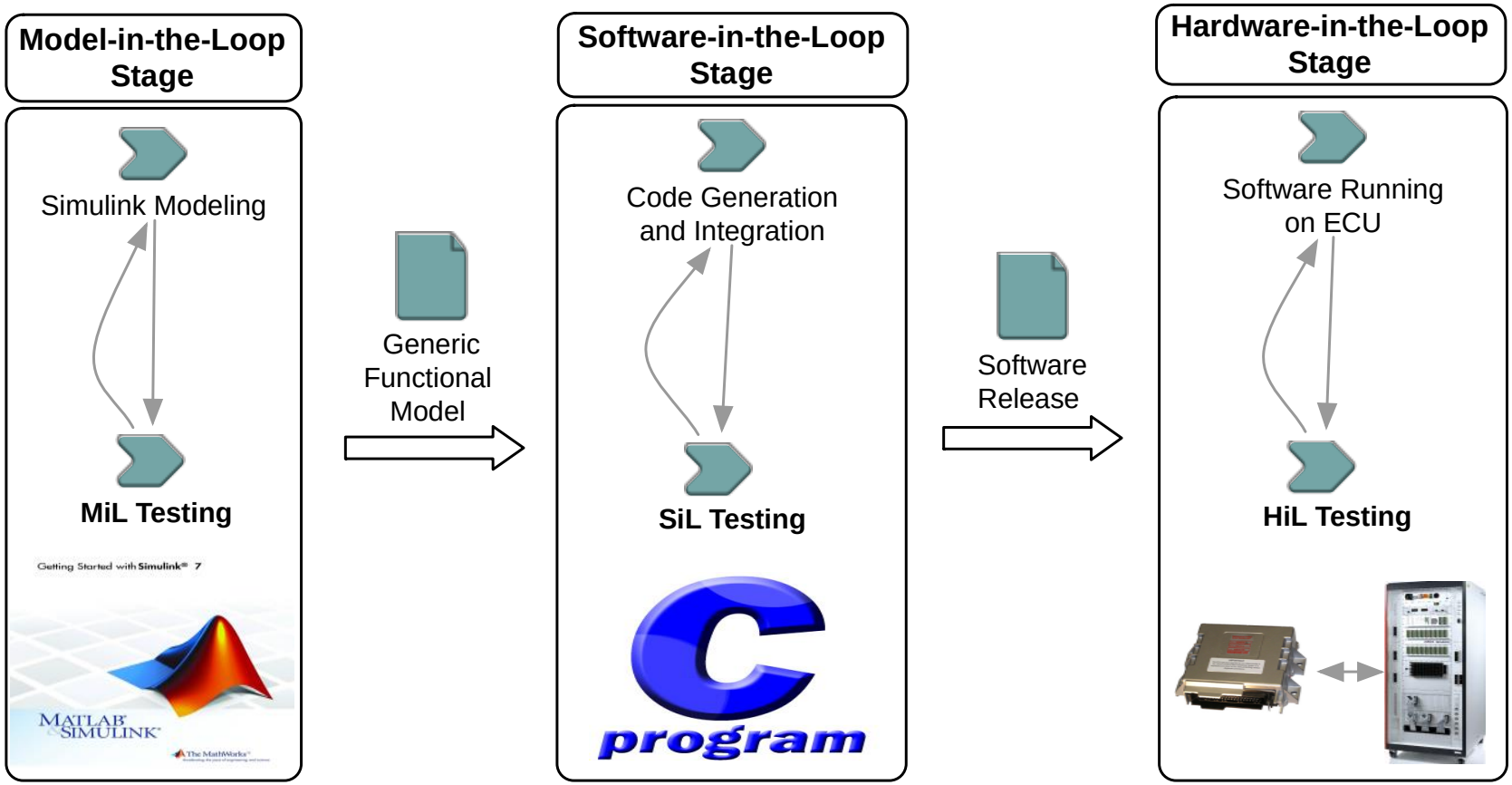
Safety and reliability

Faster time-to-market

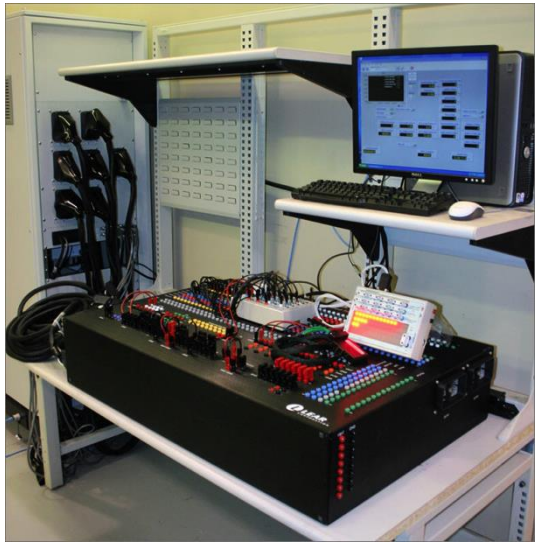
Greenhouse gas emission laws

Less fuel consumption

Three major software development stages in the automotive domain



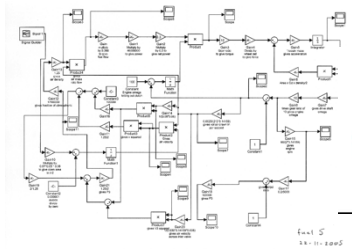
Major Challenges in MiL-SiL-HiL Testing



- Manual test case generation
- Complex functions at MiL, and large and integrated software/embedded systems at HiL
- Lack of precise requirements and testing objectives
- Hard to interpret the testing results

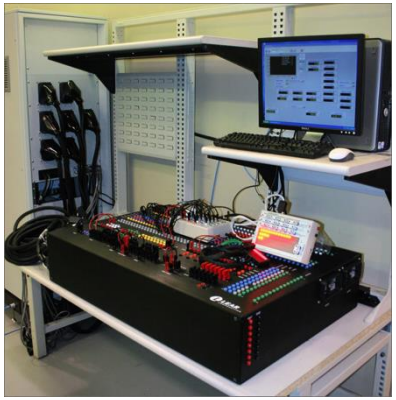
MiL testing

Requirements



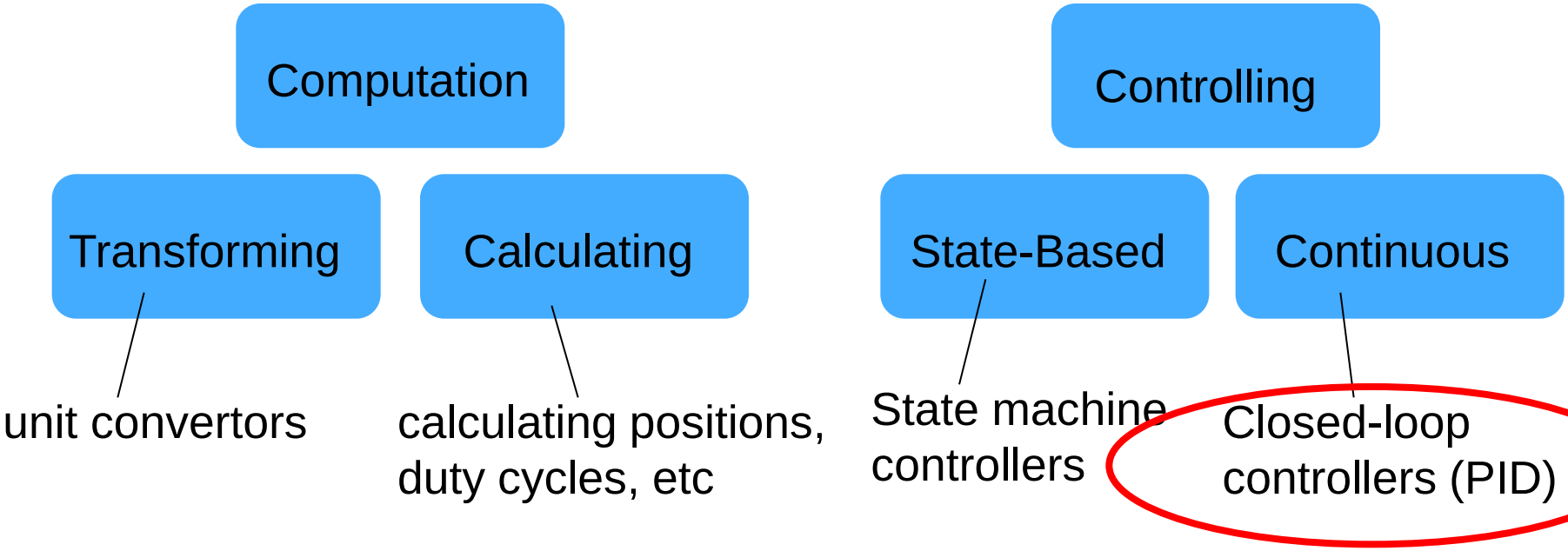
MATLAB
SIMULINK

Individual Functions



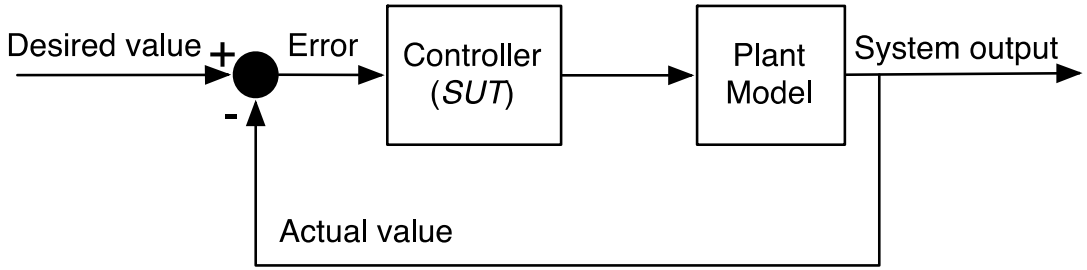
The ultimate goal of MiL testing is to ensure that individual functions behave correctly and timely on any hardware configuration

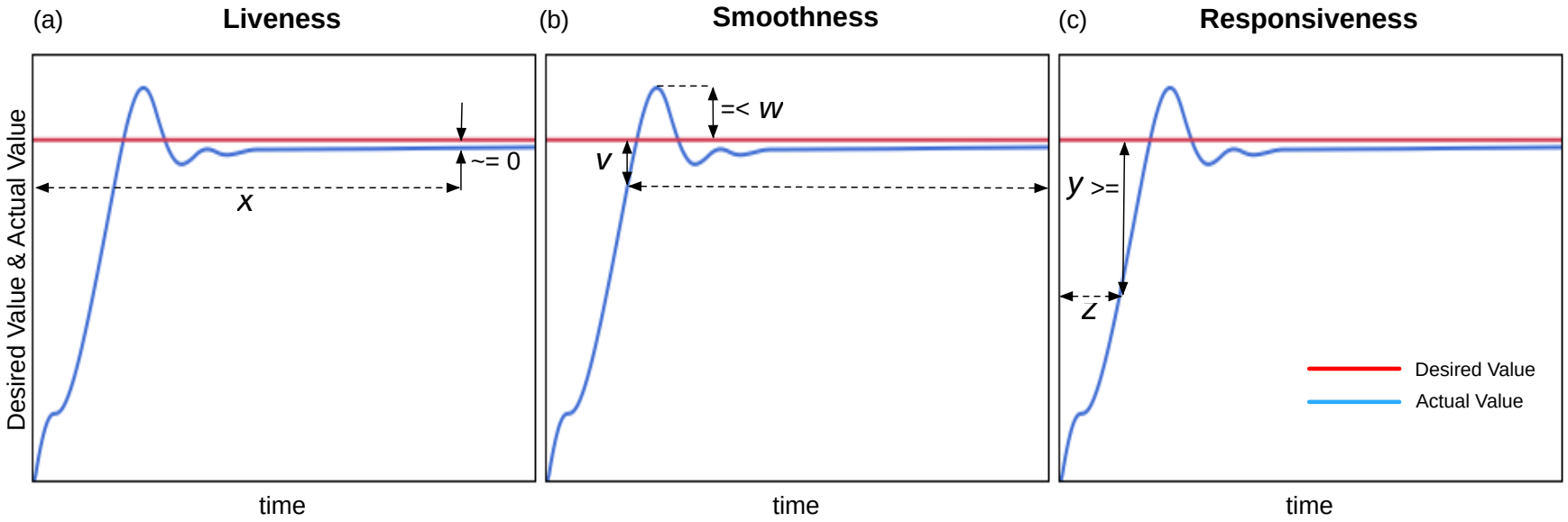
A Taxonomy of Automotive Functions



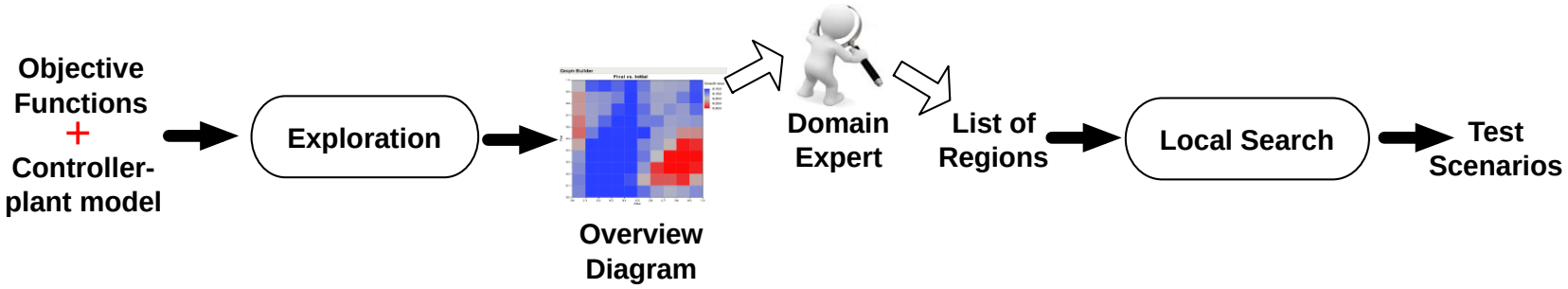
Different testing strategies are required for different types of functions

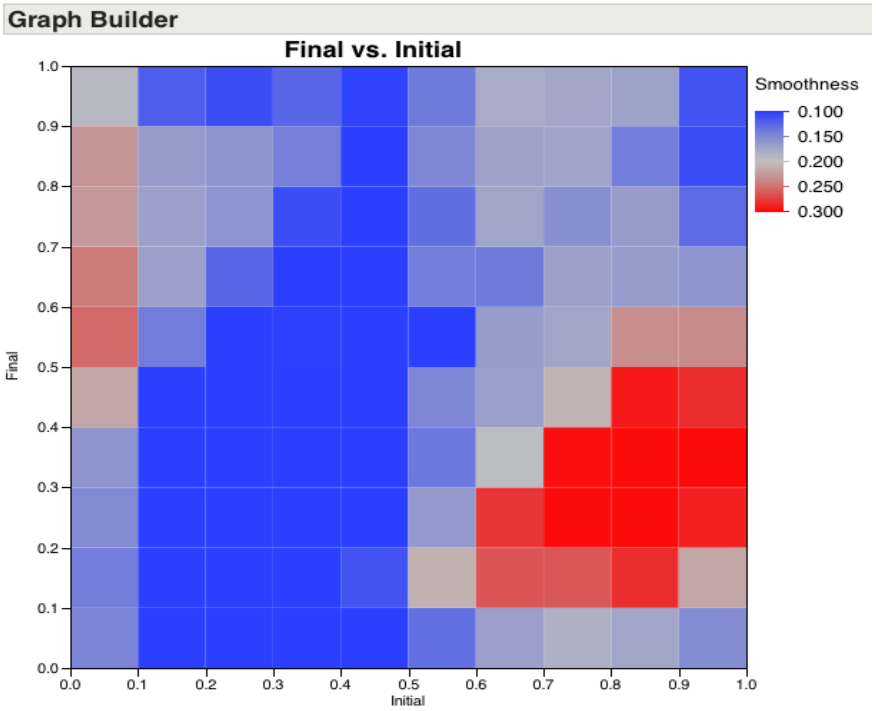
Controller Plant Model and its Requirements

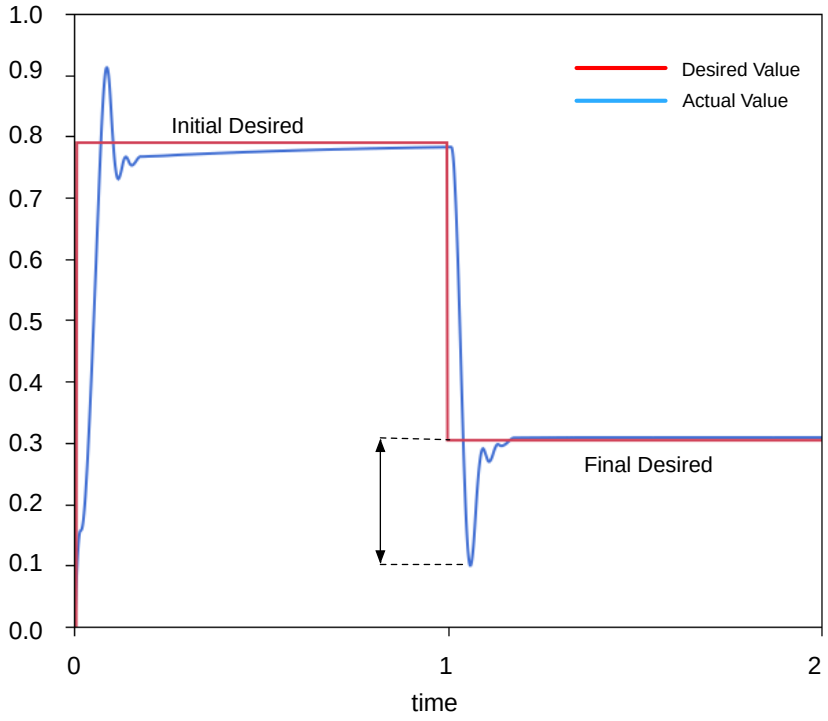




MiL-Testing of Continuous Controllers

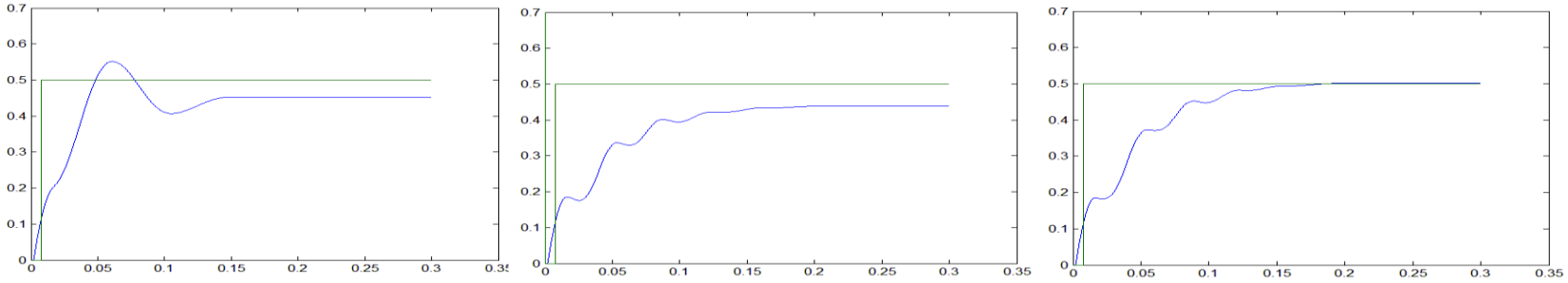






Search Strategy

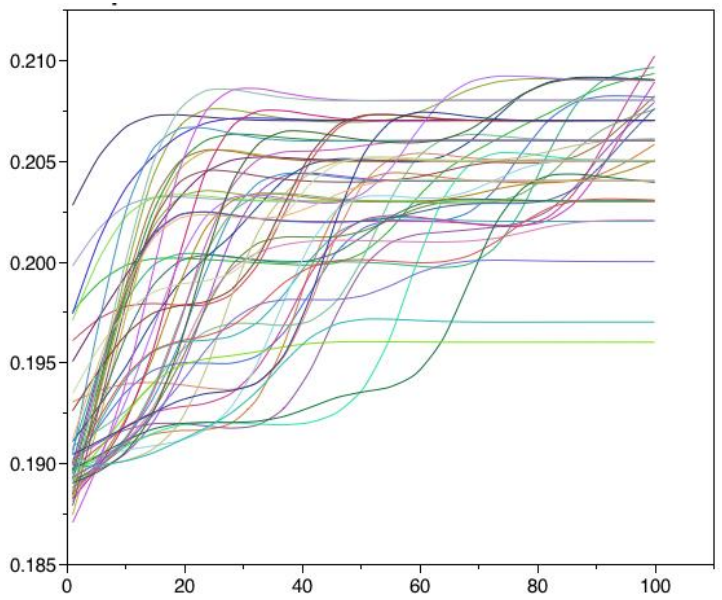
- **Search:**
  - Inputs: Initial and desired values, configuration parameters
  - Example search technique: (1+1) EA (Evolutionary Algorithm)
- **Search Objective:**
  - Find worst case scenarios for liveness, smoothness, responsiveness -> objective functions
  - For each scenario -> simulation



- **Result:**
  - worst case scenarios or values to the input variables that are more likely to break the requirement at MiL level
  - stress test cases based on actual hardware (HiL)

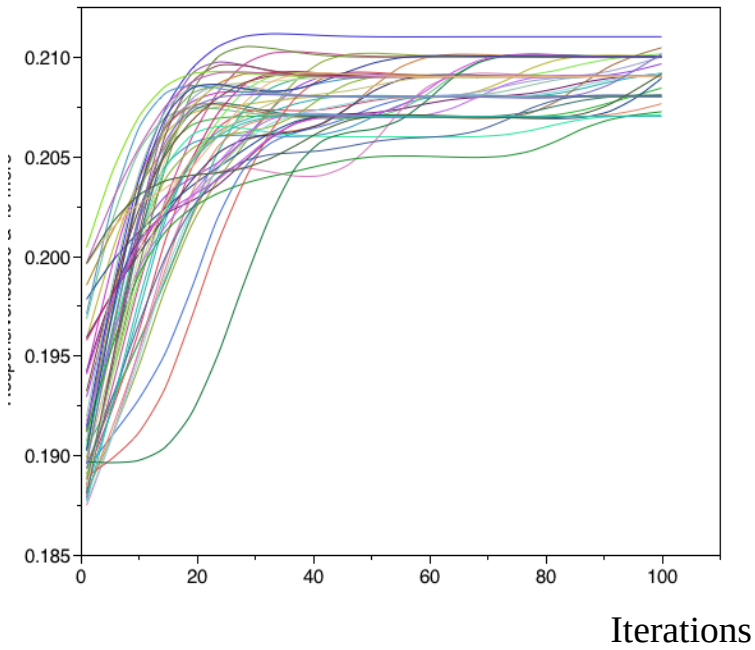
Random Search vs. (1+1)EA
Example with Responsiveness Analysis

Responsiveness



Random

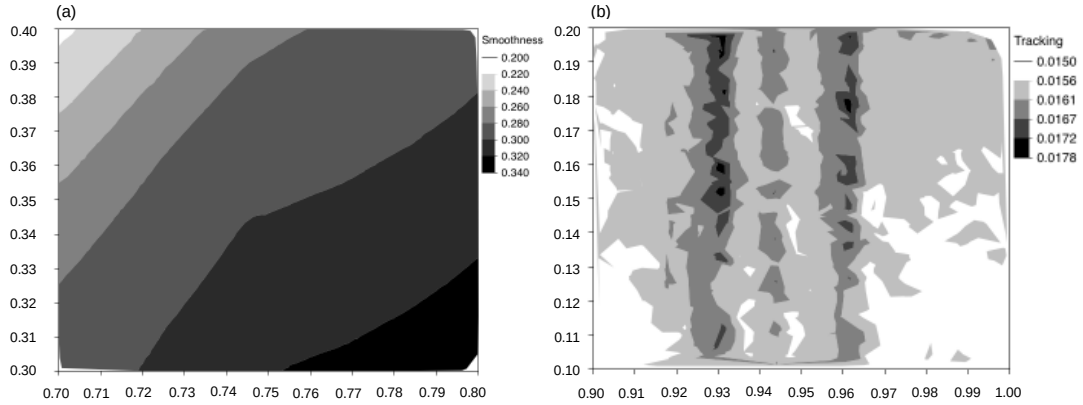
Responsiveness



(1+1) EA

Conclusions

- We found much worse scenarios during MiL testing than our partner had found so far
- They are running them at the HiL level, where testing is much more expensive: MiL results -> test selection for HiL
- But further research is needed:
  - To deal with the many configuration parameters
  - To dynamically adjust search algorithms in different subregions



MBT Projects Sample (< 5 years)

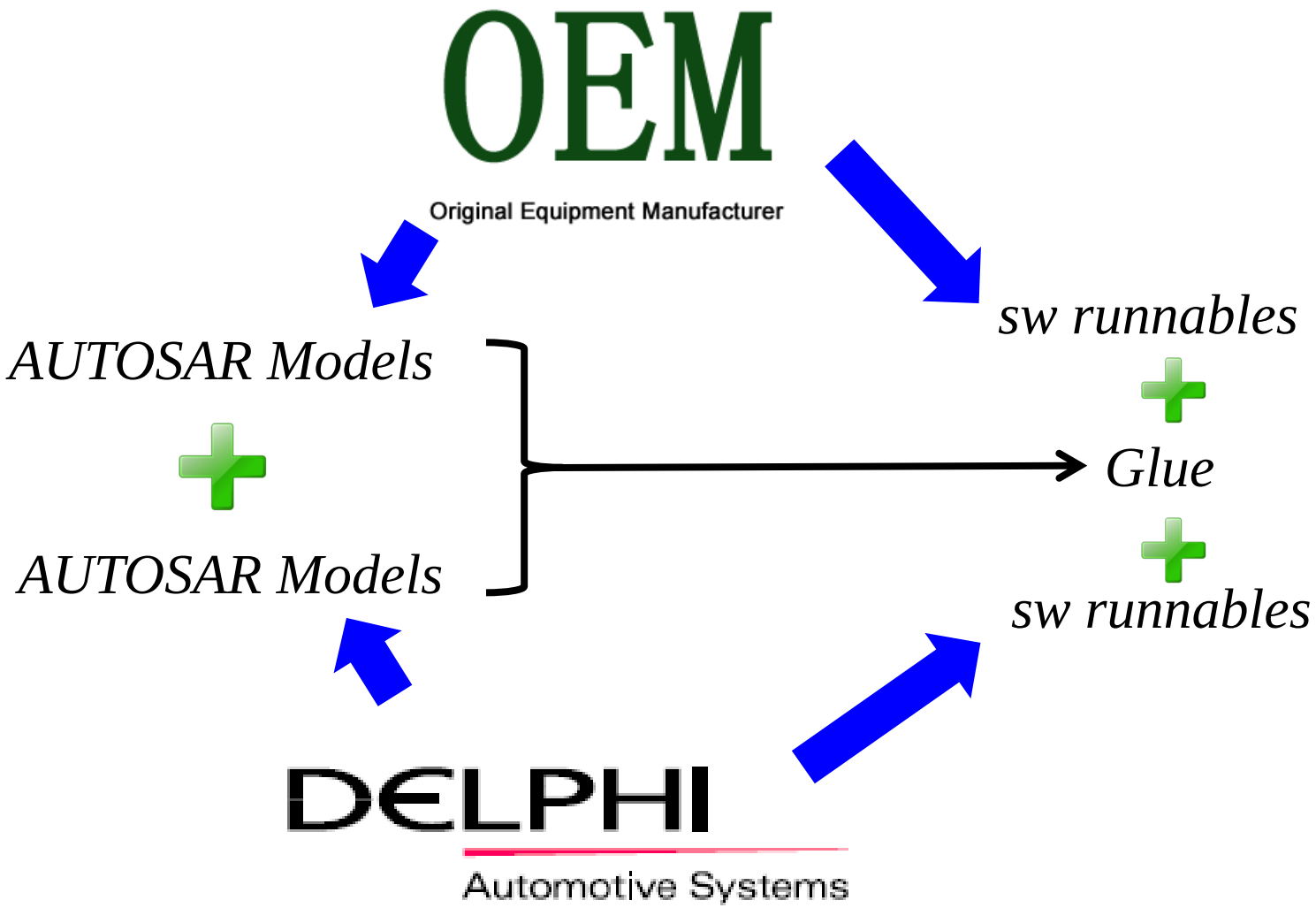
Company	Domain	Objective	Notation	Automation
ABB	Robot controller	Safety	UML	Constraint Solver
Cisco	Video conference	Robustness	UML profile	Metaheuristic
Kongsberg Maritime	Oil&gas, safety critical drivers	CPU usage	UML+MARTE	Constraint Solver
WesternGeco	Marine seismic acquisition	Functional testing	UML profile + MARTE	Metaheuristic
SES	Satellite operator	Functional testing	UML profile	Metaheuristic
Delphi	Automotive systems	Testing safety+performance	Matlab/Simulink	Metaheuristic
Lux. Tax department	Legal & financial	Legal Requirements testing	UML Profile	Under investigation

Verifying CPU Time Shortage Risks in Integrated Embedded Software

Today’s cars rely on integrated systems

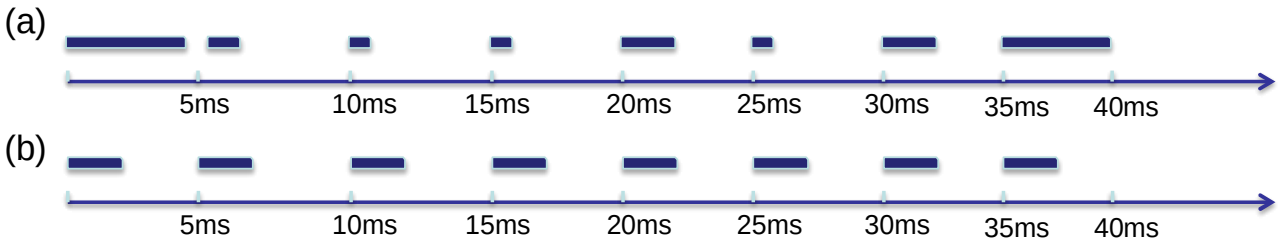
- Modular and independent development
- Many opportunities for division of labor and outsourcing
- Need for reliable and effective integration processes

Integration process in the automotive domain

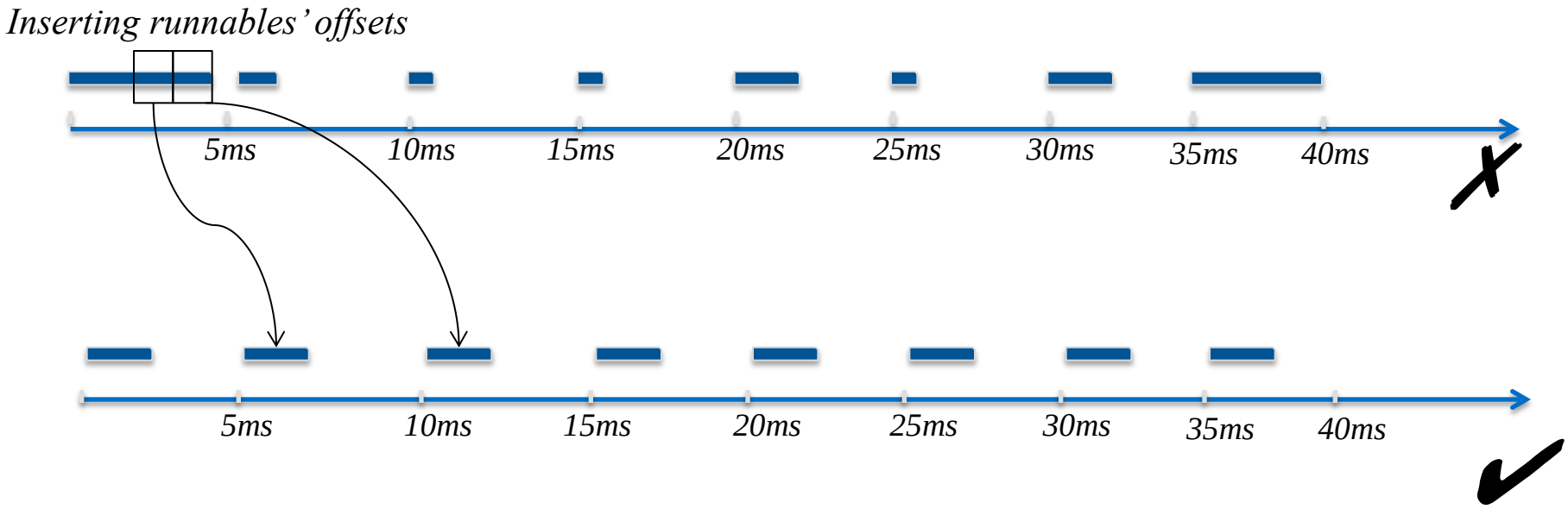


CPU Time Shortage in Integrated Embedded Software

- Challenge
  - Many OS tasks and their many runnables run within a limited available CPU time
    - The execution time of the runnables may exceed the OS cycles
- Our goal
  - Reducing the maximum CPU time used per time slot to be able to
    - Minimize the hardware cost
    - Enable addition of new functions incrementally
    - Reduce the probability of overloading the CPU in practice



Using runnable offsets (delay times)



Offsets have to be chosen such that

the maximum CPU usage per time slot is minimized, and further,

the runnables respect their period

the runnables respect the OS cycles

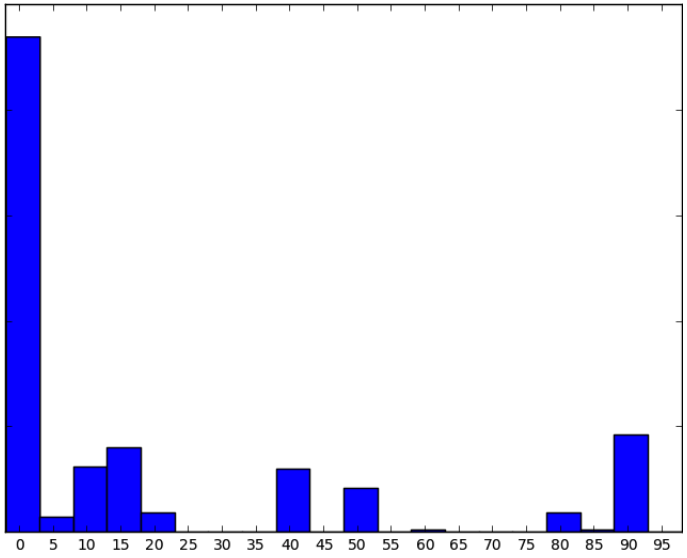
the runnables satisfy their synchronization constraints

Meta heuristic search algorithms

- Search algorithms are used to search offset values balancing CPU usage
- The objective function is the max CPU usage of a 2s-simulation of runnables
- Single-state search algorithms for discrete spaces (HC, Tabu)

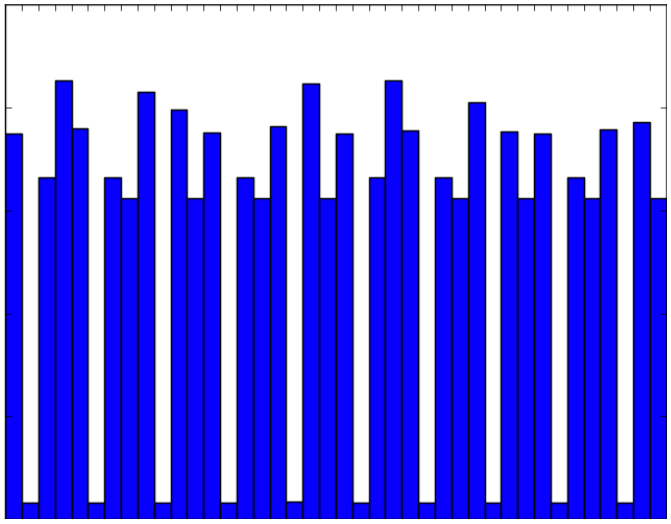
Case Study: an automotive software system with 430 runnables

5.34 ms



Running the system without offsets

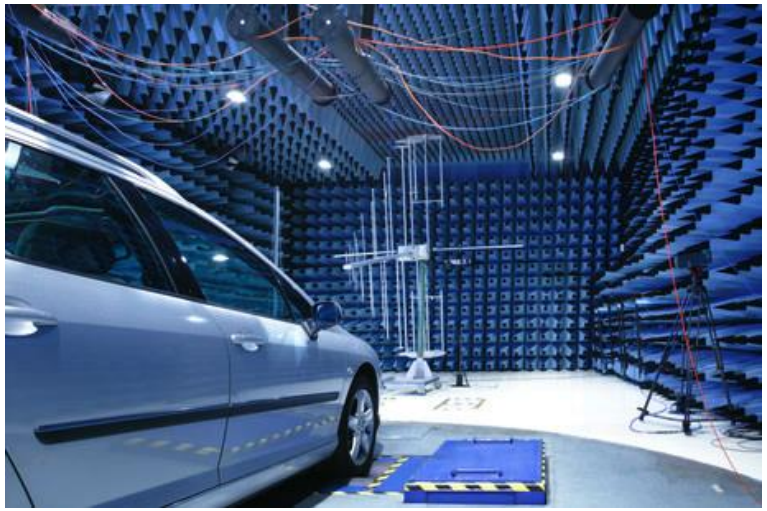
2.13 ms



Our optimized offset assignment

Conclusions

- We developed a number of search algorithms to compute offset values that reduce the max CPU time needed
- Our evaluation shows that our approach is able to generate reasonably good results for a large automotive system and in a small amount of time
- Due to large number of runnables and the orders of magnitude difference in runnables periods and their execution times, we were not able to use constraint solvers
- Current: Accounting for task time coupling constraints with multi-objective search → trade-off between relaxing coupling constraints and maximum CPU time



Questions

- What kinds of models need to be developed to support automated testing
- How expensive is test modeling?
- What are technologies enabling automated testing based on models?
- How cost-effective is model-based testing (MBT)?
- What are the limitations of MBT?
- What are the open issues regarding MBT on which research and innovation are still needed?

What kinds of models?

- What kinds of models need to be developed to support automated testing?
- Four aspects:
  - Notation
  - Modeling Methodology
    - Scope
    - Level of detail
- Factors:
  - Test objectives: Targeted faults, oracle.
  - Domain
  - Modeling skills and existing practice
- Standards: UML, SysML, MARTE, BPMN
  - Often need to be tailored or specialized

How expensive is modeling?

- From a few days to a few weeks, really depends on context
- Test models are much simpler than the systems they purport to model
- They can serve other purposes as well, e.g., specification, certification
- The real question: modeling cost versus test automation savings

What kinds of technologies?

- What are the enabler technologies for model-based testing?
- Goal: test case and oracle generation
  - Find input values, sequences of events/operations satisfying properties based on models, e.g., path in a state machine
  - Derive oracles to detect failures at run-time, e.g., state invariants, valid output sequences, metamorphic rules
  - Timing or other performance measures may be relevant
- Goals can be often re-expressed as an optimization or constraint solving problem
  - Constraint solvers, e.g., IBM CPLEX
  - Metaheuristic search, e.g., genetic algorithms
  - Main challenge: Scalability

Cost-Effective?

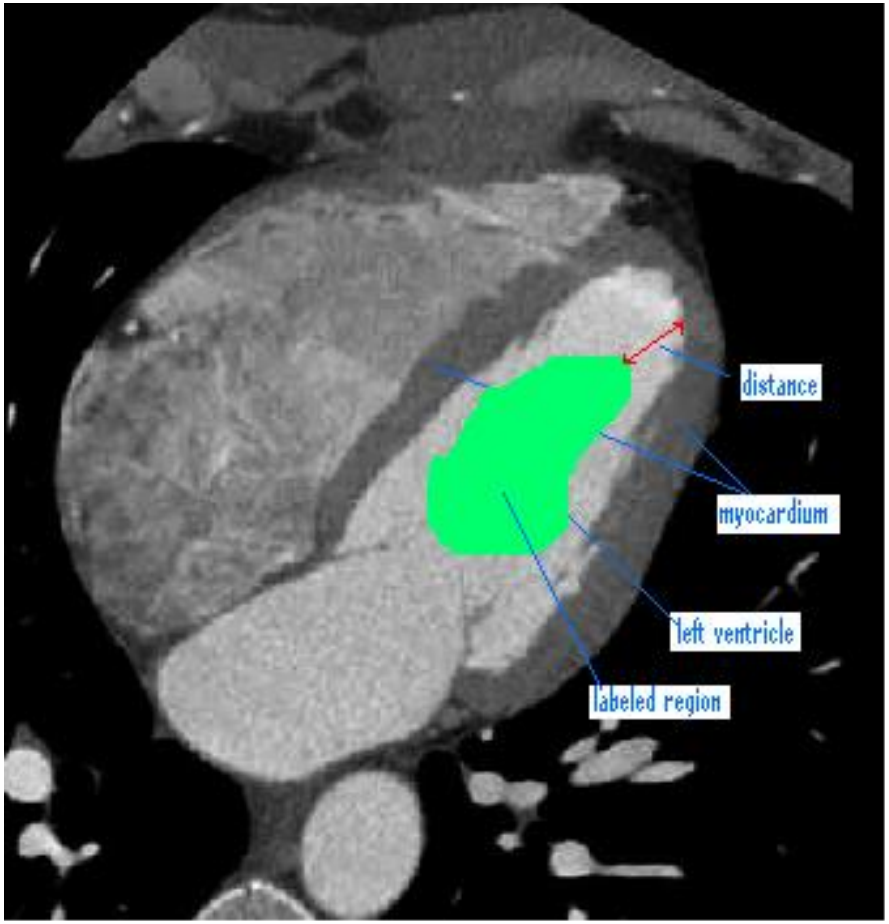
- How cost-effective is MBT?

- − Modeling training & Tools
- − Modeling overhead
- \+ Test automation scales up
- \+ Regeneration of test suites when changes
- \+ More systematic, more confidence
- \+ Traceability: impact analysis, regression

- Cost-benefit results vary according to these factors
- My experience:
  - The benefits far outweigh the costs, especially when accounting for changes (e.g., requirements)

MBT Limitations?

- Not applicable when the system or environment cannot be easily modeled with available notations and tools
- Example: No (precise) condition can be identified to automate oracles at run-time
- E.g., simulation, image segmentation, scientific computing



Open Issues?

- Scalability of test case generation
  - Quick constraint solving
- Tailoring modeling notations and methodologies to specific problems and domains
  - The number of combinations of problems and domains is large
  - Hence potential problems in tailoring commercial tools
- Empirical studies
  - There are very few credible, well-reported empirical studies
- Handling model changes
  - Impact analysis
  - Regression testing, e.g., selection, prioritization

FAQs

- What if the model is incomplete or incorrect?
  - The purpose of MBT is automation, not proof
  - The model may change as a result of failure
- Does MBT give me a proven test strategy?
  - No, but it enables you to define and automate one
  - There is not such thing as a universal, proven test strategy
- Can't I just buy and apply some commercial MBT tool?
  - How to apply MBT depends heavily on the domain, context, and test objectives
  - Heavy tailoring and investigation are required
- Isn't MBT too expensive to introduce and tailor to our needs?
  - manual testing (e.g., generation, oracle) is much more expensive and less effective

Conclusions

- Despite much hype, the hardest testing problems (test case generation, oracle) are not (really) solved yet in most contexts
- Modeling technology has matured (thanks in large part to OMG standardization efforts around the UML and MDA)
- It is now easier to support Model-based testing (MBT) and integrate it with other development activities
- MBT is a natural fit for companies using Model Driven Engineering (MDE), but is also suitable for those that are not
  - MBT is a good starting point for MDE
- In many situations, model-based testing is the only way to achieve full test automation (scalability) – the question is not whether to adopt MBT, but how.
- Much research and innovation is still required though and it must involve collaborations between research and industry

Selected References

- L. Briand, Y. Labiche, and M. Shousha, “Using genetic algorithms for early schedulability analysis and stress testing in real-time systems”, Genetic Programming and Evolvable Machines, vol. 7 no. 2, pp. 145-170, 2006
- M. Shousha, L. Briand, and Y. Labiche, “UML/MARTE Model Analysis Method for Uncovering Scenarios Leading to Starvation and Deadlocks in Concurrent Systems”, IEEE Transactions on Software Engineering 38(2), 2012.
- Z. Iqbal, A. Arcuri, L. Briand, “Empirical Investigation of Search Algorithms for Environment Model-Based Testing of Real-Time Embedded Software”, ACM ISSTA 2012
- S. Nejati, S. Di Alesio, M. Sabetzadeh, L. Briand, “Modeling and Analysis of CPU Usage in Safety-Critical Embedded Systems to Support Stress Testing”, ACM/IEEE MODELS 2012
- S. Nejati, Mehrdad Sabetzadeh, D. Falessi, L. C. Briand, T. Coq, “A SysML-based approach to traceability management and design slicing in support of safety certification: Framework, tool support, and case studies”, Information & Software Technology 54(6): 569-590 (2012)
- L. Briand et al., “Traceability and SysML Design Slices to Support Safety Inspections: A Controlled Experiment”, forthcoming in ACM Transactions on Software Engineering and Methodology, 2013

Selected References (cont.)

- Rajwinder Kaur Panesar-Walawege, Mehrdad Sabetzadeh, Lionel C. Briand: Supporting the verification of compliance to safety standards via model-driven engineering: Approach, tool-support and empirical validation. Information & Software Technology 55(5): 836-864 (2013)
- Razieh Behjati, Tao Yue, Lionel C. Briand, Bran Selic: SimPL: A product-line modeling methodology for families of integrated control systems. Information & Software Technology 55(3): 607-629 (2013)
- Hadi Hemmati, Andrea Arcuri, Lionel C. Briand: Achieving scalable model-based testing through test case diversity. ACM Trans. Softw. Eng. Methodol. 22(1): 6 (2013)
- Nina Elisabeth Holt, Richard Torkar, Lionel C. Briand, Kai Hansen: State-Based Testing: Industrial Evaluation of the Cost-Effectiveness of Round-Trip Path and Sneak-Path Strategies. ISSRE 2012: 321-330
- Razieh Behjati, Tao Yue, Lionel C. Briand: A Modeling Approach to Support the Similarity-Based Reuse of Configuration Data. MoDELS 2012: 497-513
- Shaukat Ali, Lionel C. Briand, Andrea Arcuri, Suneth Walawege: An Industrial Application of Robustness Testing Using Aspect-Oriented Modeling, UML/MARTE, and Search Algorithms. MoDELS 2011: 108-122