Making Model-Driven Verification Practical and Scalable: Experiences and Lessons Learned

Lionel Briand
IEEE Fellow, FNR PEARL Chair

Interdisciplinary Centre for ICT Security, Reliability, and Trust (SnT)
University of Luxembourg, Luxembourg

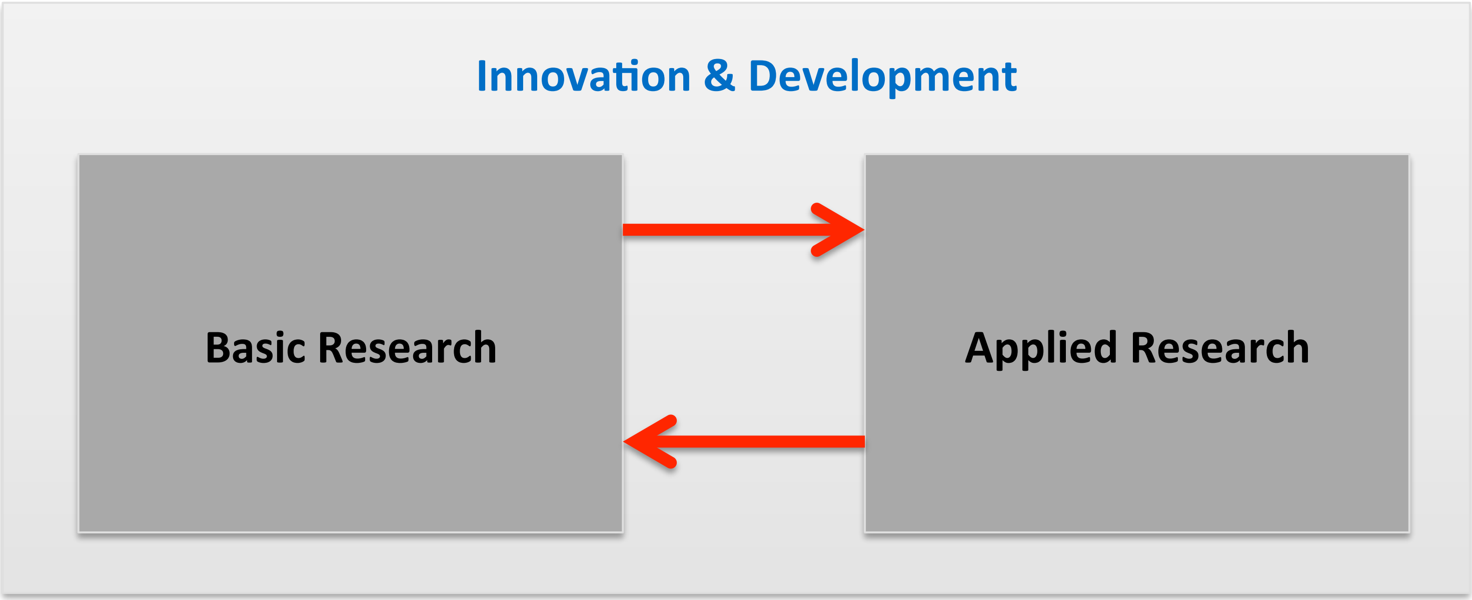
SAM, Valencia, 2014

SnT Software Verification and Validation Lab

- SnT centre, Est. 2009: Interdisciplinary, ICT security-reliability-trust
- 230 scientists and Ph.D. candidates, 20 industry partners
- SVV Lab: Established January 2012, www.svv.lu
- 25 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

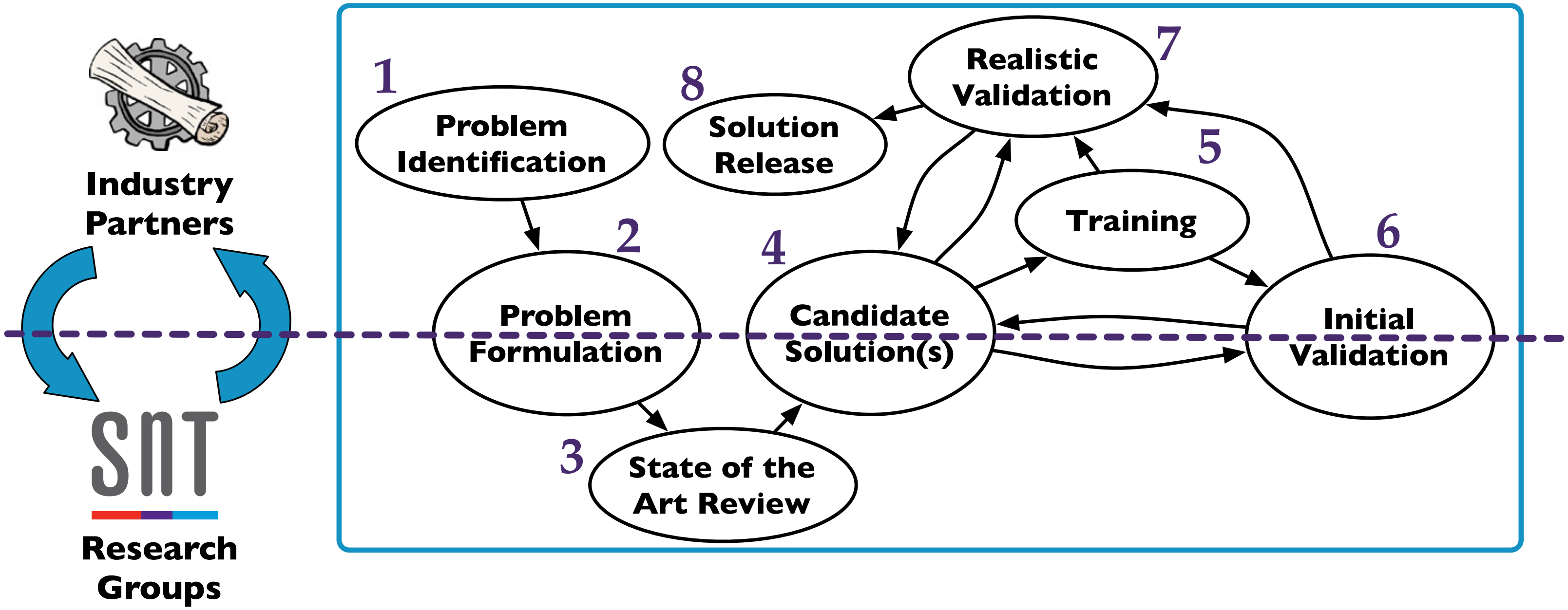
Schneiderman, 2013



- Basic and applied research take place in a rich context
- Basic Research is also driven by problems raised by applied research, which is itself fed by innovation and development
- Publishable research results and focused practical solutions that serve an existing market.

Collaboration in Practice

- Well-defined problems in context
- Realistic evaluation
- Long term industrial collaborations



Motivations

- The term “verification” is used in its wider sense: Defect detection and removal
- One important application of models is to drive and automate verification
- In practice, despite significant advances in model-based testing, this is not commonly part of practice
- Decades of research have not yet significantly and widely impacted practice

Applicability?
Scalability?

Definitions

- *Applicable:* Can a technology be efficiently and effectively applied by engineers in realistic conditions?
  - realistic ≠ universal
  - includes usability
- *Scalable:* Can a technology be applied on large artifacts (e.g., models, data sets, input spaces) and still provide useful support within reasonable effort, CPU and memory resources?

Outline

- Project examples, with industry collaborations
- Lessons learned regarding developing applicable and scalable solutions (our research paradigm)
- Meant to be an interactive talk – I am also here to learn

Some Past Projects (< 5 years)

Company	Domain	Objective	Notation	Automation
Cisco	Video conference	Robustness testing	UML profile	Search, model transformation
Kongsberg Maritime	Oil & Gas	CPU usage	UML+MARTE	Constraint Solving
WesternGeco	Marine seismic acquisition	Functional testing	UML profile + MARTE	Search, constraint solving
SES	Satellite	Functional and robustness testing, requirements QA	UML profile	Search, Model mutation, NLP
Delphi	Automotive systems	Testing safety +performance	Matlab/Simulink	Search, machine learning, statistics
CTIE	Legal & financial	Legal Requirements testing	UML Profile	Model transformation, constraint checking
HITEC	Crisis Support systems	Security, Access Control	UML Profile	Constraint verification, machine learning, Search
CTIE	eGovernment	Conformance testing	UML Profile, BPMN, OCL extension	Domain specific language, Constraint checking
IEE	Automotive, sensor systems	Functional and Robustness testing, traceaibility and certification	UML profile, Use Case Modeling extension, Matlab/Simulink	NLP, Constraint solving

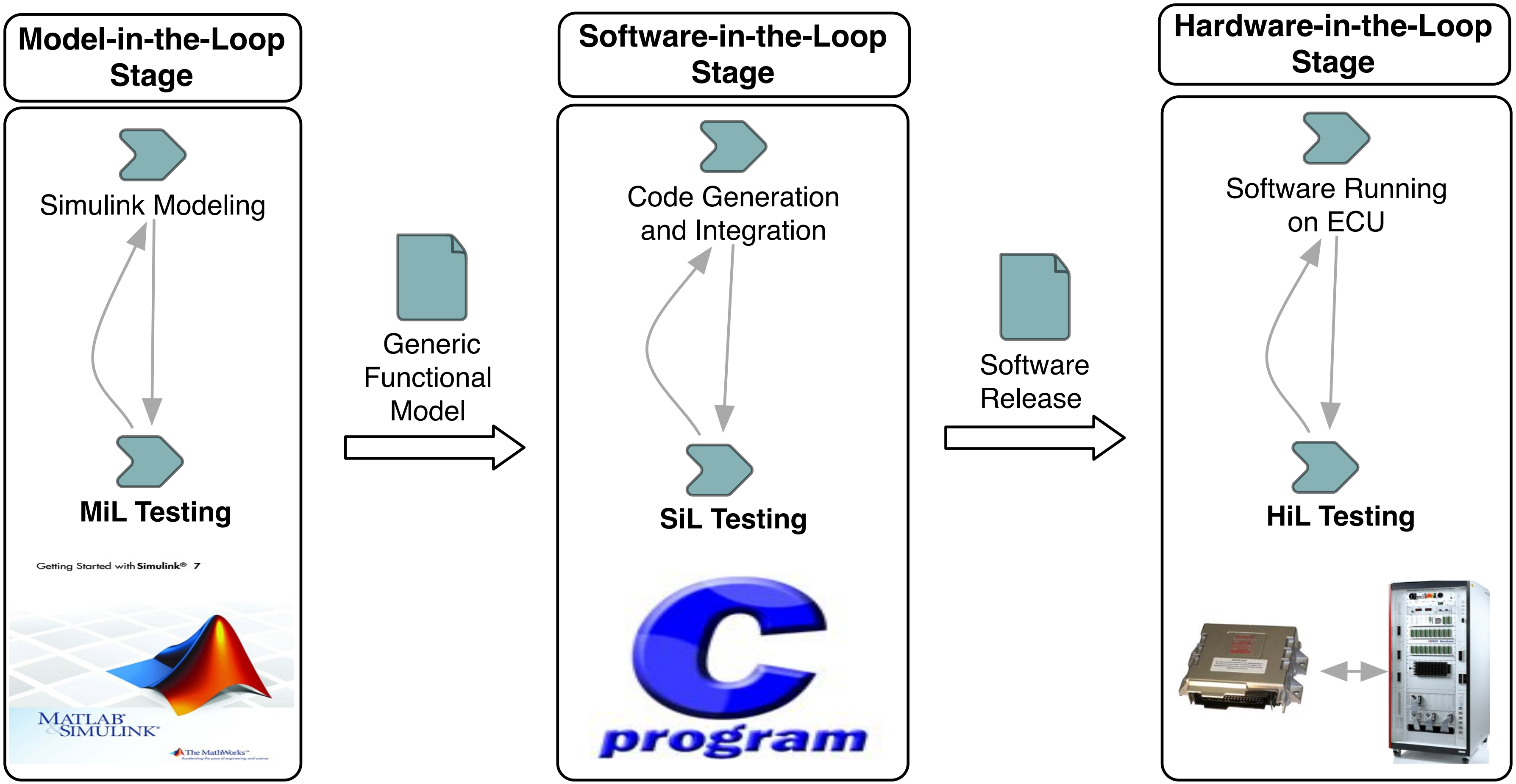
Testing Closed-Loop Controllers

References:

- *R. Matinnejad et al., "MiL Testing of Highly Configurable Continuous Controllers: Scalable Search Using Surrogate Models", IEEE/ACM ASE 2014*
- *R. Matinnejad et al., "Search-Based Automated Testing of Continuous Controllers: Framework, Tool Support, and Case Studies", Information and Software Technology (2014)*

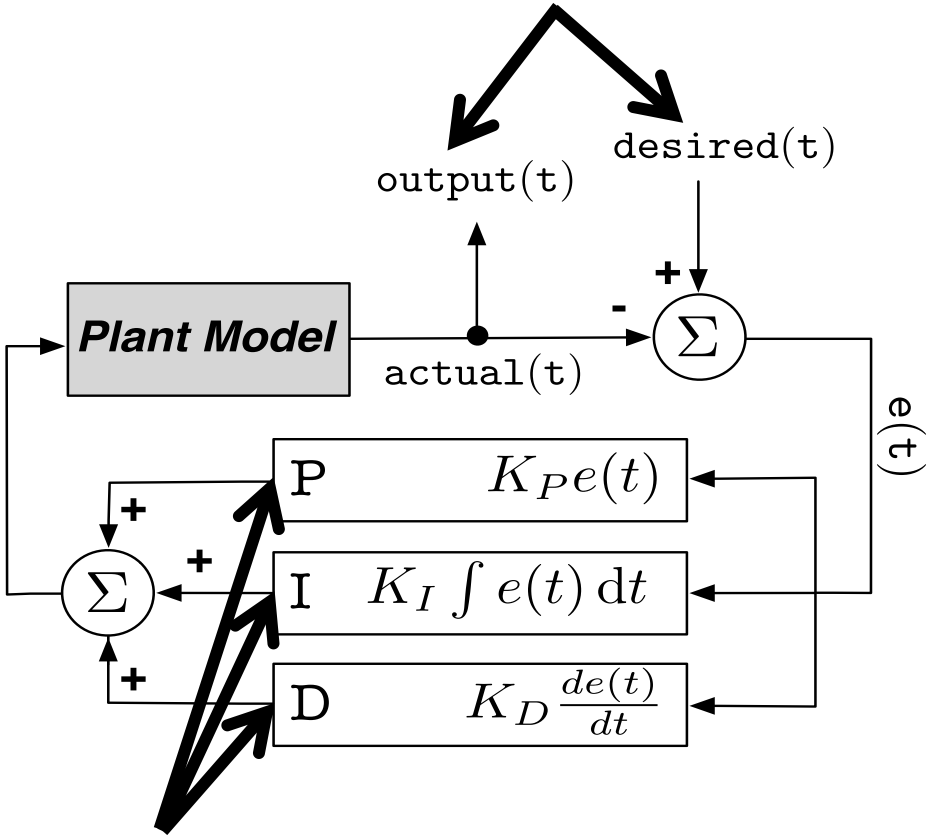
Dynamic continuous controllers are present in many embedded systems

Development Process (Delphi)



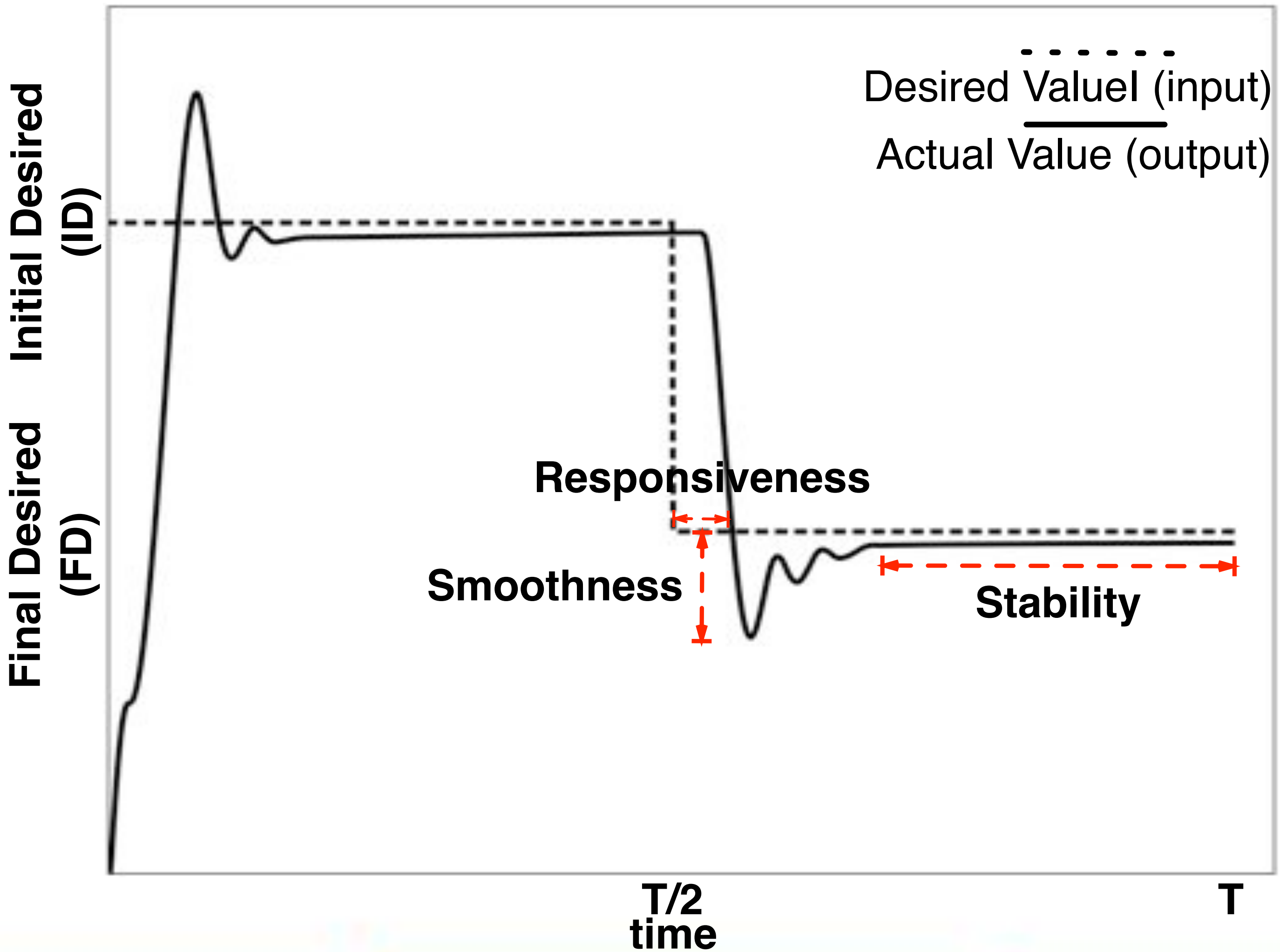
Controllers at MIL

Inputs: Time-dependent variables



Configuration Parameters

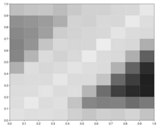
Inputs, Outputs, Test Objectives



Process and Technology

Objective Functions based on Requirements
\+
Controller-plant model

1. Exploration



HeatMap Diagram

Domain Expert

List of Critical Regions

2. Single-State Search

Worst-Case Scenarios

CoCoTest

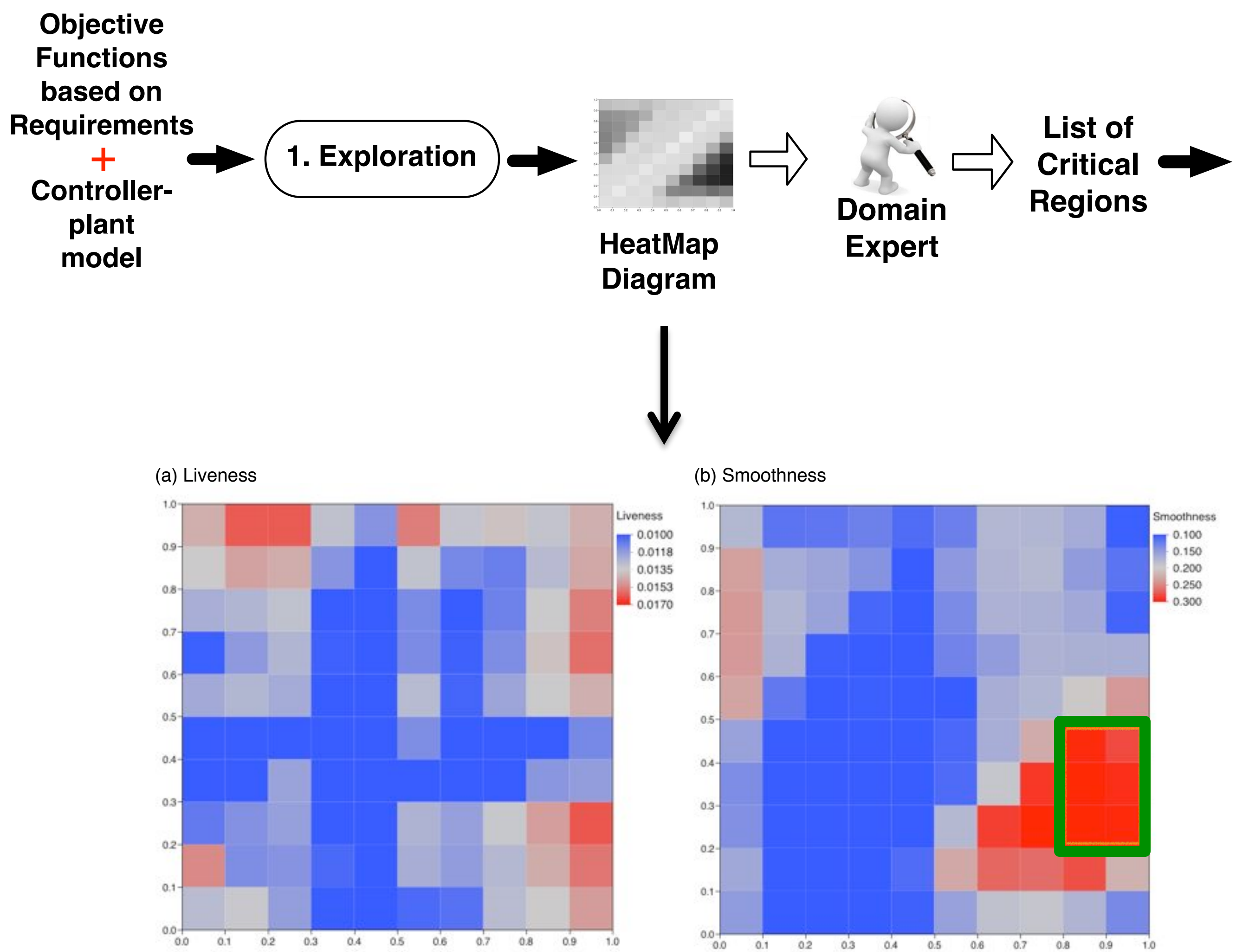
https://sites.g

Final Desired (FD)

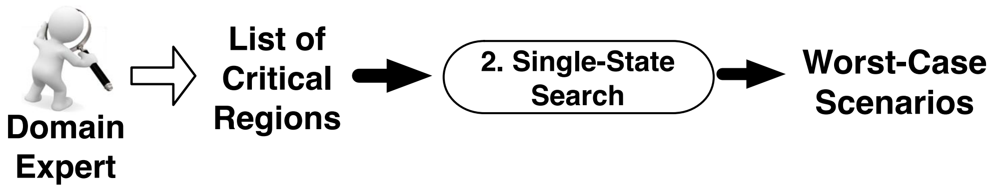
Worst Case(s)?

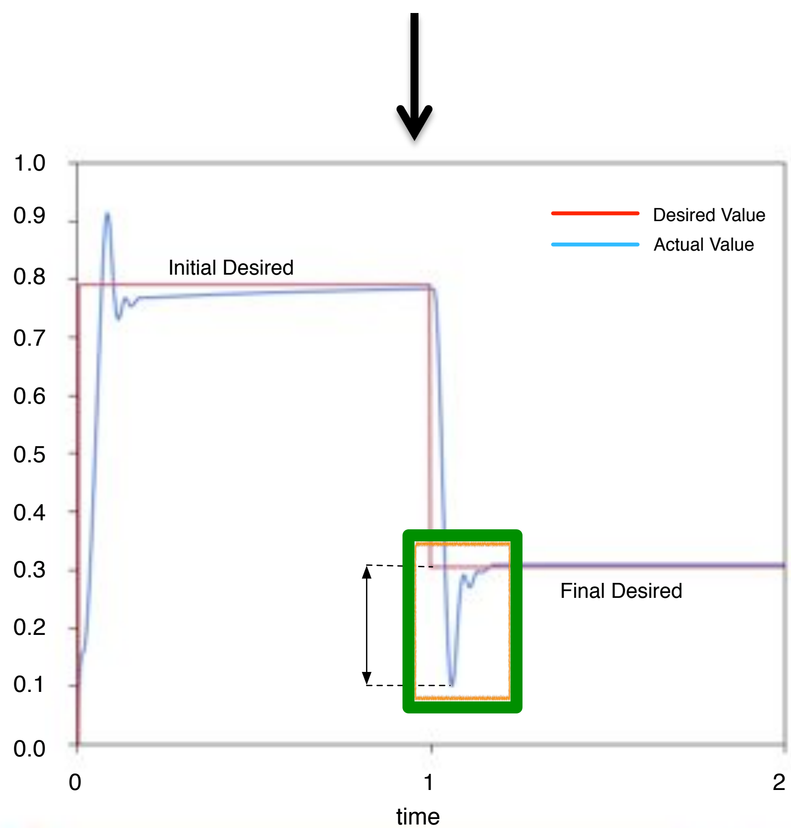
Initial Desired (ID)

Process and Technology (2)



Process and Technology (3)





Challenges, Solutions

- Achieving scalability with configuration parameters:
  - Simulink simulations are expensive
  - Sensitivity analysis to eliminate irrelevant parameters
  - Machine learning (Regression trees) to partition the space automatically and identify high-risk areas
  - Surrogate modeling (statistical and machine learning prediction) to predict properties and avoid simulation, when possible

Results

- Automotive controllers on Electronics Control Units

- Our approach enabled our partner to identify worst-case scenarios that were much worse than known and expected scenarios, entirely automatically

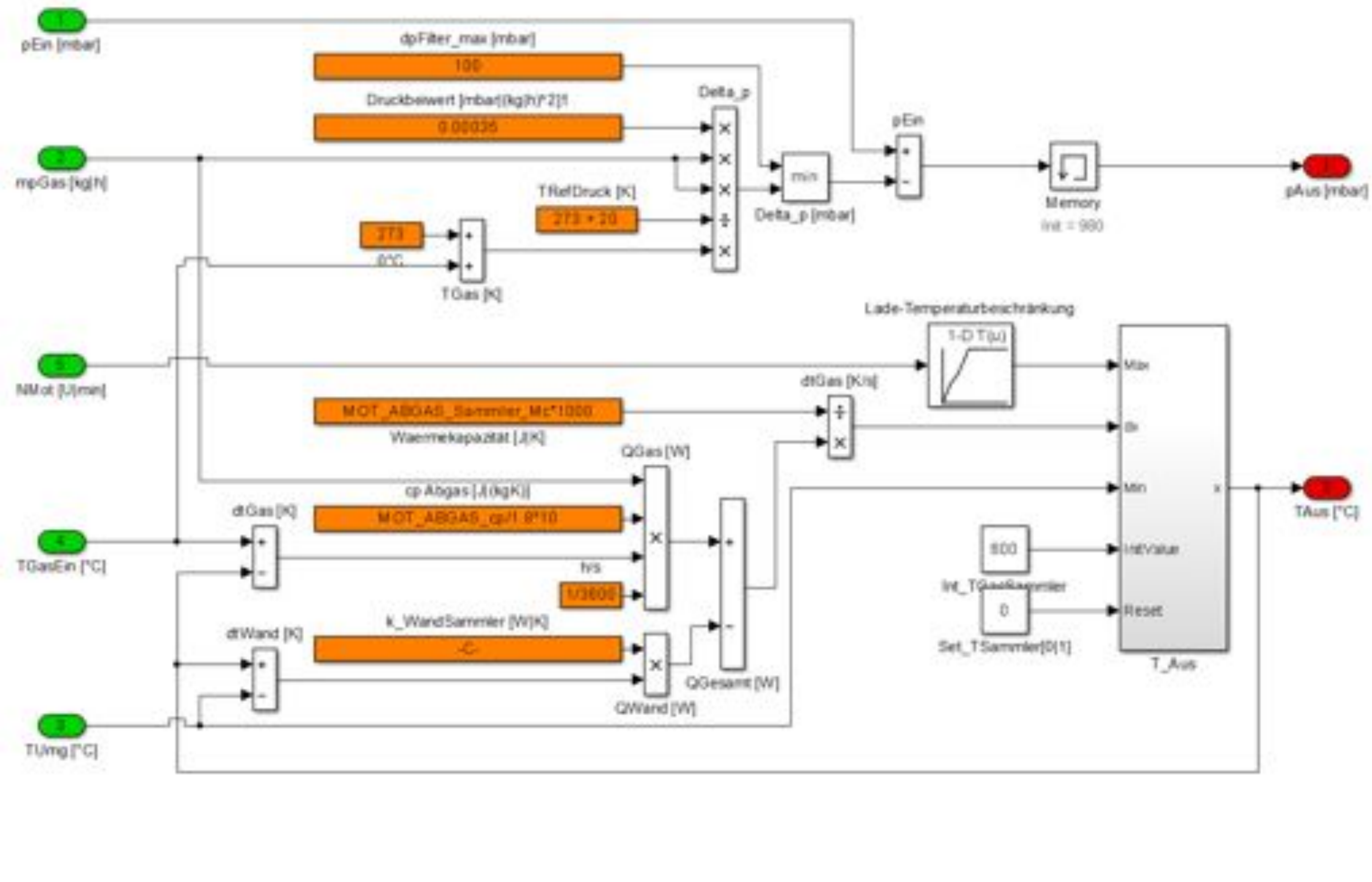
Fault Localisation in Simulink Models

Reference:

- *Bing Liu et al., "Kanvoo: Fault Localization in Simulink Models", submitted*

Context and Problem

- Simulink models

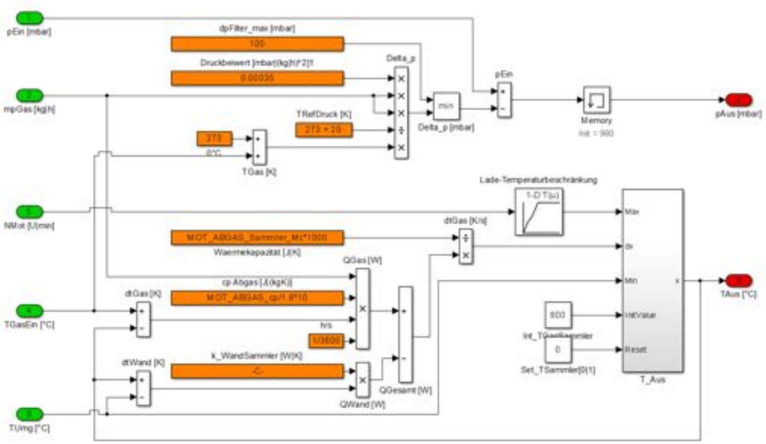


-

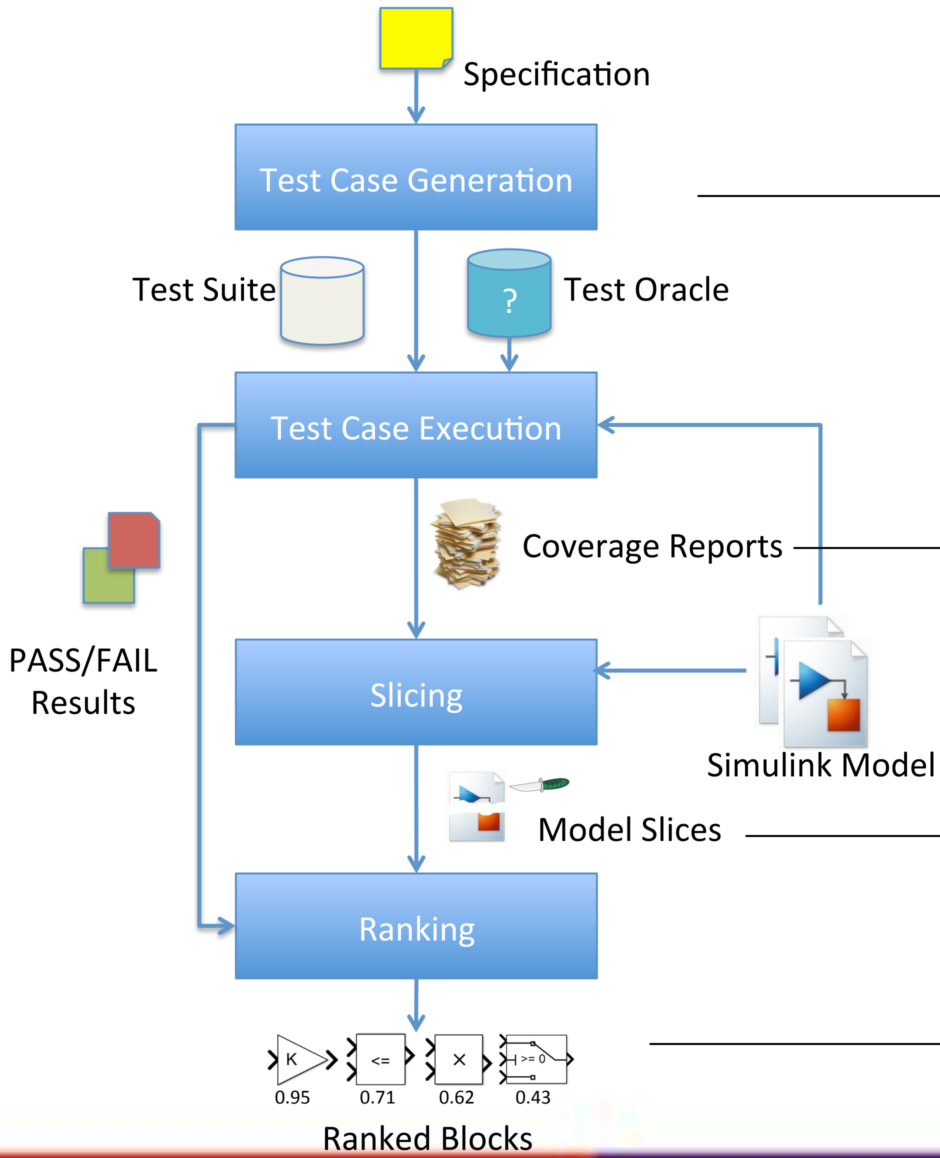
-

Context and Problem (2)

- Simulink models
  - are <u>complex</u>
    - hundreds of blocks and lines
    - many hierarchy levels
    - continuous functions
  - might be <u>faulty</u>
    - output signals do not match
    - wrong connection of lines
    - wrong operators in blocks
- Debugging Simulink models is
  - difficult
  - time-consuming
  - but yet crucial
- Automated techniques to support debugging?



Solution Overview



Any test strategy

Provided by Matlab tool

One slice for each test case and output

For each test case and output, or overall

Evaluation and Challenges

- Good accuracy overall: 5-6% blocks must be inspected on average to detect faults

- But less accurate predictions for certain faults: Low observability
- Possible Solution: Augment test oracle (observability)
  - Use subsystems outputs
  - Iterate at deeper levels of hierarchy
  - Tradeoff: cost of test oracle vs. debugging effort
  - 2.3% blocks on average

- 5-6%: still too many blocks for certain models
- Information requirements to help further filtering blocks?

Modeling and Verifying Legal Requirements

Reference:

- *G. Soltana et al., “ UML for Modeling Procedural Legal Rule”, IEEE/ACM MODELS 2014*
- *M. Adedjouma et al., “Automated Detection and Resolution of Legal Cross References”, RE 2014*

Context and Problem

- CTIE: Government computer centre in Luxembourg
- Large government (information) systems
- Implement legal requirements, must comply with the law
- The law usually leaves room for interpretation and changes on a regular basis, many cross-references
- Involves many stakeholders, IT specialists but also legal experts, etc.

Article Example

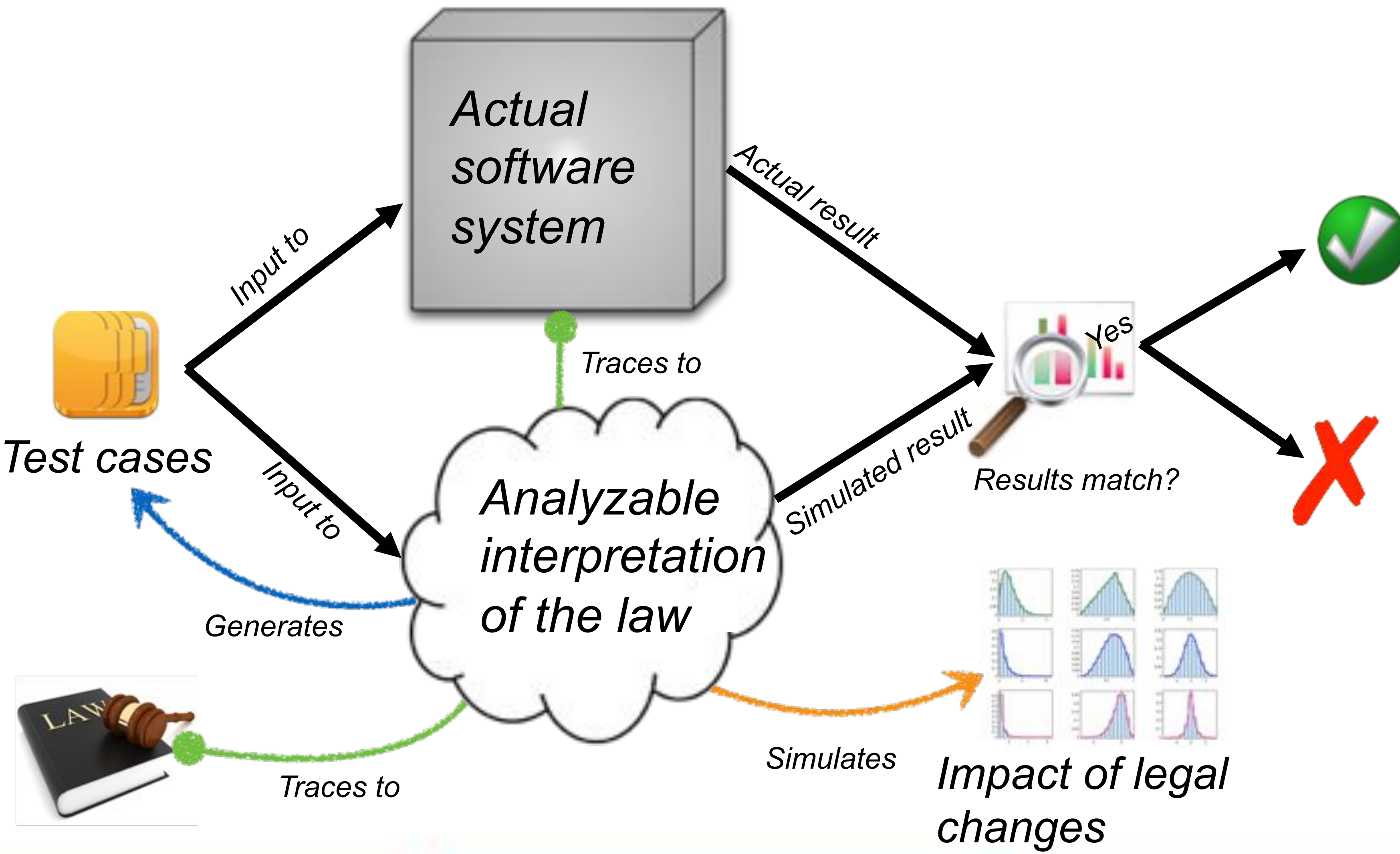
Art. 105bis *[…]The commuting expenses deduction (FD) is defined as a function over the distance between the principal town of the municipality on whose territory the taxpayer's home is located and the place of taxpayer's work. The distance is measured in units of distance expressing the kilometric distance between [principal] towns. A ministerial regulation provides these distances.*

The amount of the deduction is calculated as follows:
If the distance exceeds 4 units but is less than 30 units, the deduction is € 99 per unit of distance.
The first 4 units does not trigger any deduction and the deduction for a distance exceeding 30 units is limited to € 2,574.

Project Objectives

Objective	**Benefits**
Specification of legal requirements • including rationale and traceability to the text of law	• Make interpretation of the law explicit • Improve communication • Prerequisite for automation
Checking consistency of legal requirements	• Prevent errors in the interpretation of the law to propagate
Automated test strategies for checking system compliance to legal requirements	• Provide effective and scalable ways to verify compliance
Run-time verification mechanisms to check compliance with legal requirements	
Analyzing the impact of changes in the law	• Decrease costs and risks associated with change • Make change more predictable

Solution Overview



Research Steps

1. Conduct grounded theory study

- What information content should we expect?
- What are the complexity factors?

2. Build UML profile

- Explicit means for capturing information requirements
- Basis for modeling methodology
- Target: Legal experts and IT specialists

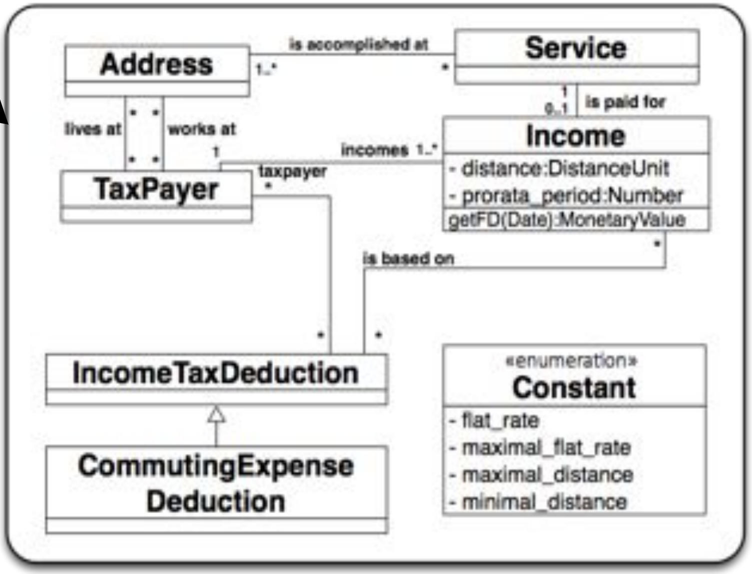
3. Model Transformation to enable V&V

- Target existing automation techniques
- Solvers for testing
- MATLAB for simulation

Example

***Art. 105bis** […]**The commuting expenses deduction (FD)** is defined as a function over the distance between the principal town of the municipality on whose territory the taxpayer's home is located and the place of taxpayer's work. The **distance** is measured in units of distance expressing the kilometric distance between [principal] towns. A ministerial regulation provides these distances.*

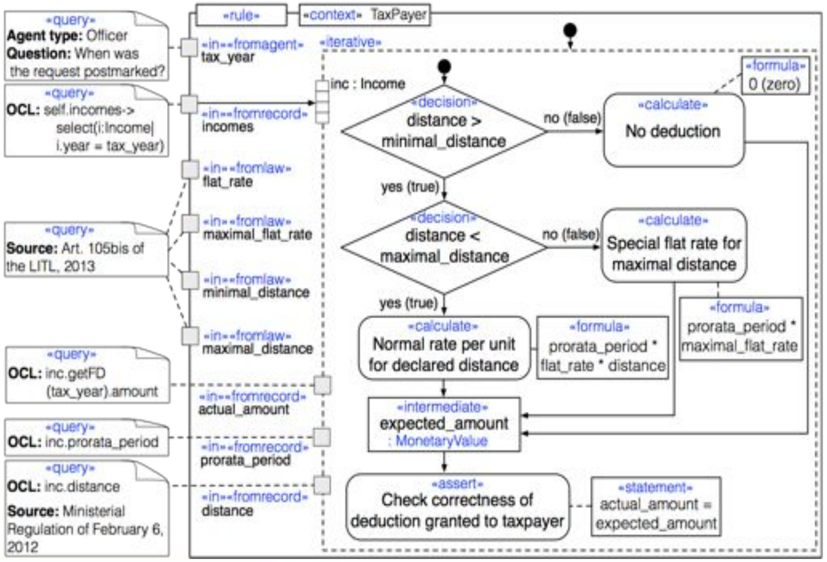
Interpretation + Traces



Example

The amount of the deduction is calculated as follows:
*If the distance exceeds **4** units but is less than **30 units**, the deduction is **€ 99** per unit of distance.*
*The first **4 units** does not trigger **any deduction** and the deduction for a distance **exceeding 30** units is limited to **€ 2,574**.*

Interpretation + Traces



Challenges and Results

- Profile must lead to models that are:
  - understandable by both IT specialists and legal experts
  - precise enough to enable model transformation and support our objectives
  - tutorials, many modeling sessions with legal experts

- In theory, though such legal requirements can be captured by OCL constraints alone, this is not applicable

- That is why we resorted to customized activity modeling, carefully combined with a simple subset of OCL

- Many traces to law articles, dependencies among articles: automated detection (NLP) of cross-references

Run-Time Verification of Business Processes

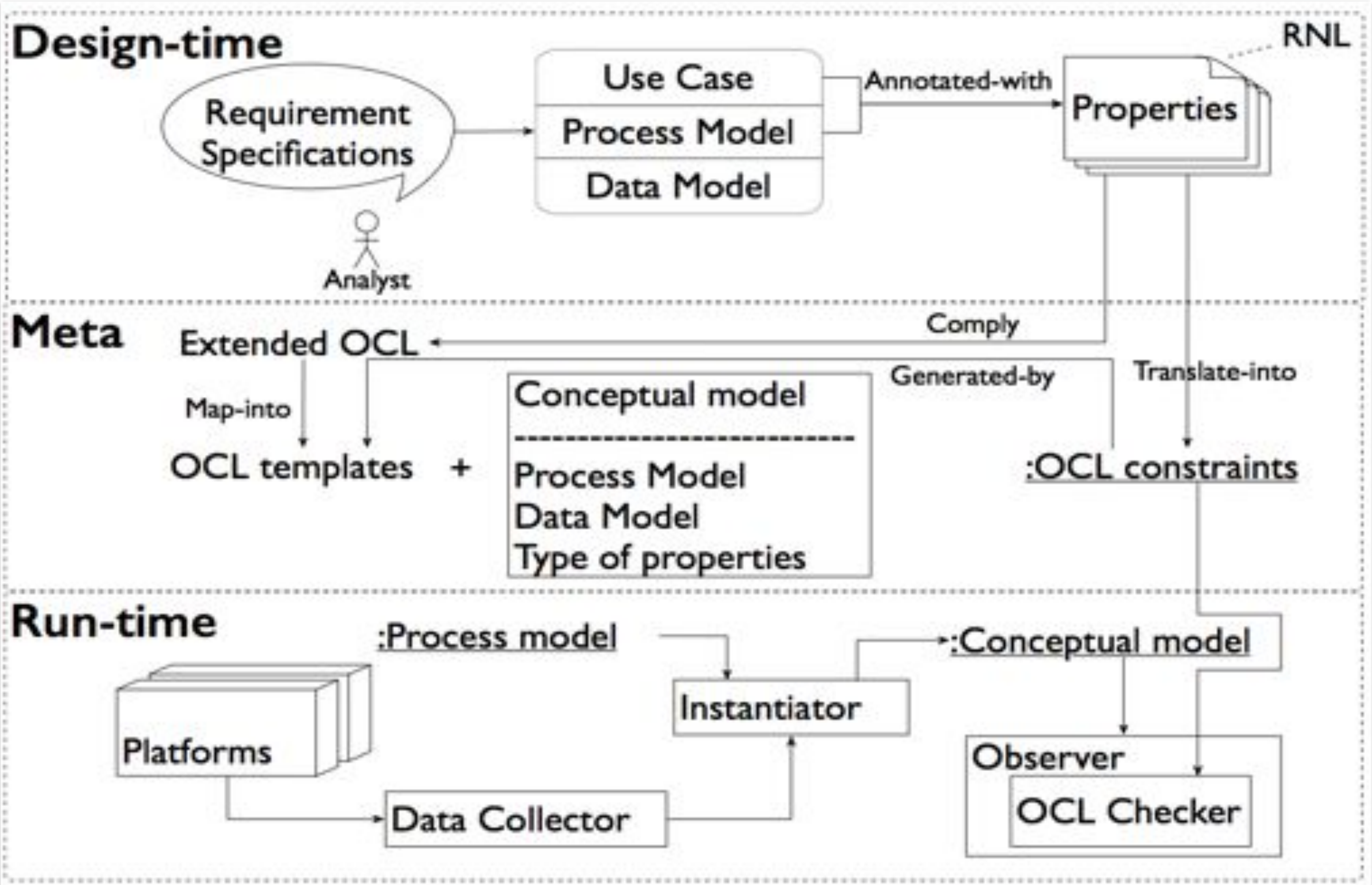
References:

- *W. Dou et al., "OCLR: a More Expressive, Pattern-based Temporal Extension of OCL", ECMFA 2014*
- *W. Dou et al., "Revisiting Model-Driven Engineering for Run-Time Verification of Business Processes", IEEE/ACM SAM 2014*
- *W. Dou et al., "A Model-Driven Approach to Offline Trace Checking of Temporal Properties with OCL", submitted*

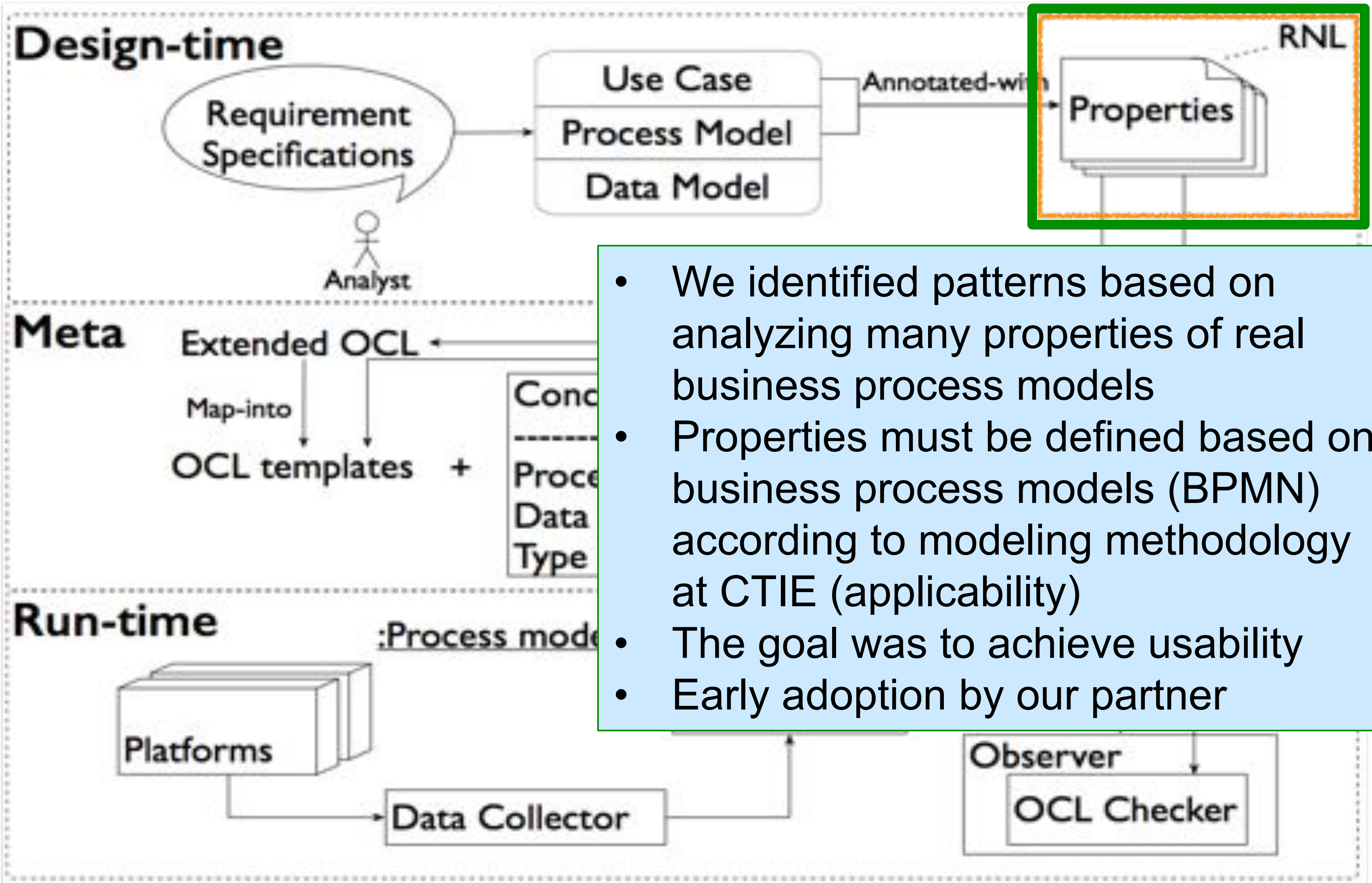
Context and Problem

- CTIE: Government Computing Centre of Luxembourg
- E-government systems mostly implemented as business processes
- CTIE models these business processes
- Business models have temporal properties that must be checked
  - Temporal logics not applicable
  - Limited tool support (scalability)
- Goal: Efficient, scalable, and practical off-line and run-time verification

Solution Overview

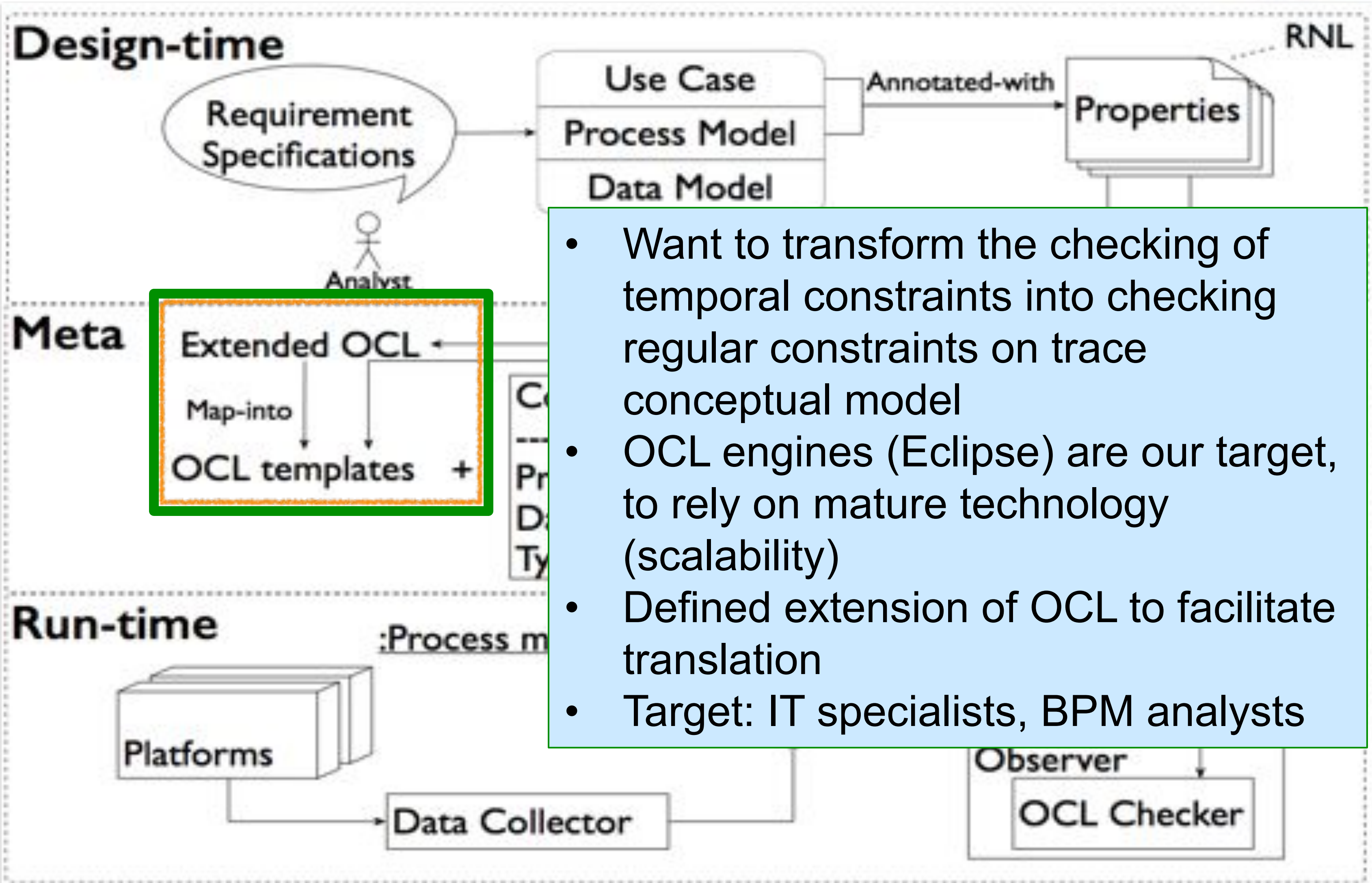


Solution Overview



- We identified patterns based on analyzing many properties of real business process models
- Properties must be defined based on business process models (BPMN) according to modeling methodology at CTIE (applicability)
- The goal was to achieve usability
- Early adoption by our partner

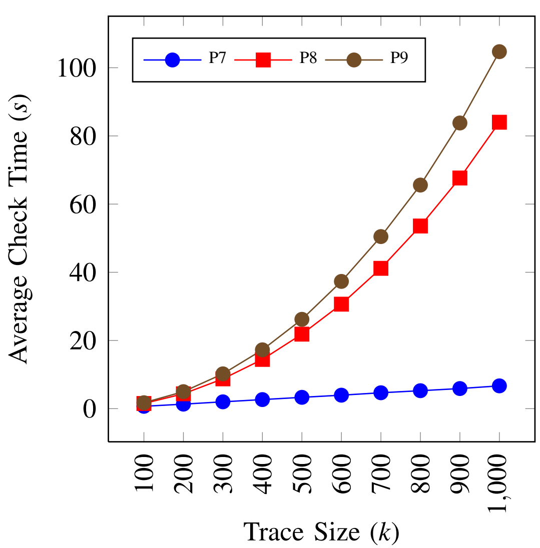
Solution Overview



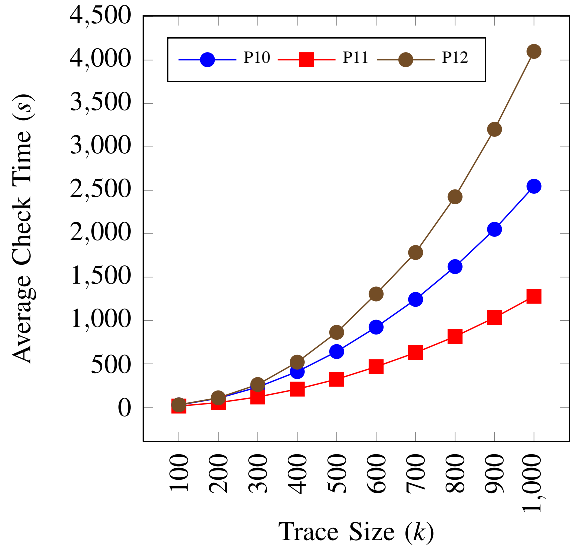
- Want to transform the checking of temporal constraints into checking regular constraints on trace conceptual model
- OCL engines (Eclipse) are our target, to rely on mature technology (scalability)
- Defined extension of OCL to facilitate translation
- Target: IT specialists, BPM analysts

Scalability Analysis

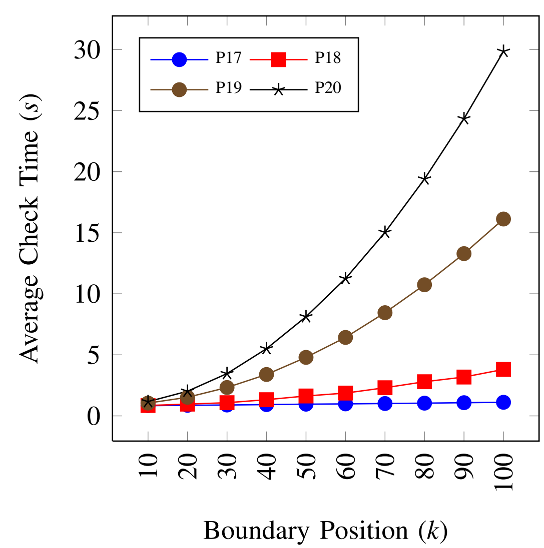
- Analyzed 47 properties in Identity Card Management System
- *“Once a card request is approved, the applicant is notified within three days; this notification has to occur before the production of the card is started.”*
- Scalability: Check time as a function of trace size …



(a) *globally* / P7–P9



(b) *globally* / P10–P12



(c) *before* / P17–P20

Schedulability Analysis and Stress Testing

References:

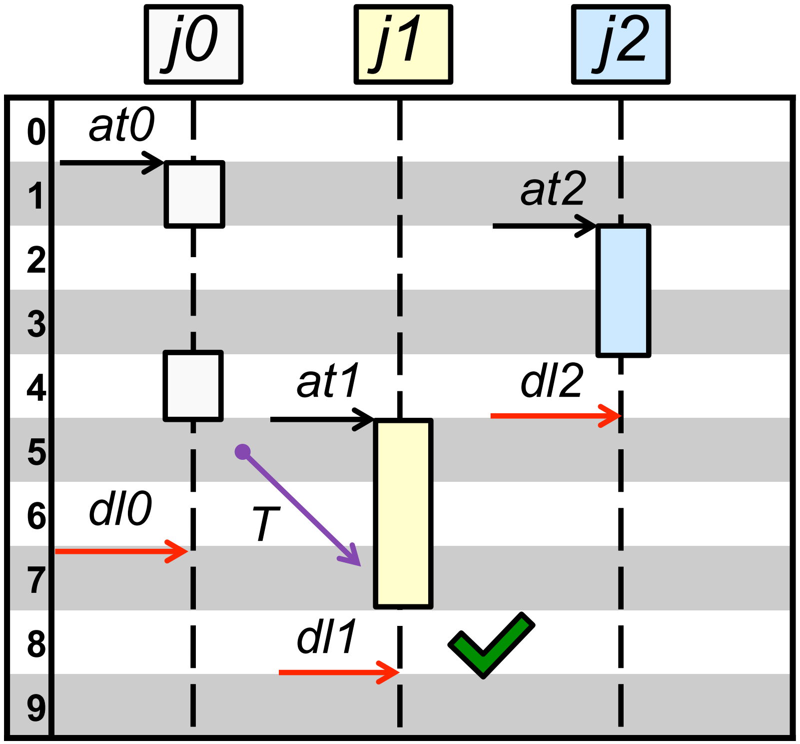
- *S. Nejati, S. Di Alesio, M. Sabetzadeh, and L. Briand, "Modeling and analysis of cpu usage in safety-critical embedded systems to support stress testing," in IEEE/ACM MODELS 2012.*
- *S. Di Alesio, S. Nejati, L. Briand. A. Gotlieb, "Stress Testing of Task Deadlines: A Constraint Programming Approach", ISSRE 2013, San Jose, USA*
- *S. Di Alesio, S. Nejati, L. Briand. A. Gotlieb, "Worst-Case Scheduling of Software Tasks – A Constraint Optimization Model to Support Performance Testing, Constraint Programming (CP), 2014*

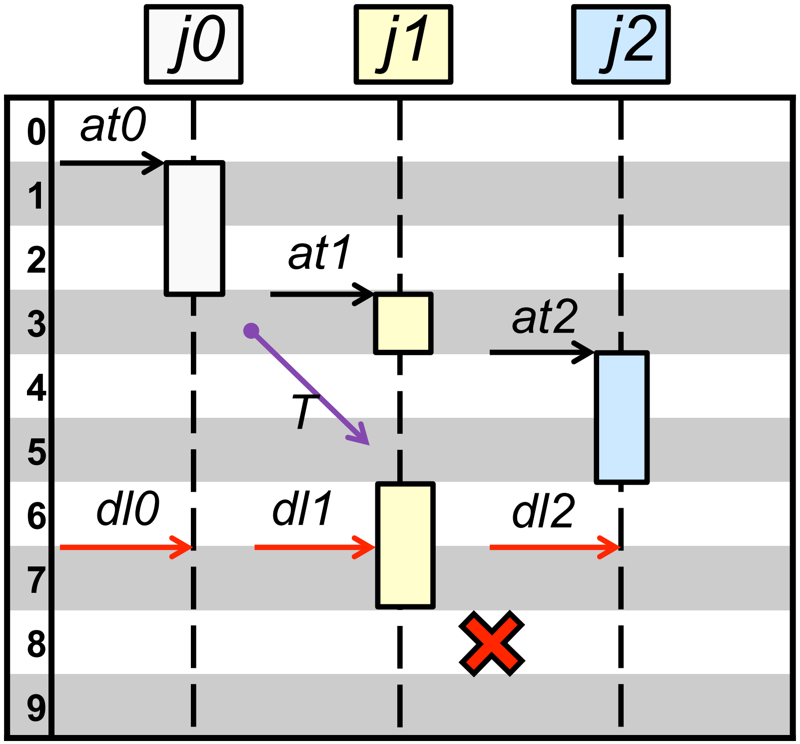
Problem

- Real-time, concurrent systems (RTCS) have concurrent interdependent tasks which have to finish before their deadlines
- Some task properties depend on the environment, some are design choices
- Tasks can trigger other tasks, and can share computational resources with other tasks
- Schedulability analysis encompasses techniques that try to predict whether all (critical) tasks are schedulable, i.e., meet their deadlines
- Stress testing runs carefully selected test cases that have a high probability of leading to deadline misses
- Testing in RTCS is typically expensive, e.g., hardware in the loop

Arrival Times Determine Deadline Misses

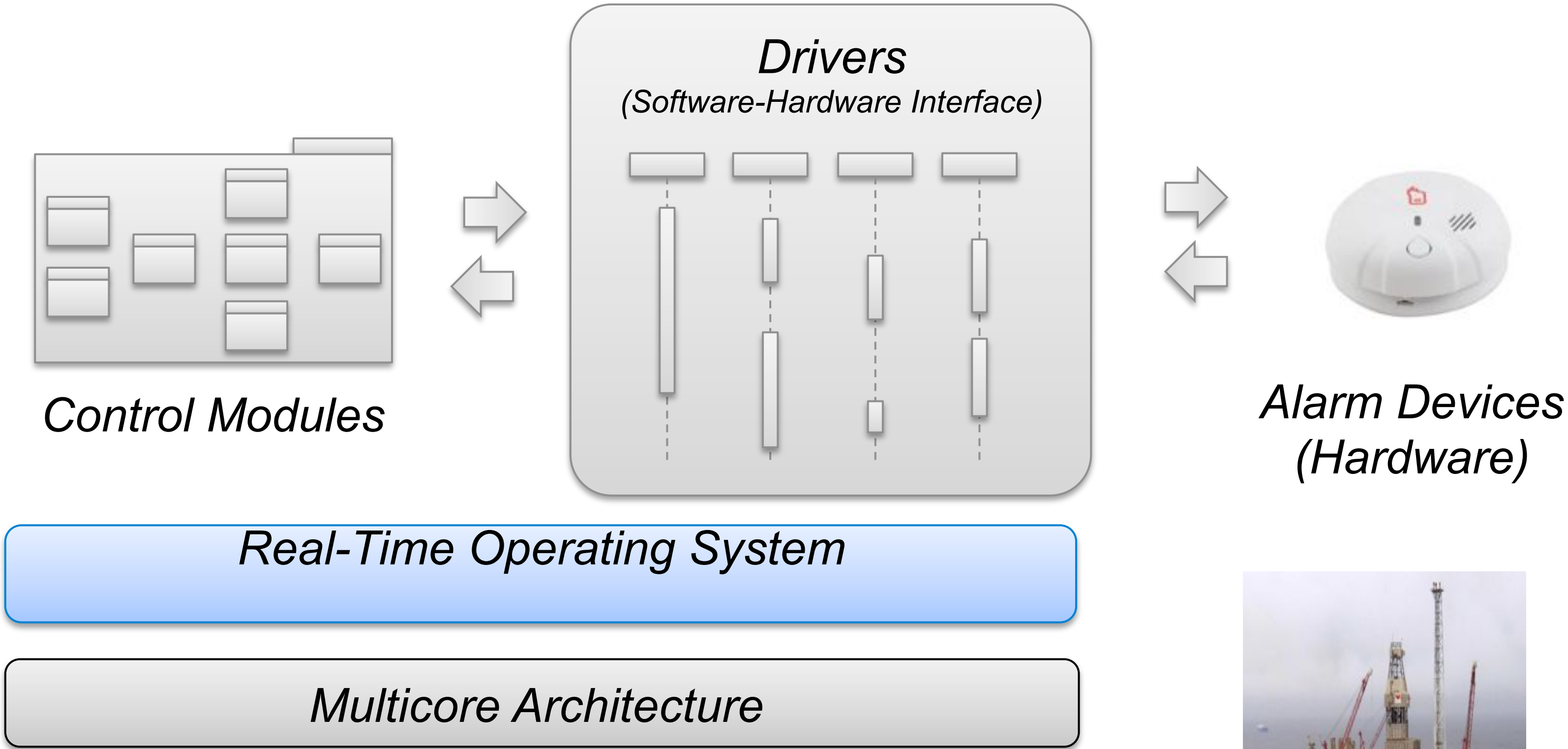
j0, j1 , j2 arrive at at0 , at1 , at2 and must finish before dl0 , dl1 , dl2





J1 can miss its deadline dl1 depending on when at2 occurs!

Context



Monitor gas leaks and fire in oil extraction platforms

Challenges and Solutions

- Ranges for arrival times form a very large input space
- Task interdependencies and properties constrain what parts of the space are feasible
- We re-expressed the problem as a constraint optimisation problem
- Constraint programming

Constraint Optimization

Constraint Optimization Problem

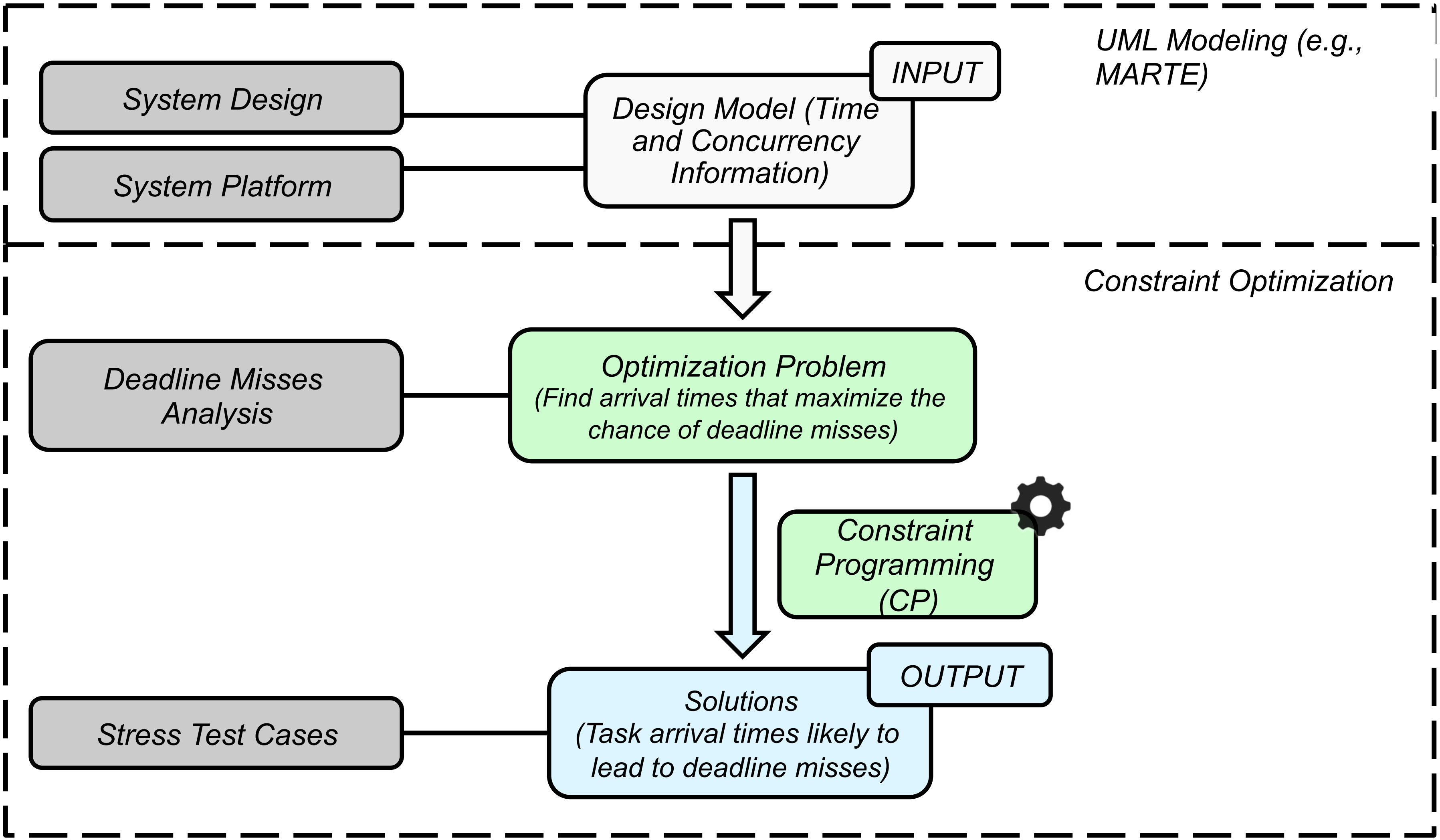
Static Properties of Tasks
(Constants)

Dynamic Properties of Tasks
(Variables)

OS Scheduler Behaviour
(Constraints)

Performance Requirement
(Objective Function)

Process and Technologies

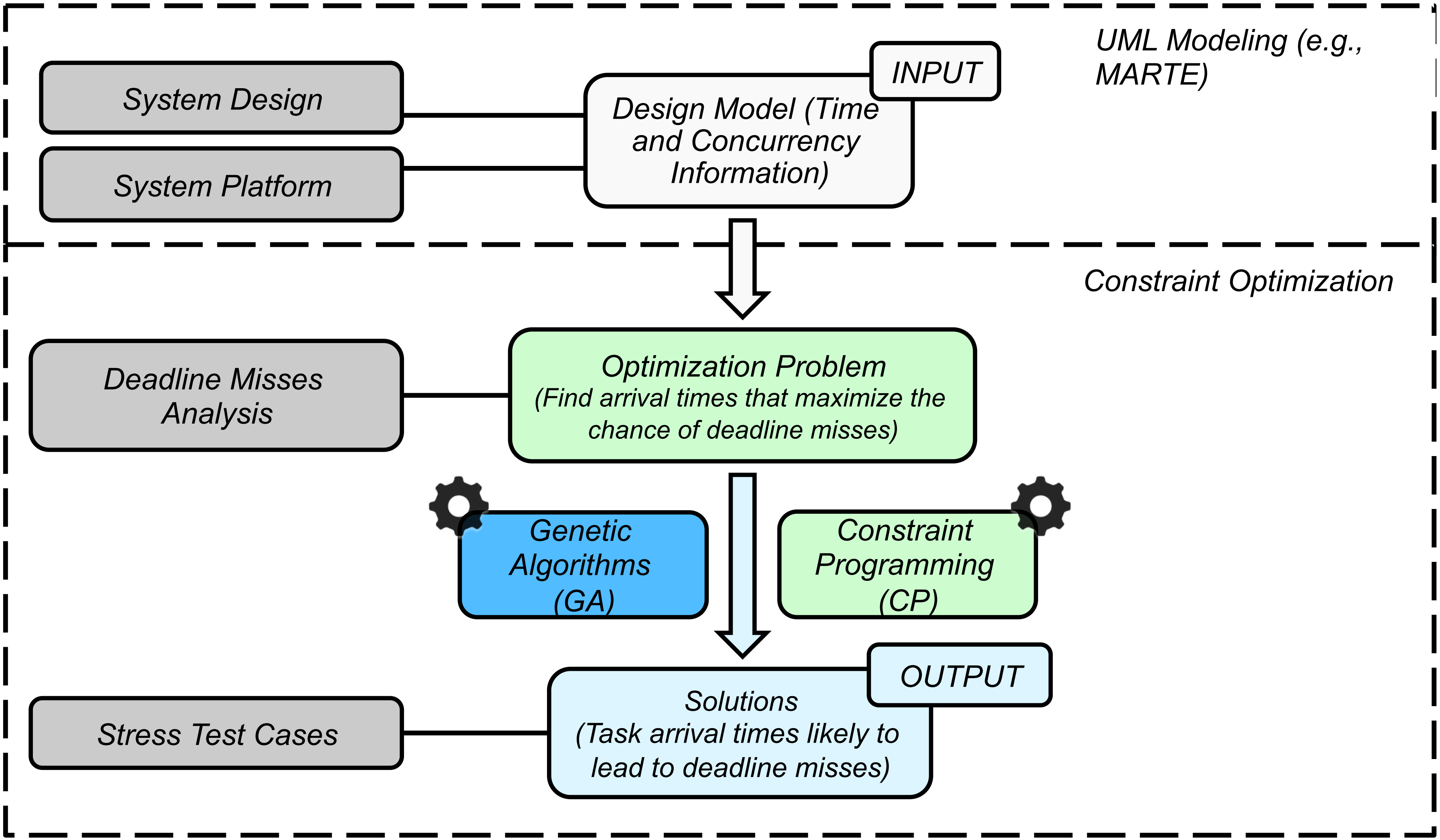


Challenges and Solutions (2)

- Scalability problem: Constraint programming (e.g., IBM CPLEX) cannot handle such large input spaces (CPU, memory)

- Solution: Combine metaheuristic search and constraint programming
  - metaheuristic search identifies high risk regions in the input space
  - constraint programming finds provably worst-case schedules within these (limited) regions

Process and Technologies



Applicable? Scalable?

Scalability examples

- This is the most common challenge in practice
- Testing closed-loop controllers
  - Large input and configuration space
  - Smart search optimization heuristics (machine learning)
- Fault localization
  - Large number of blocks and lines in Simulink models
  - Even a small percentage of blocks to inspect can be impractical
  - Additional information to support decision making? Incremental fault localisation?
- Schedulability analysis and stress testing
  - Constraint programming cannot scale by itself
  - Must be carefully combined with genetic algorithms

Scalability examples (2)

- Verifying legal requirements
  - Traceability to the law is complex
  - Many provisions and articles
  - Many dependencies within the law
  - Natural Language Processing: Cross references, support for identifying missing modeling concepts
- Run-time Verification of Business Processes
  - Traces can be large and properties complex to verify
  - Transformation of temporal properties into regular OCL properties, defined on a trace conceptual model
  - Incremental verification at regular time intervals
  - Heuristics to identify subtraces to verify

Scalability: Lessons Learned

- Scalability must be part of the problem definition and solution from the start, not a refinement or an after-thought
- It often involves heuristics, e.g., meta-heuristic search, NLP, machine learning, statistics
- Scalability often leads to solutions that offer “best answers” within time constraints, not guarantees
- Solutions to scalability are multi-disciplinary
- Scalability analysis should be a component of every research project – otherwise it is unlikely to be adopted in practice
- How many papers in MODELS or SAM do include even a minimal form of scalability analysis?

Applicability

- Definition?
- Usability: Can the target user population efficiently apply it?
- Assumptions: Are working assumptions realistic, e.g., realistic information requirements?
- Integration into the development process, e.g., are required inputs available in the right form and level of precision?

Applicability examples

- Testing closed-loop controllers
  - Working assumption: availability of sufficiently precise plant (environment) models
  - Means to visualize relevant properties in the search space (inputs, configuration), to get an overview and focus search on high-risk areas
- Schedulability analysis and stress testing
  - Availability of tasks architecture models
  - Precise WCET analysis
  - Applicability requires to assess risk based on near-deadline misses

Applicability examples (2)

- Fault localization:
  - Trade-off between # of model outputs considered versus cost of test oracles
  - Better understanding of the mental process and information requirements for fault localization
- Run-time verification of business process models
  - Temporal logic not usable by analysts
  - Language closer to natural language, directly tied to business process model
  - Easy transition to industry strength constraint checker
- Verifying legal requirements
  - Modeling notation must be shared by IT specialists and legal experts
  - One common representation for many applications, with traces to the law to handle changes
  - Multiple model transformation targets

Applicability: Lessons Learned

- Make working assumptions explicit: Determine the context of applicability
- Make sure those working assumptions are at least realistic in some industrial domain and context
- Assumptions don't need to be universally true – they rarely are anyway
- Run usability studies – do it for real!

Conclusions

- In most research endeavors, applicability and scalability are an after-thought, a secondary consideration, when at all considered
- Implicit assumptions are often made, often unrealistic in any context
- Problem definition in a vacuum
- Not adapted to research in an engineering discipline
- Leads to limited impact
- Research in model-based V&V is necessarily multi-disciplinary
- User studies are required and far too rare
- In engineering research, there is no substitute to reality

Acknowledgements

PhD. Students:

- Marwa Shousha
- Reza Matinnejad
- Stefano Di Alesio
- Wei Dou
- Ghanem Soltana
- Bing Liu

Scientists:

- Shiva Nejati
- Mehrdad Sabetzadeh
- Domenico Bianculli
- Arnaud Gotlieb
- Yvan Labiche

Making Model-Driven Verification Practical and Scalable: Experiences and Lessons Learned

Lionel Briand
IEEE Fellow, FNR PEARL Chair

Interdisciplinary Centre for ICT Security, Reliability, and Trust (SnT)
University of Luxembourg, Luxembourg

SAM, Valencia, 2014

SVV lab: svv.lu
SnT: www.securityandtrust.lu