SV.lu
software verification & validation

SnT
securityandtrust.lu

Testing of Cyber-Physical Systems:

Diversity-driven Strategies

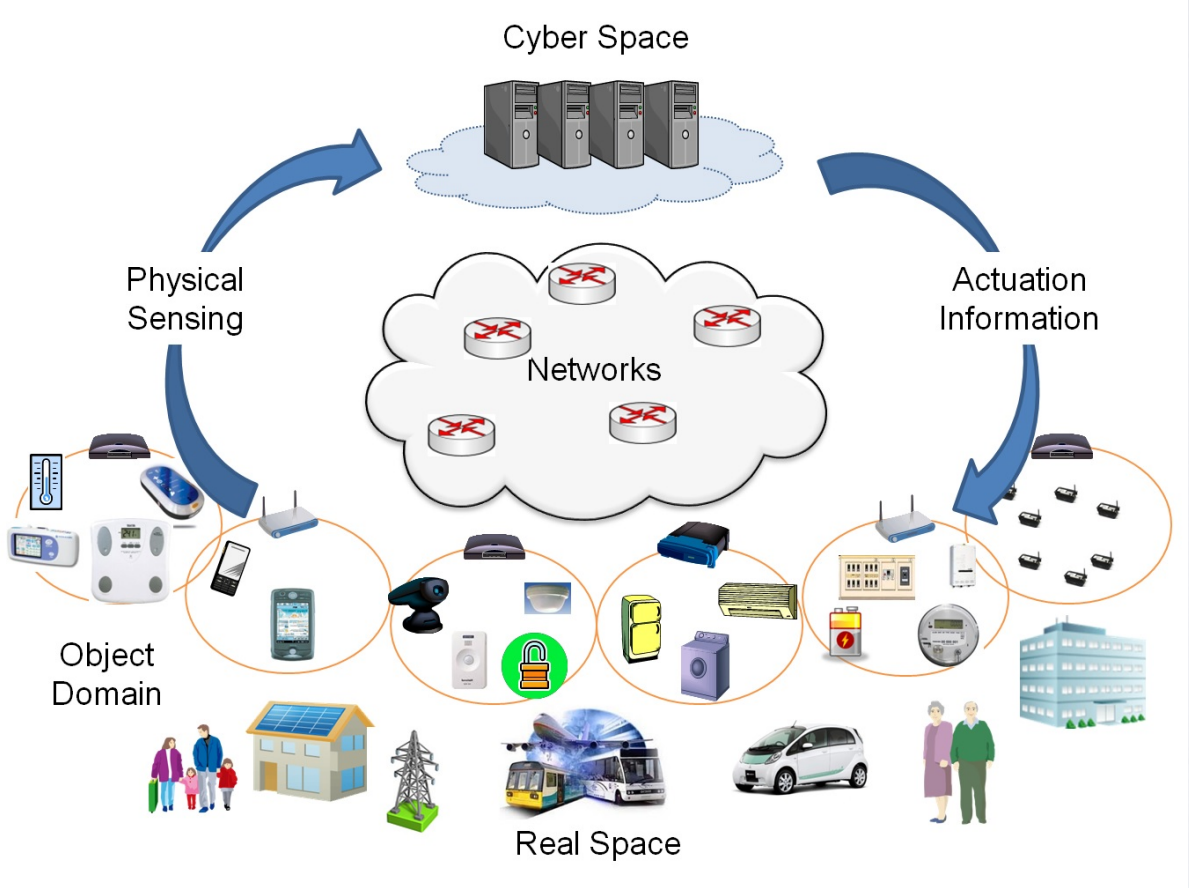
Lionel Briand

COW 57, London, UK

uni.lu
UNIVERSITÉ DU LUXEMBOURG

Cyber-Physical Systems

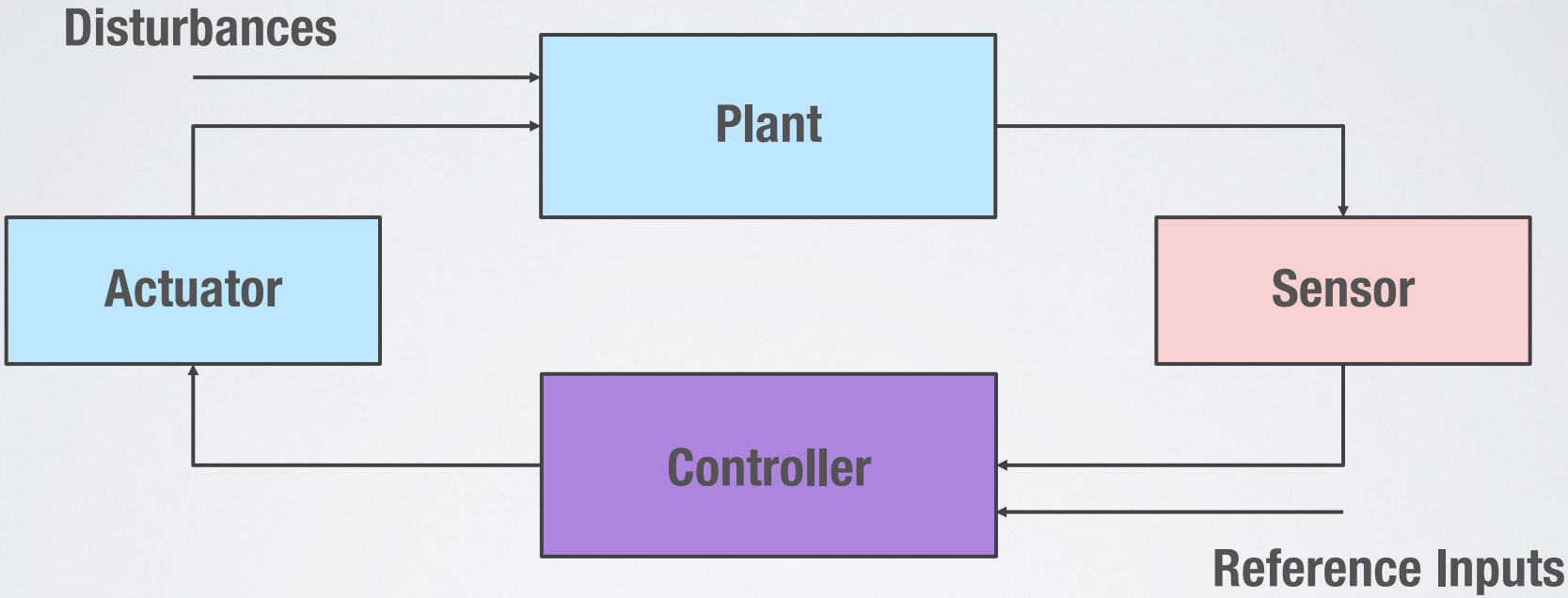
- A system of collaborating computational elements controlling physical entities



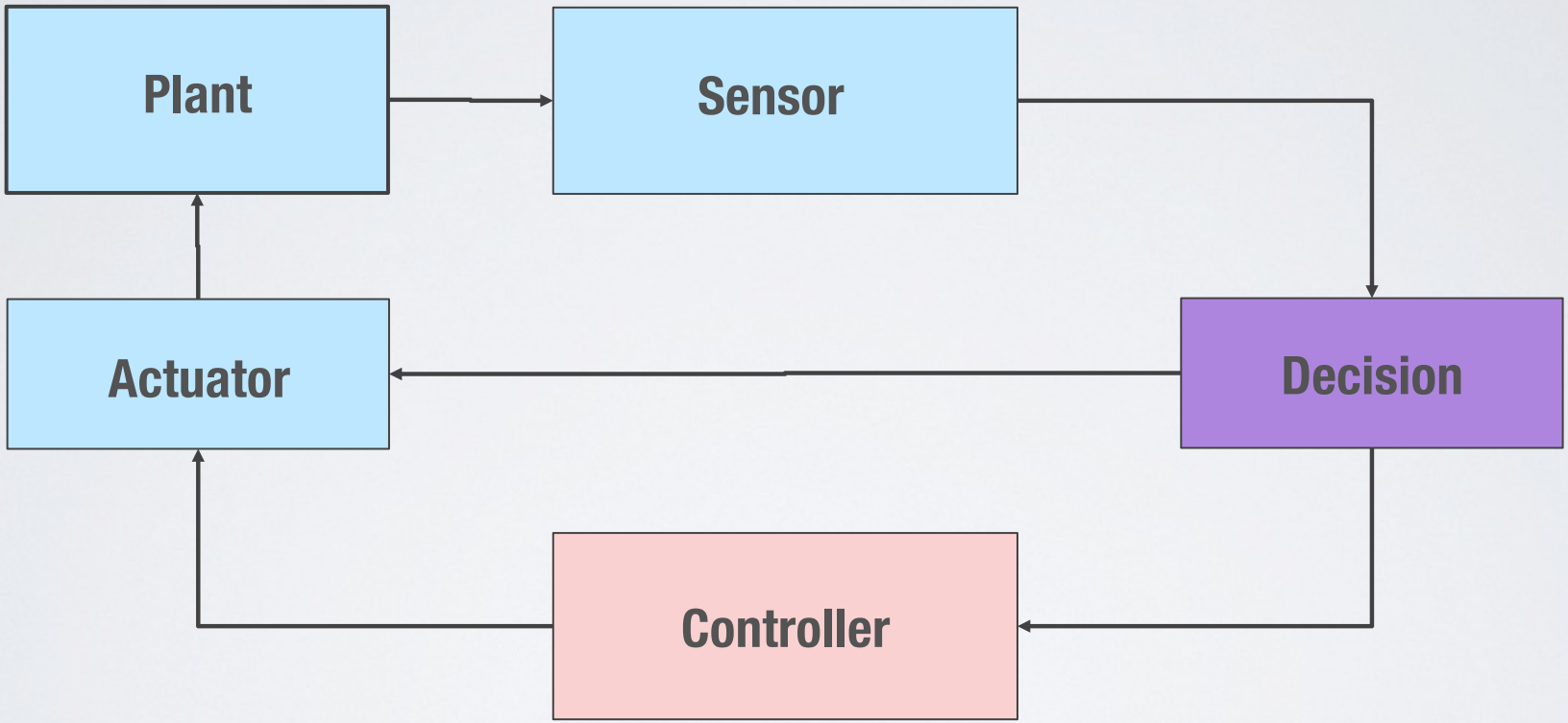
Context

- Projects on verification of cyber-physical systems, control systems, autonomous systems, …
- Focus on safety, performance, resource usage …
- Automotive, satellite, energy, manufacturing …

Controllers



Decision-Making Components



Development Process

Model-in-the-Loop Stage

Functional modeling:

- Controllers
- Plant
- Decision

Continuous and discrete Simulink models

Model simulation and testing

MATLAB SIMULINK

Software-in-the-Loop Stage

Architecture modelling

- Structure
- Behavior
- Traceability

System engineering modeling (SysML)

Analysis:

- Model execution and testing
- Model-based testing
- Traceability and change impact analysis
- ...

(partial) Code generation

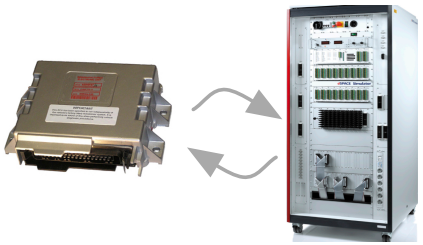
C SysML

Hardware-in-the-Loop Stage

Deployed executables on target platform

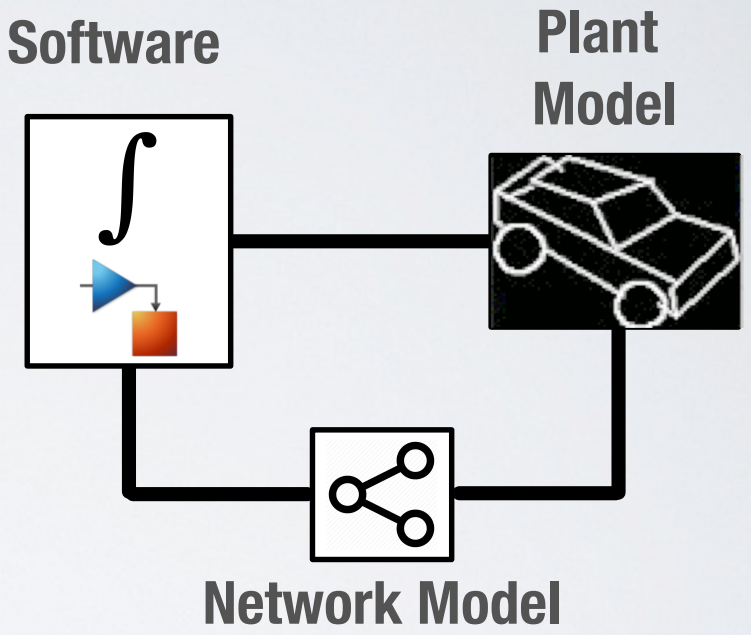
Hardware (Sensors ...)
Analog simulators

Testing (expensive)

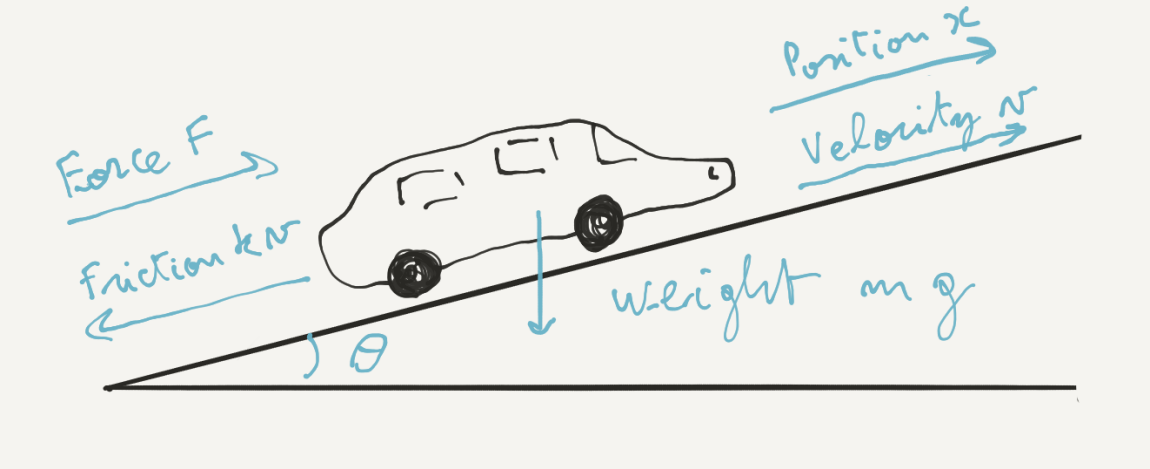


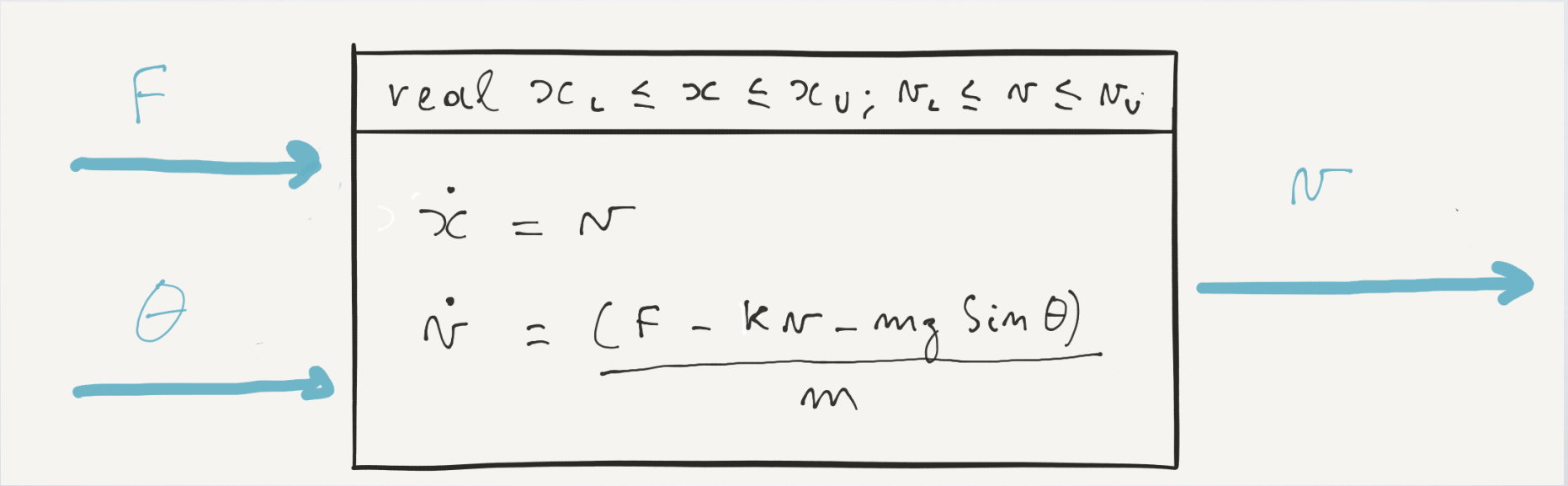
Simulink Models - Simulation

- Simulation Models
  - heterogeneous
  - continuous behavior
  - are used for
    - algorithm design testing
    - comparing design options



Cruise Control: Plant





Problem

- How do we automatically verify and test CP functional models (e.g., controller, plant, decision) at MiL?
- What types of requirements / properties do we check?
- Commercial tools:
  - Cannot handle continuous operators, floating point models (e.g., SLDV, Reactis)
  - Based on model structural coverage: low fault detection
  - Can only handle linear systems and specific properties (Linear analysis toolbox)

Challenges

- **Limited work** on verification and testing of controllers and decision-making components in CP systems
- Space of test input signals is **extremely large**.
- Model execution, especially when involving plant models, is **extremely expensive**.

More Challenges

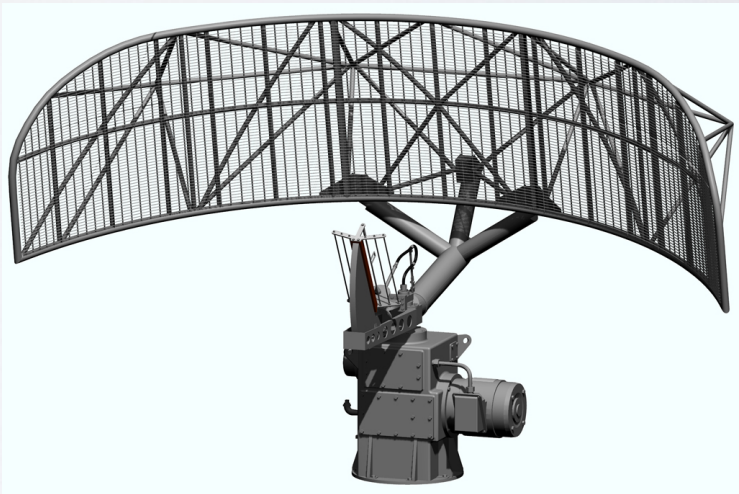
- Test oracles are not simple Boolean properties and easily known in advance – they involve analyzing changes in value over time (e.g., signal patterns) and assessing levels of risk.
- Simulatable plant model of the physical environment is not always available or fully accurate and precise.

Diversity Strategies

- **Strategy:** Maximize diversity of test cases
- **Assumption:** "the more diverse the test cases the higher their fault revealing capacity"
- **Challenge:** Define "diversity" in context, pair-wise similarity computation, test selection algorithm
- **Examples of early work**
  - ISSRE 2003, Leon and Podgurski, filtering and prioritizing test cases, relying on code coverage
  - FSE 2010, Hemmati et al., Similarity-based test selection applied to model-based testing based on state models for control systems, selection with GA
- **Full control on test budget:** Maximize fault detection for a given test suite size

Testing Controllers

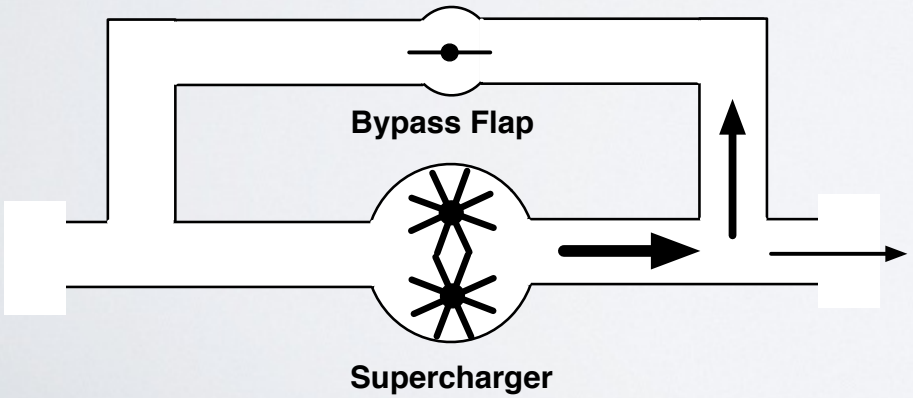
Controllers are Pervasive



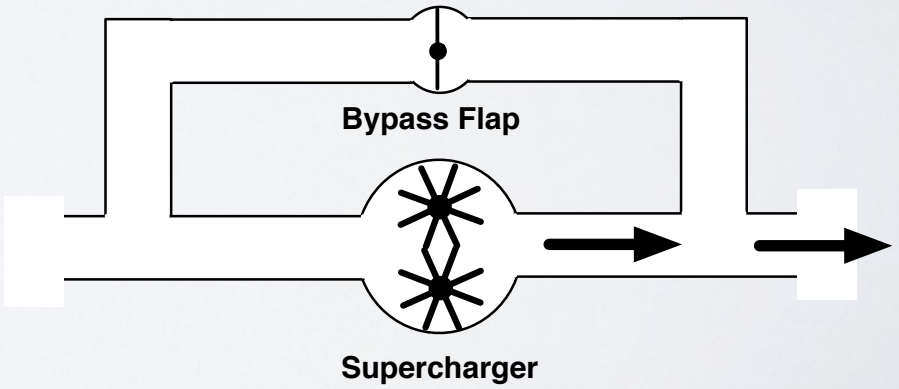
Simple Example

- Supercharger bypass flap controller
  - ✓Flap position is bounded within [0..1]
  - ✓Implemented in MATLAB/Simulink
  - ✓34 (sub-)blocks decomposed into 6 abstraction levels

DELPHI

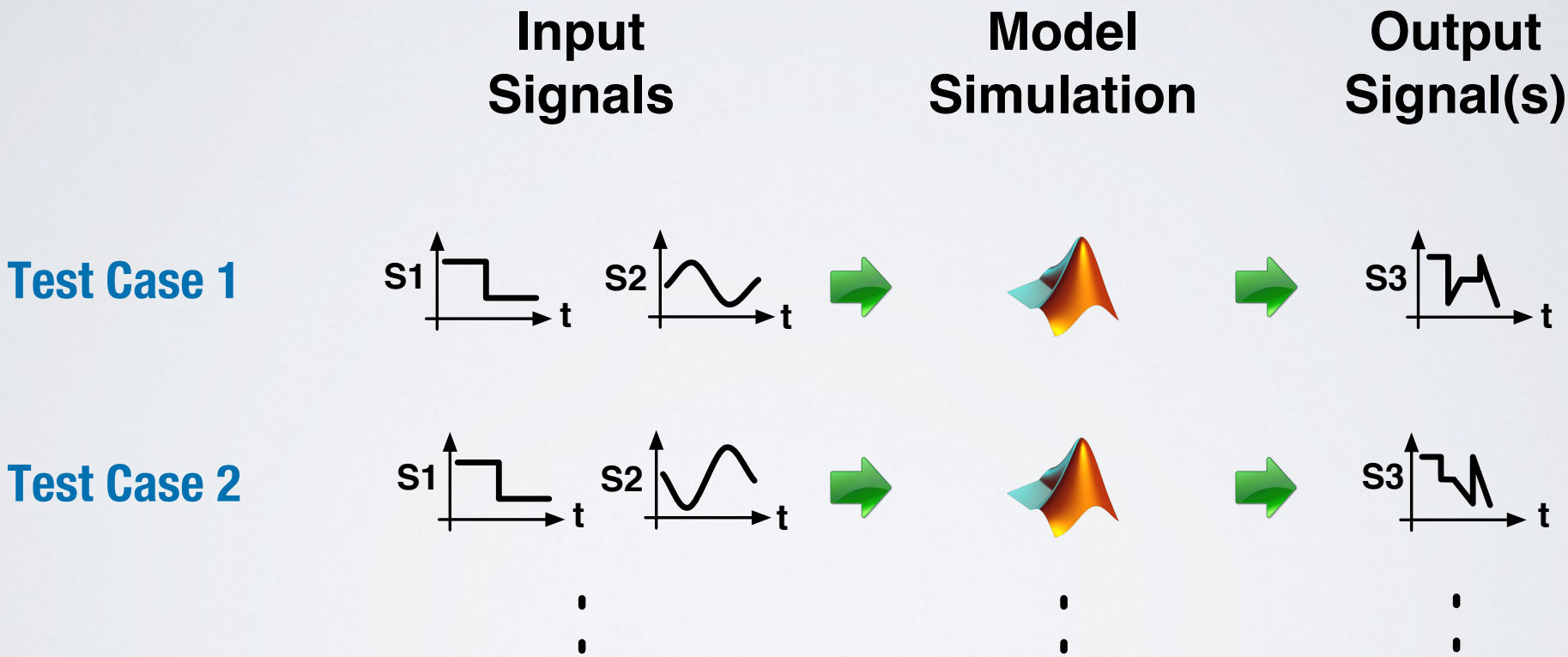


Flap position = 0 (open)

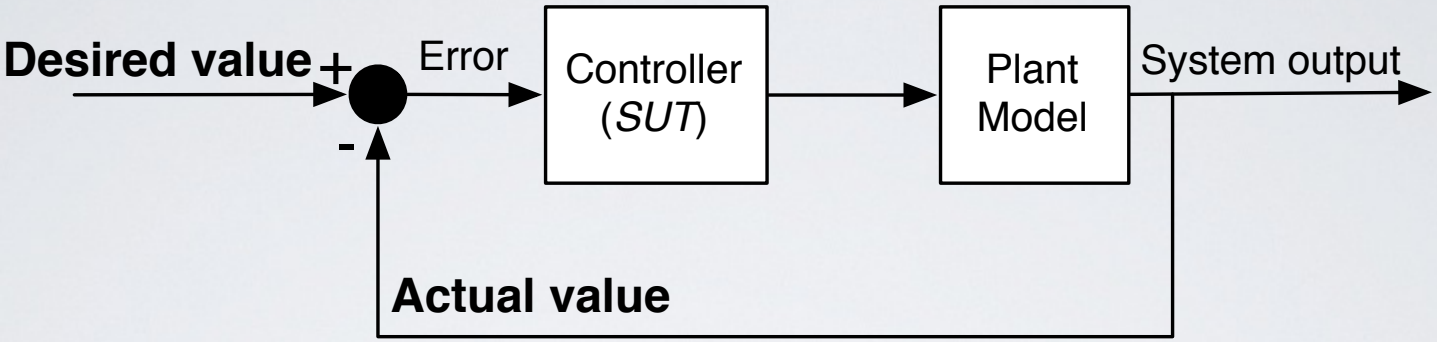


Flap position = 1 (closed)

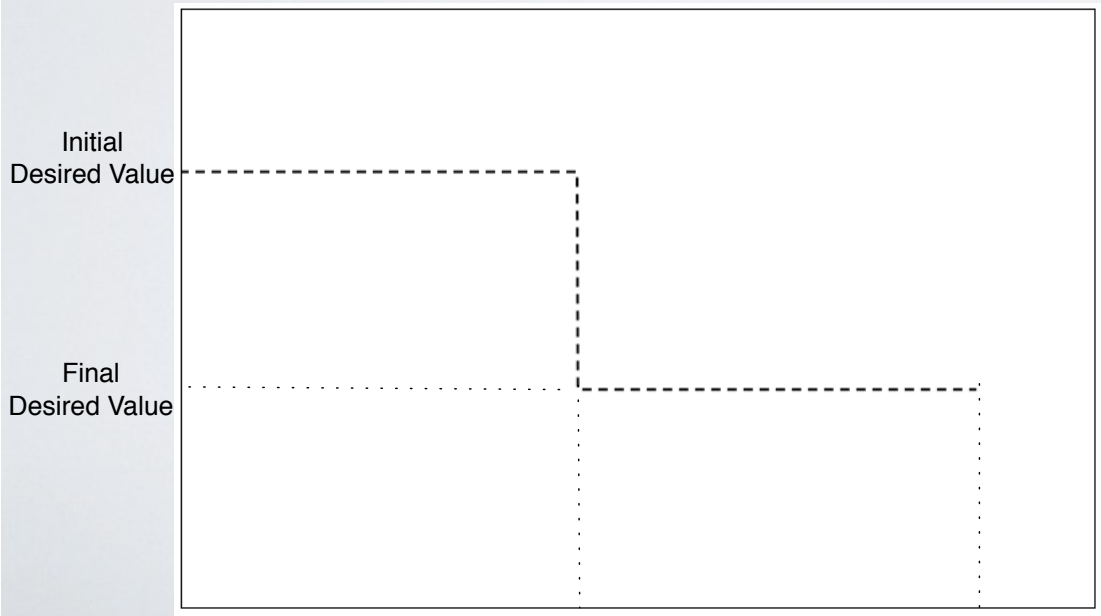
MiL Test Cases



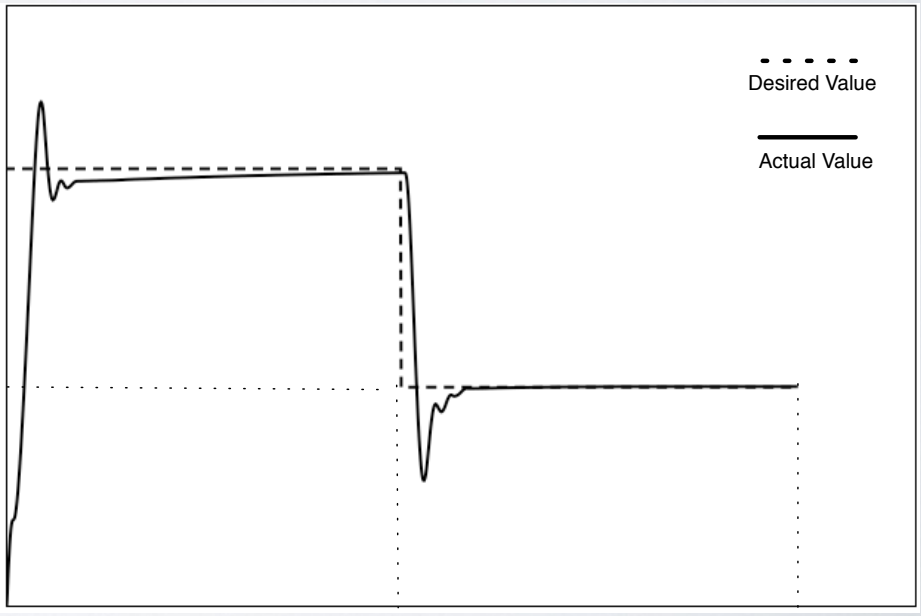
MiL Testing of Controllers



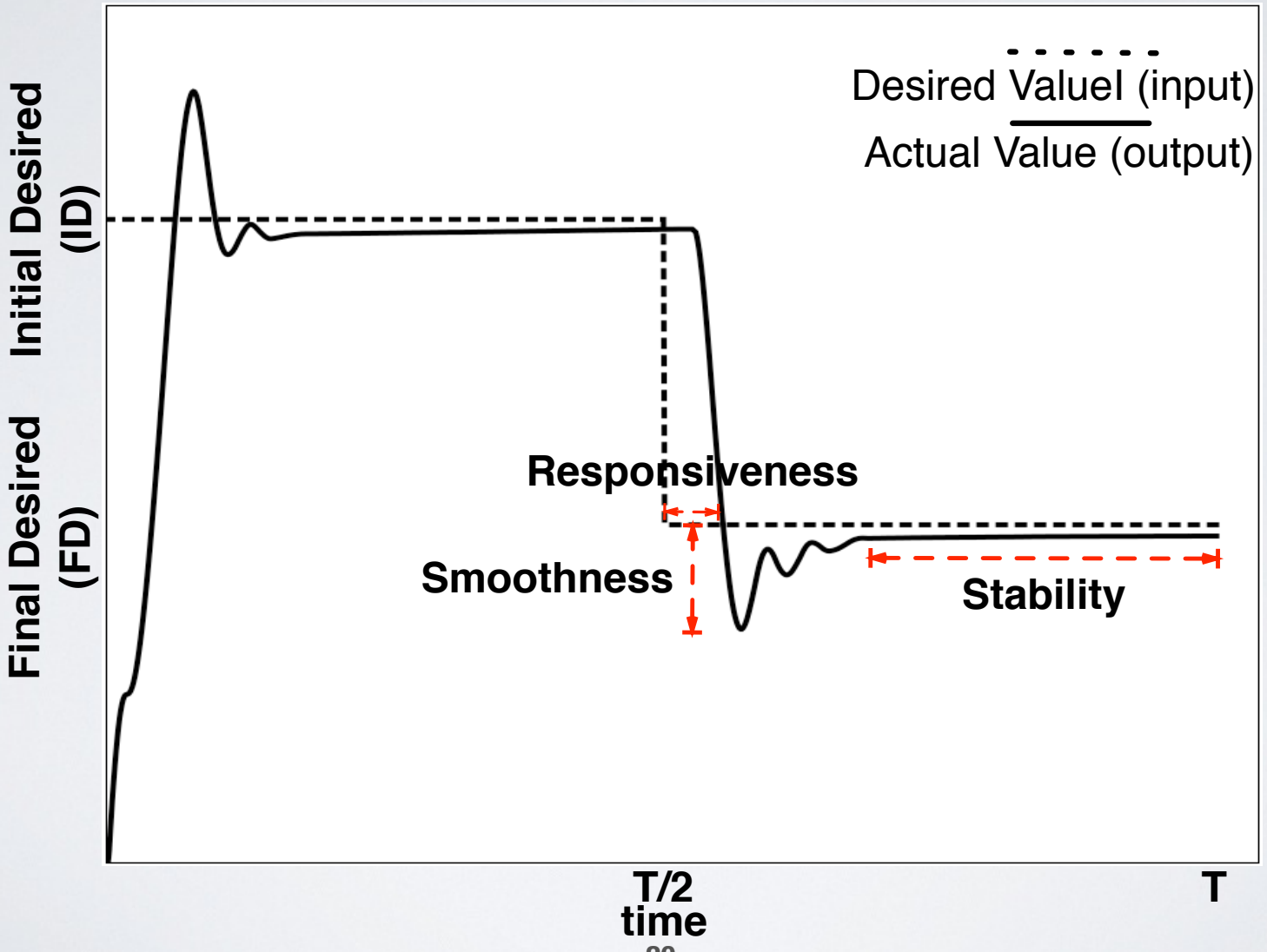
Test Input



Test Output

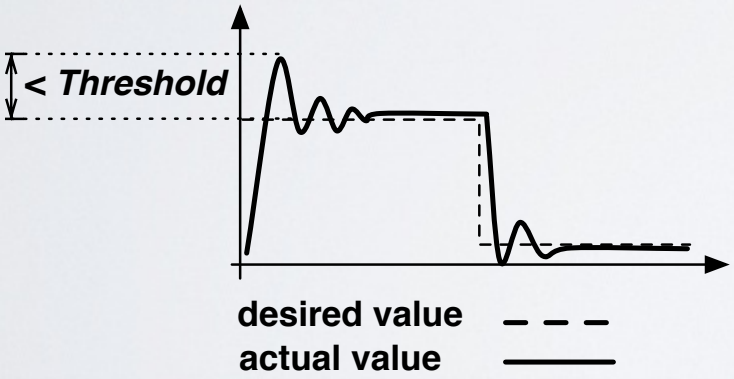


Requirements and Test Objectives

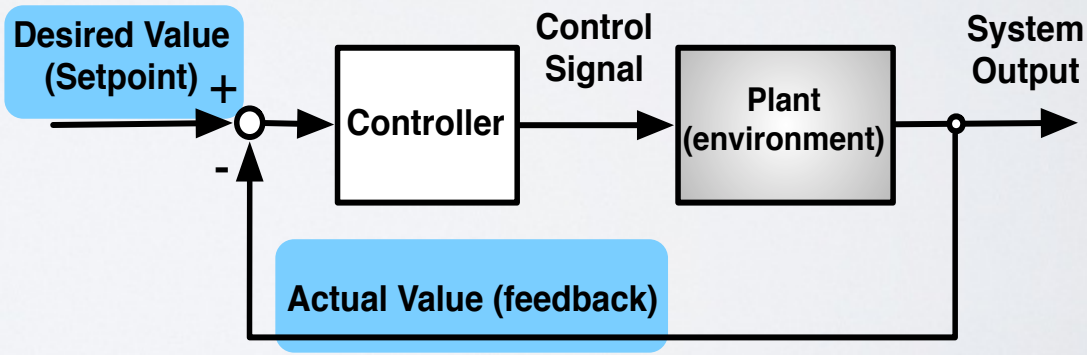


Test Generation Approach

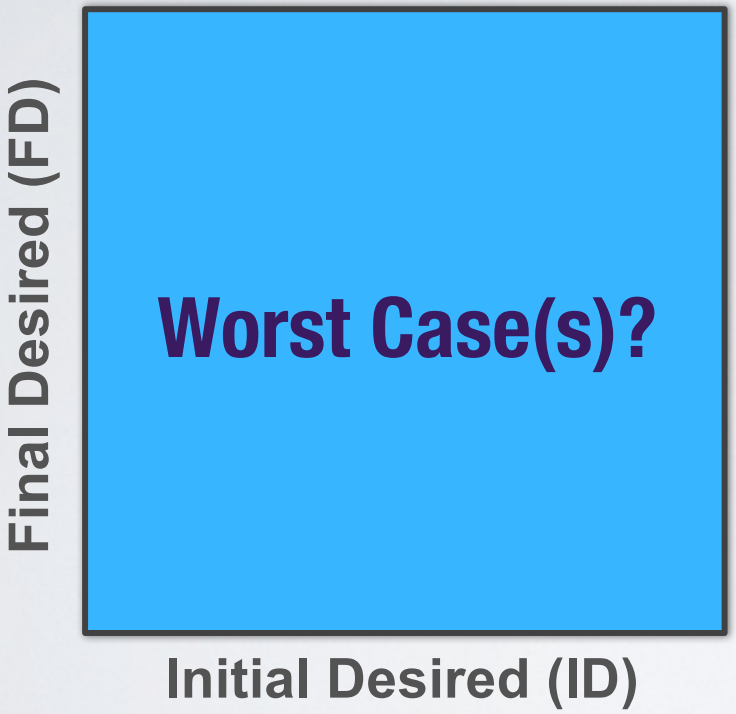
- We formalize controller's requirements in terms of **desired** and **actual** outputs
- We rely on **controller's feedback** to automate test oracles



Smoothness

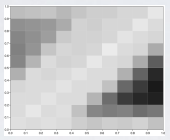


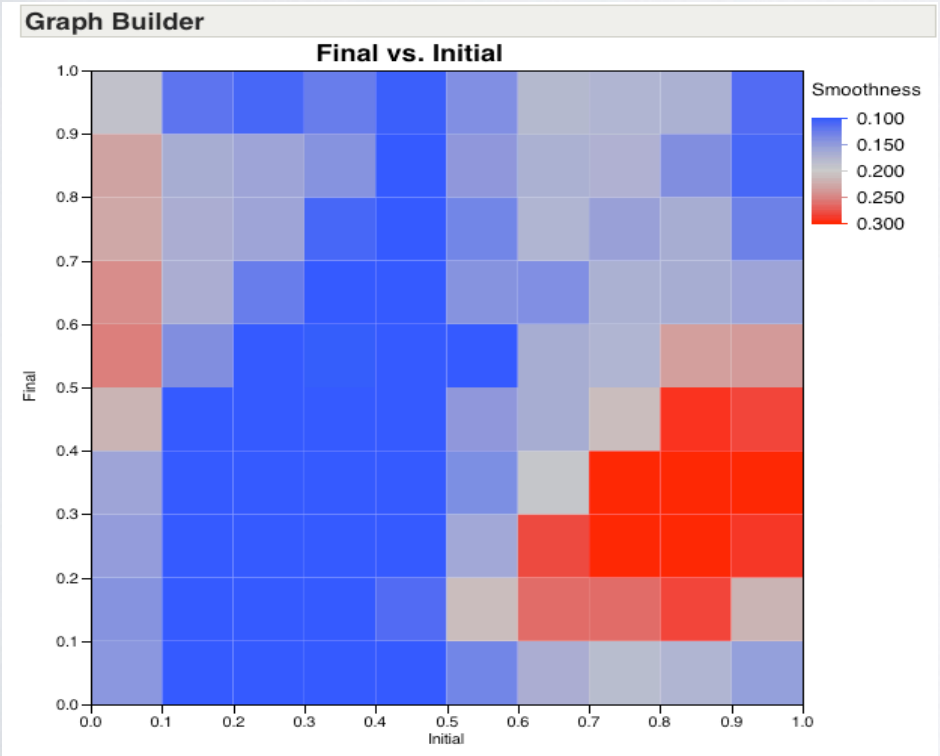
A Search-Based Test Approach

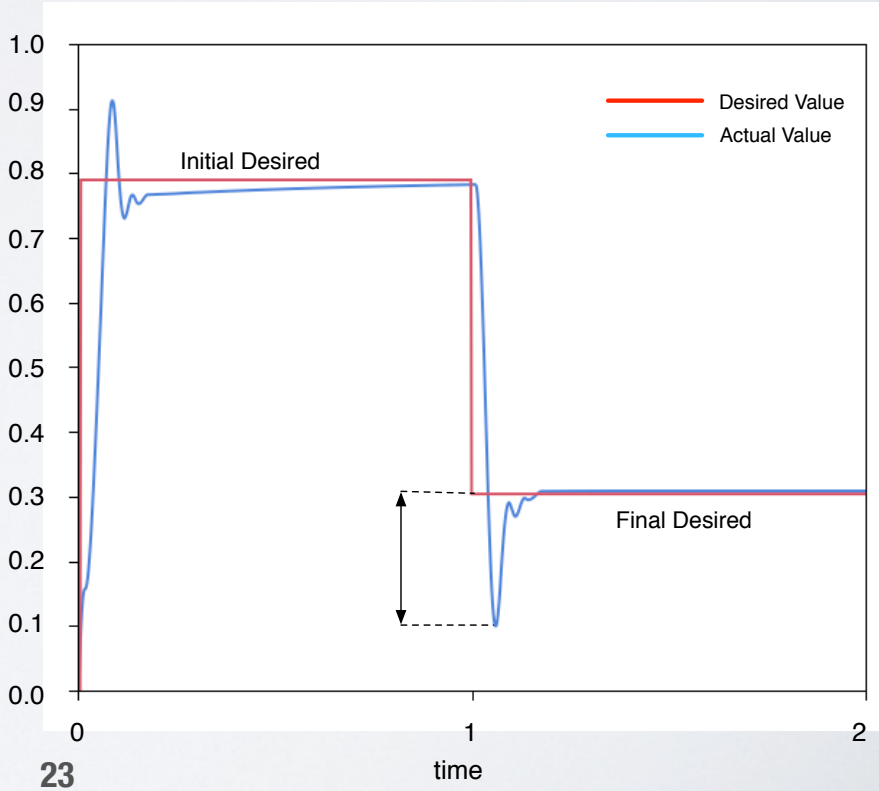


- Search directed by model execution feedback
- Finding worst case inputs
- Possible because of automated oracle (feedback loop)
- Different worst cases for different requirements
- Worst cases may or may not violate requirements

Initial Solution

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





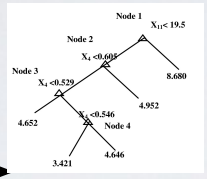
Results

- We found much worse scenarios during MiL testing than our partner had found so far
- These scenarios are also run at the HiL level, where testing is much more expensive: MiL results -> test selection for HiL
- But further research was needed:
  - Simulations are expensive
  - Configuration parameters

Final Solution

Objective Functions + Controller Model (Simulink)

1.Exploration with Dimensionality Reduction



Regression Tree

Domain Expert

List of Critical Partitions

2.Search with Surrogate Modeling

Worst-Case Scenarios

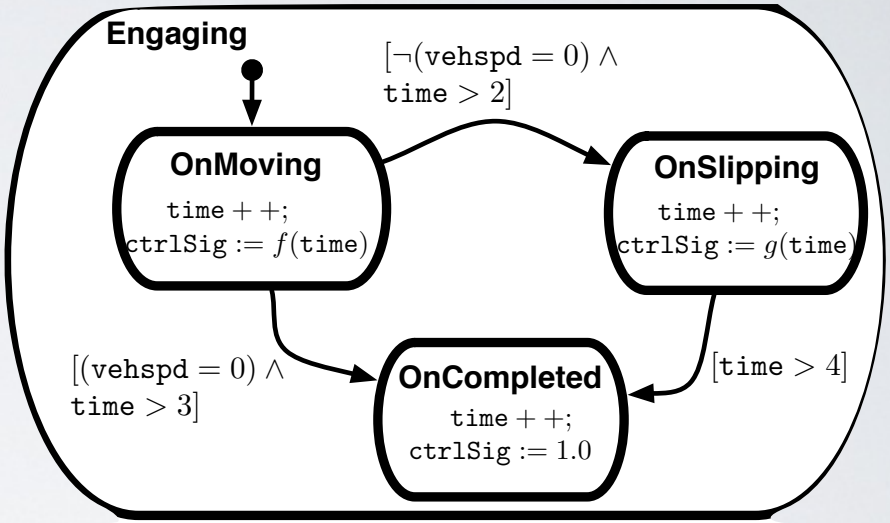
Dimensionality reduction to identify the significant variables (Elementary Effect Analysis)

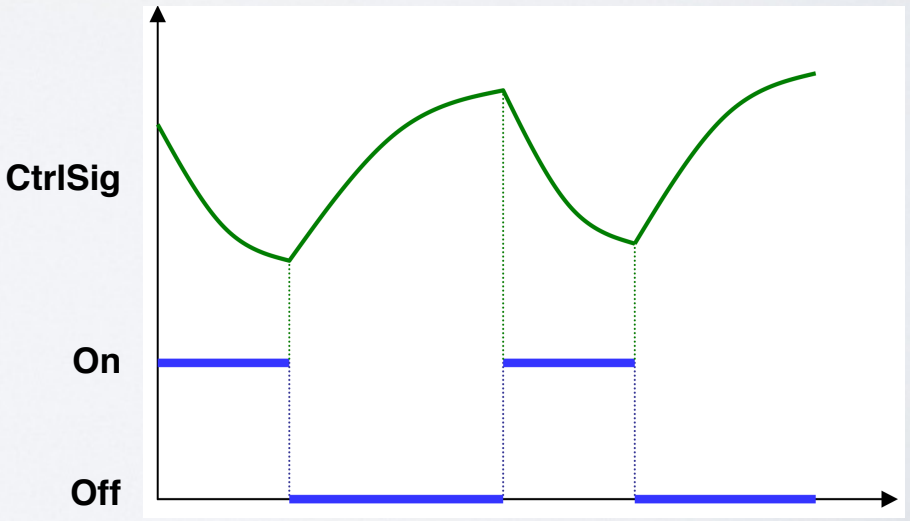
Visualization of the 8-dimension space using regression trees

Surrogate modeling to predict the fitness function and speed up the search (Machine learning)

Open Loop Controllers

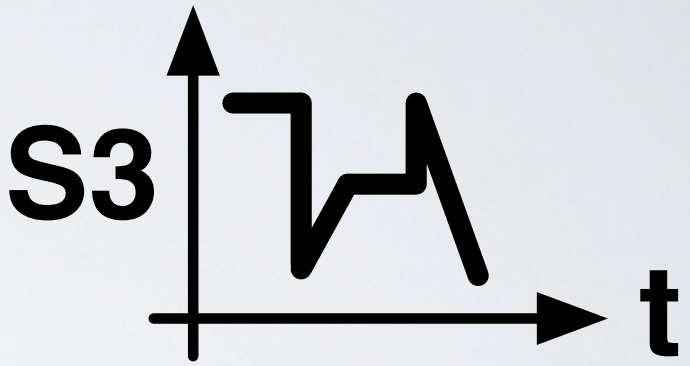
- Mixed discrete-continuous behavior: Simulink stateflows
- No plant model: Much quicker simulation time
- No feedback loop -> no automated oracle
- The main testing cost is the manual analysis of output signals
- Goal: Minimize test suites
- Challenge: Test selection
- Entirely different approach to testing

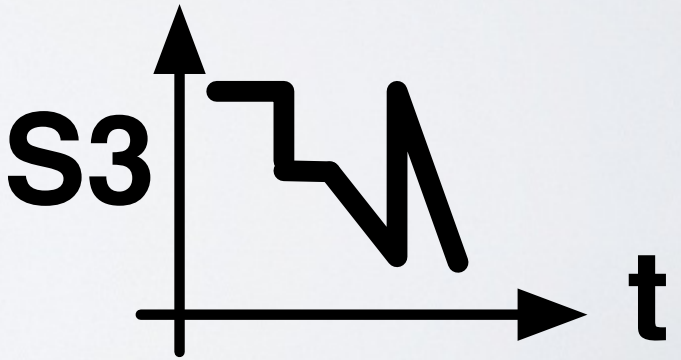




Selection Strategies Based on Search

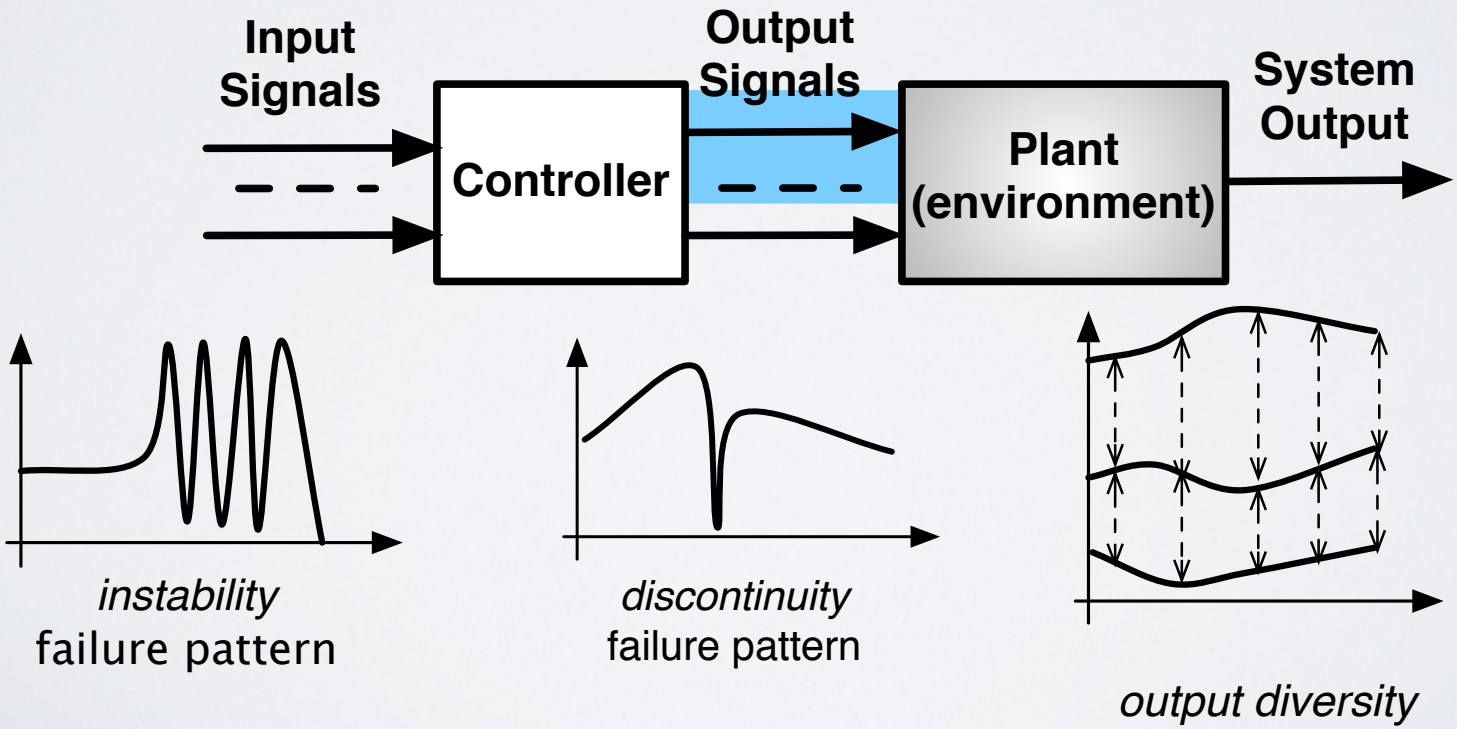
- White-box structural coverage
  - State Coverage
  - Transition Coverage
- Input signal diversity
- Output signal diversity
- Failure-Based selection criteria
  - Domain specific failure patterns
  - Output Stability
  - Output Continuity



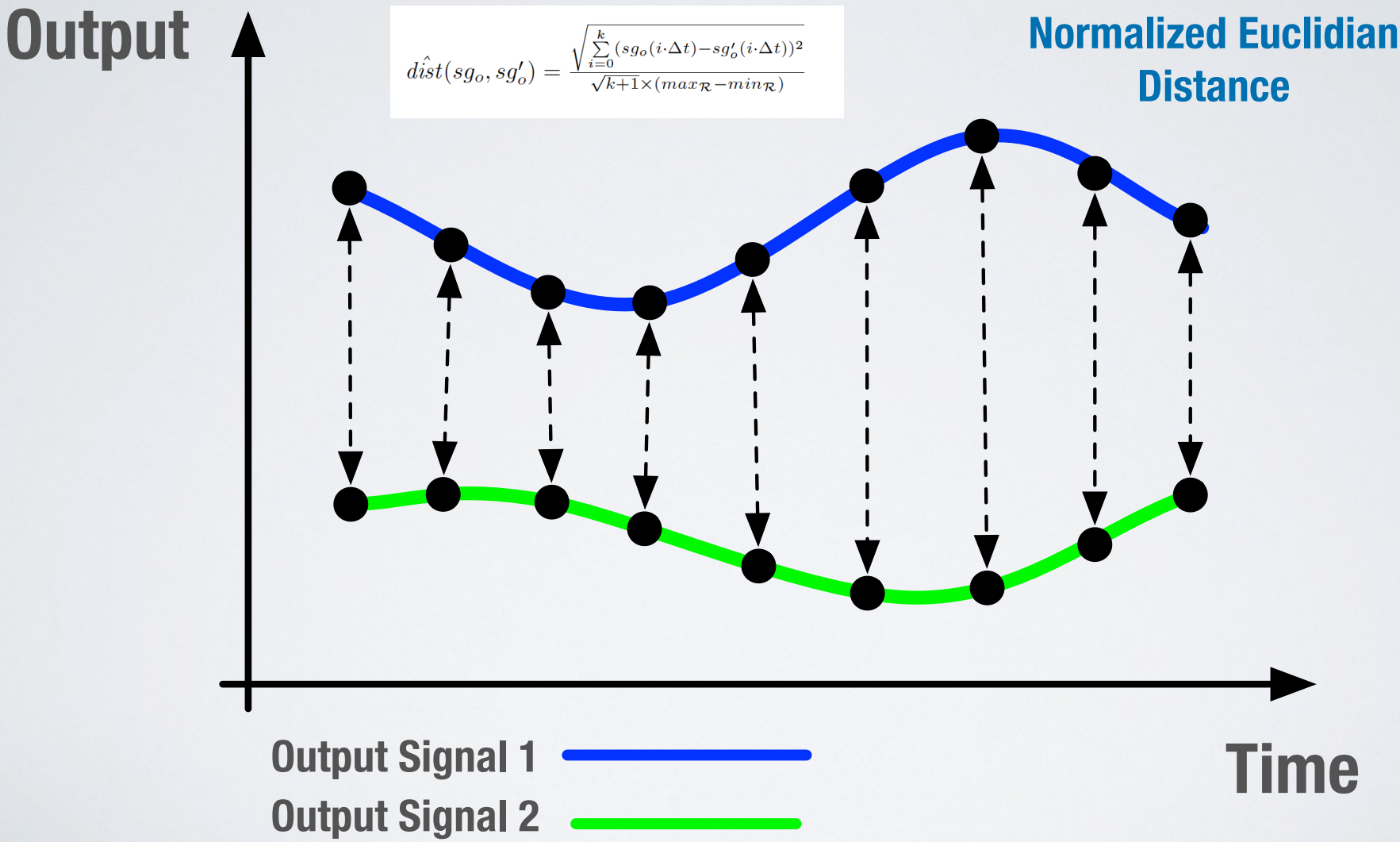


Test Generation Approach

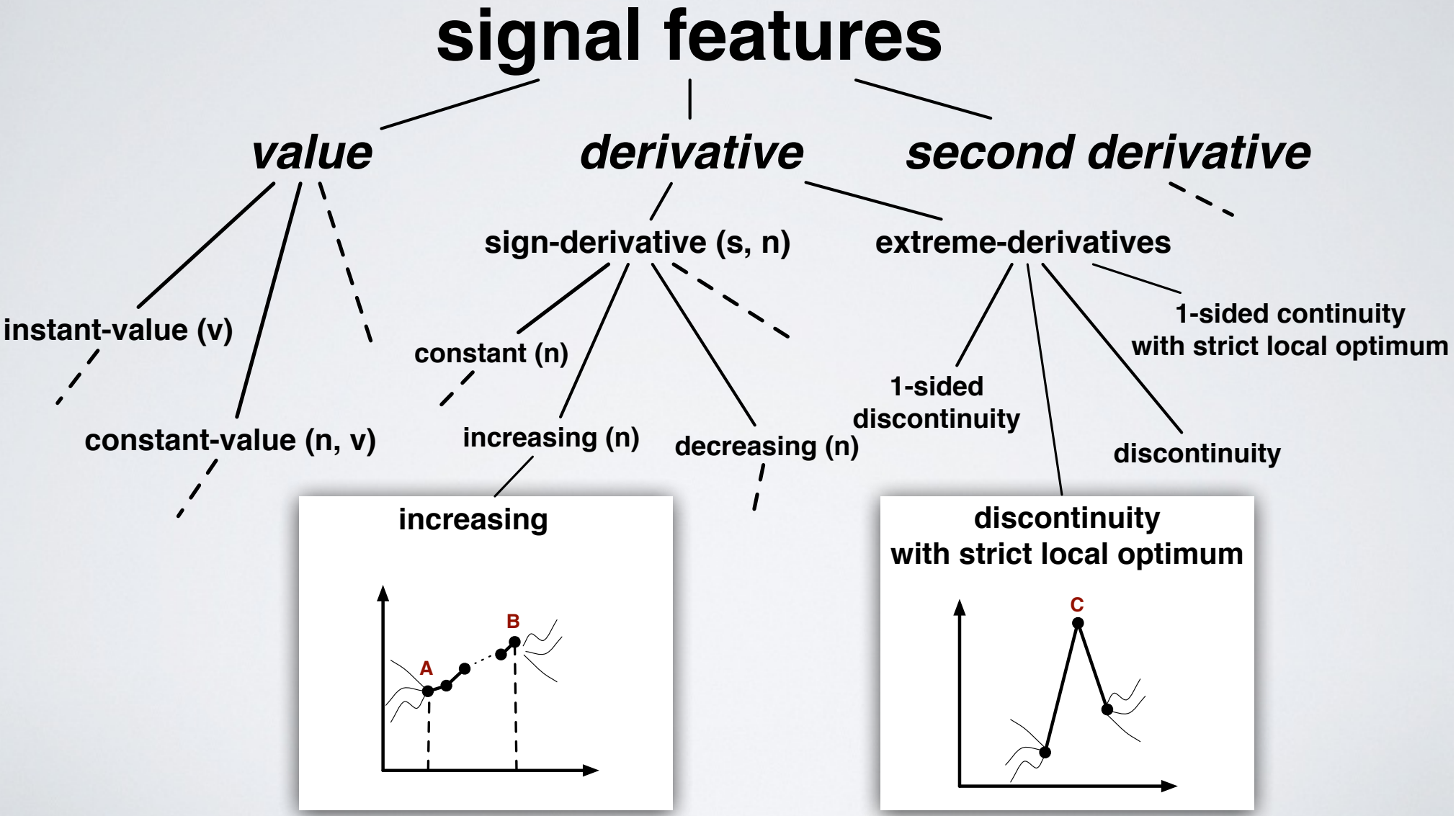
- We assume **test oracles are manual**
- We rely on output signals to **produce small test suites** with **high fault-revealing ability**



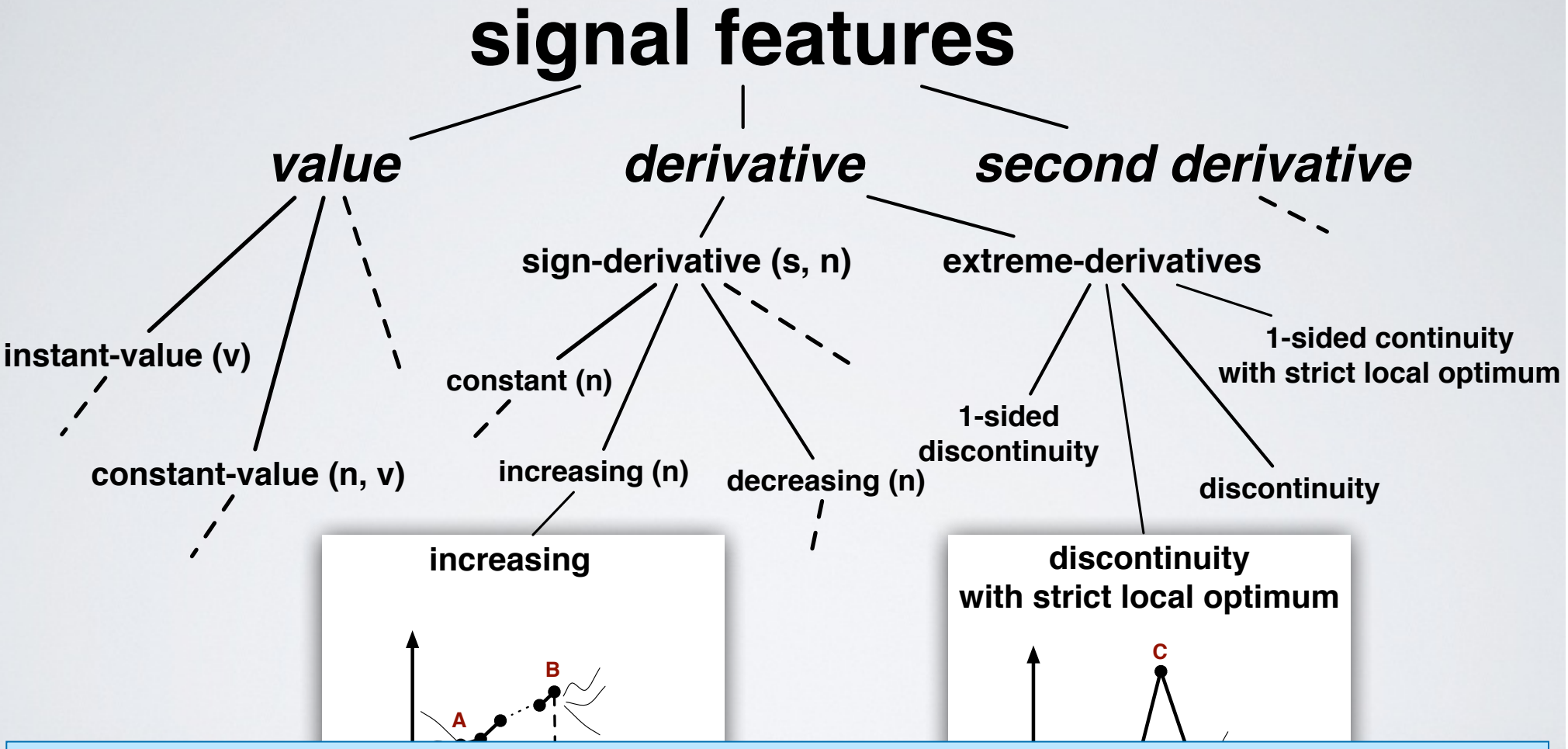
Output Diversity -- Vector-Based



Output Diversity -- Feature-Based



Output Diversity -- Feature-Based

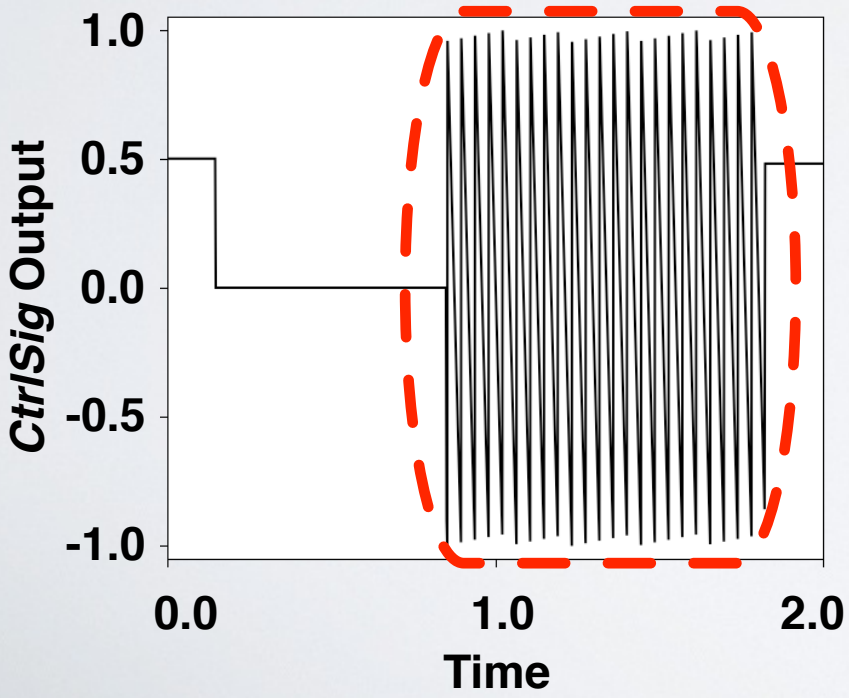


Similarity: To which extent any part of a signal is similar to a feature

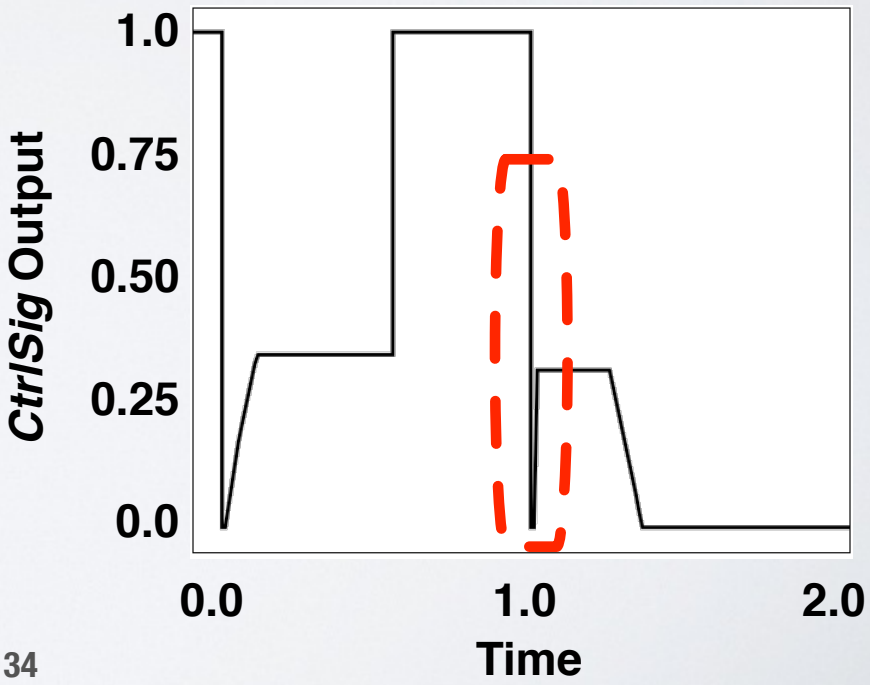
Failure-based Test Generation

- Search: Maximizing the likelihood of presence of specific failure patterns in output signals
- Domain-specific failure patterns elicited from engineers

Instability



Discontinuity



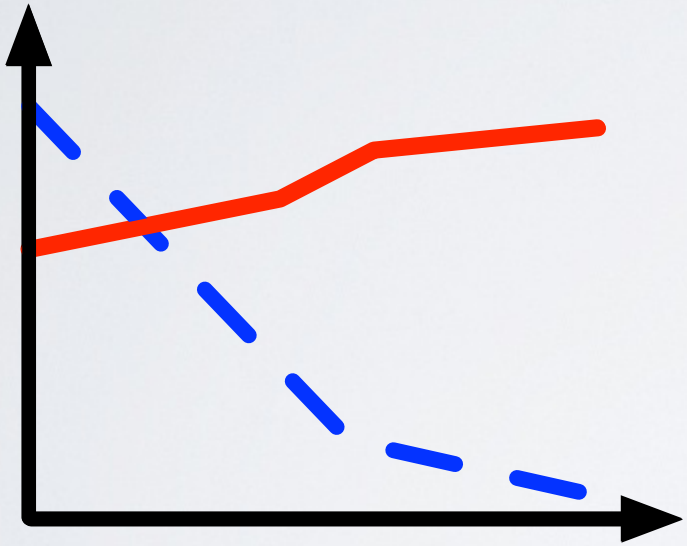
Search

- **Whole test suite generation** approach
- Used when objective functions characterize the test suite
- Optimize test objective for a given test suite size (budget for manual oracles)
- **Maximize the minimum distances** of each output signal vector from the other output signal vectors
- Adaptation of **Simulated Annealing**

Fault-Revealing Ability

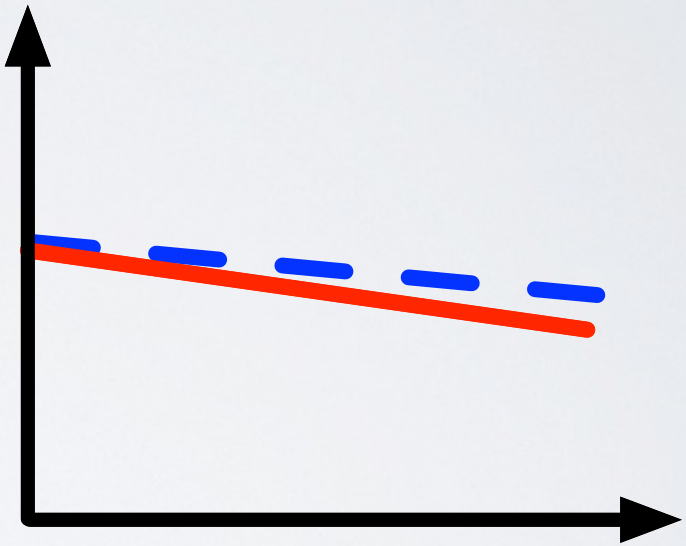
Covers the fault **and**

is Likely to reveal it



Covers the fault **but**

is very unlikely to reveal it



Faulty Model Output

Correct Model Output

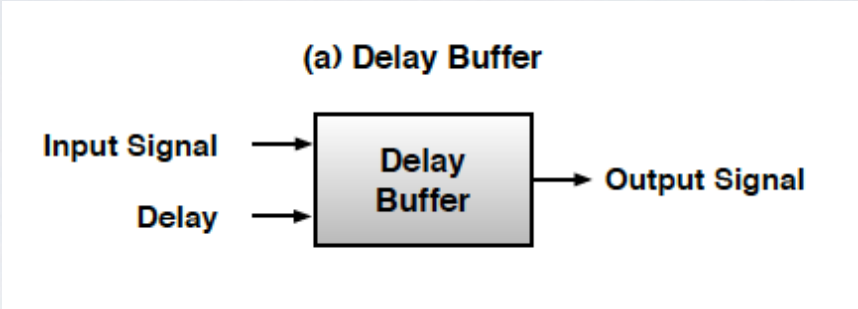
Results

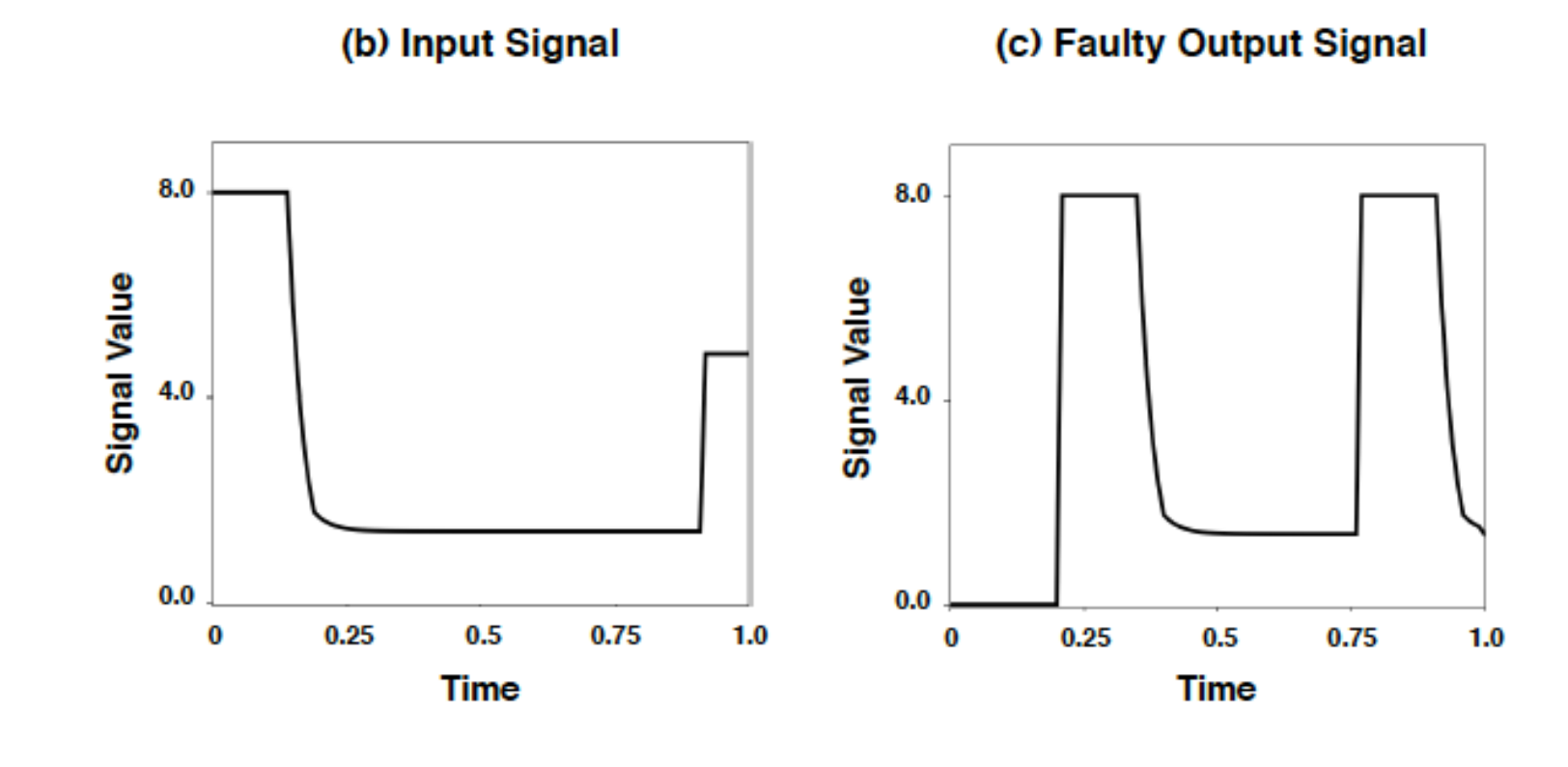
- The test cases resulting from state/transition coverage algorithms **cover the faulty parts** of the models
- However, they **fail to generate output signals that are sufficiently distinct** from expectations, hence yielding a low fault revealing rate
- **Diversity strategies** significantly outperforms coverage-based and random testing
- **Output-based algorithms** are much more effective, both based on diversity and failure patterns

Results

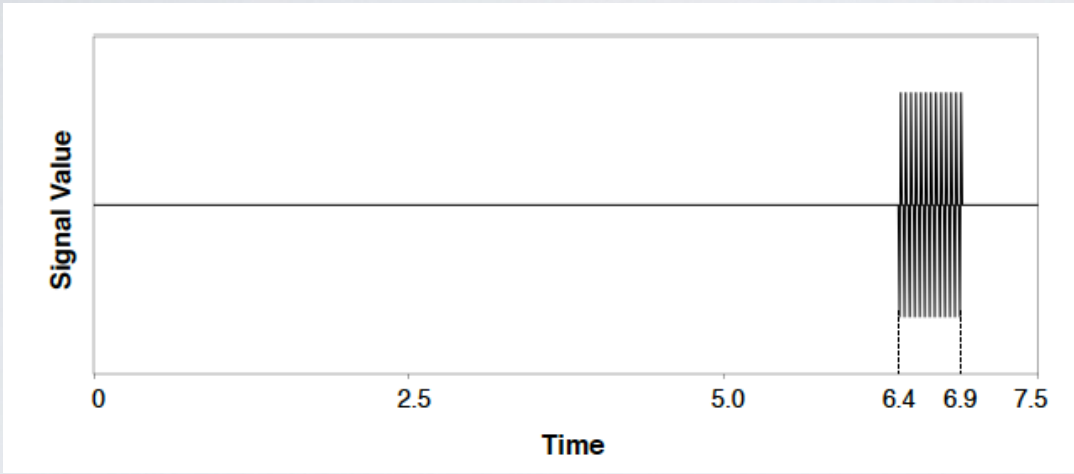
- **Feature-based diversity** fares significantly better than vector-based diversity
- Strategies based on **failure patterns** find different types of faults than diversity strategies
- **Existing commercial tools:** Not effective at finding faults, not applicable to entire Simulink models, e.g., Simulink Design Verifier

Example Failures



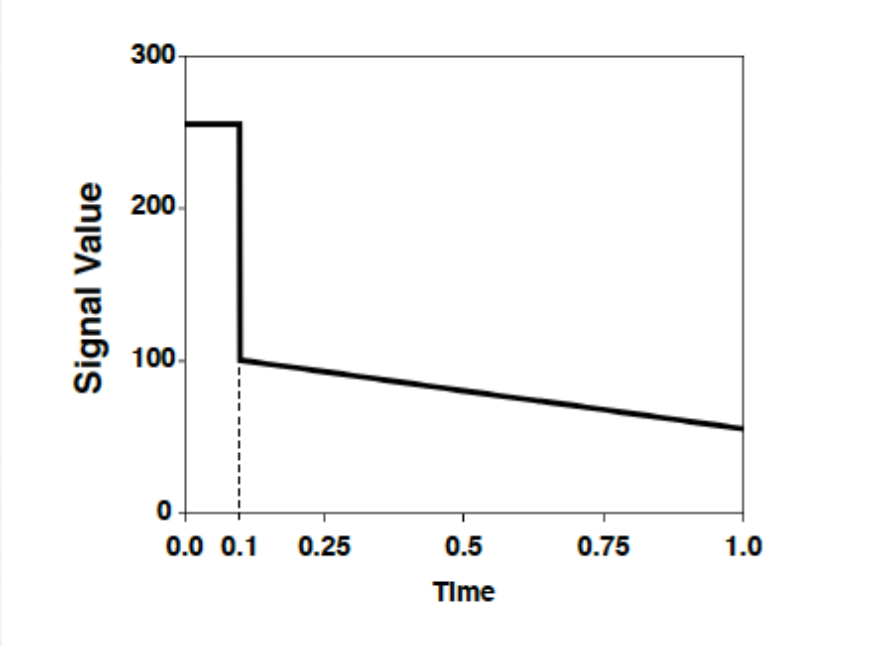


Example Failures



Instability

Discontinuity



Conclusions

- Maximizing output diversity helps identify scenarios where the discrepancy between the produced and expected signal is large
- Useful when test output signals are analyzed manually
- Simulink models and their outputs are complex
- Helps maximize fault detection within a fixed budget
- Properly defining diversity was a challenge

Reflections on Diversity

- Useful strategy when no precise guidance, directly related to the test objectives, is available for the search
- In the general, the key issue is how to define diversity in an optimal way given the objectives
- In practice, how diversity is defined also depends on what information is available at a reasonable cost
- The time complexity of computing diversity is a major cost of the search – it must be accounted for

Acknowledgements

- Shiva Nejati
- Reza Matinnejad
- Delphi Automotive Systems, Luxembourg

References

- R. Matinnejad et al., “MiL Testing of Highly Configurable Continuous Controllers: Scalable Search Using Surrogate Models”, IEEE/ACM ASE 2014 (Distinguished paper award)
- R. Matinnejad et al., “Effective Test Suites for Mixed Discrete-Continuous Stateflow Controllers”, ACM ESEC/FSE 2015 (Distinguished paper award)
- R. Matinnejad et al., “Automated Test Suite Generation for Time-continuous Simulink Models“, IEEE/ACM ICSE 2016
- R. Matinnejad et al., “Test Generation and Test Prioritization for Simulink Models with Dynamic Behavior“, under minor revision with IEEE TSE

SV.lu
software verification & validation

SnT
securityandtrust.lu

Testing of Cyber-Physical Systems:

Diversity-driven Strategies

Lionel Briand

COW 57, London, UK

uni.lu
UNIVERSITÉ DU LUXEMBOURG