Model-based Testing of Automotive Systems

Eckard Bringmann and Andreas Krämer

ICST'08

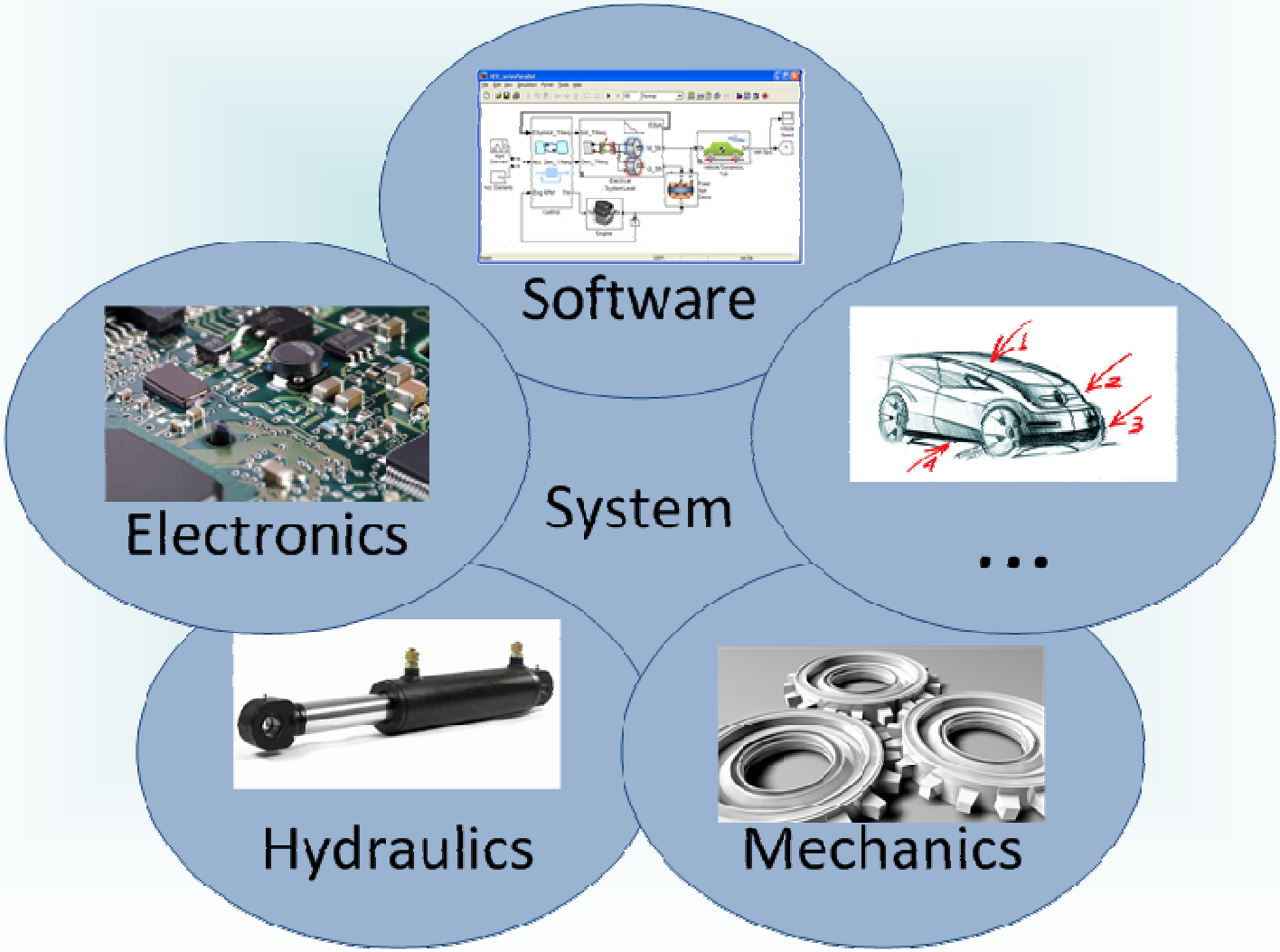
Presented by Julia Rubin

on November 21, 2012

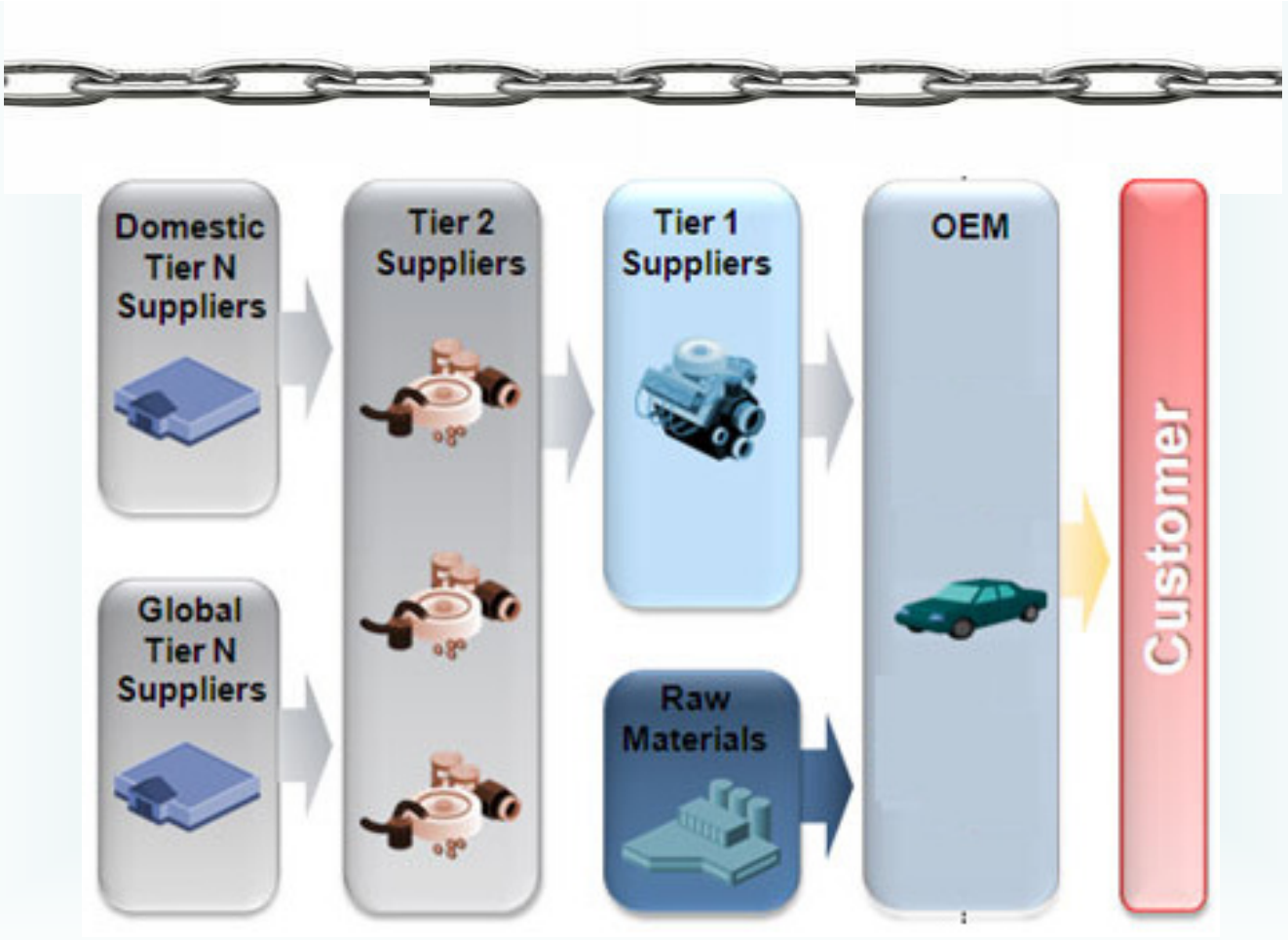




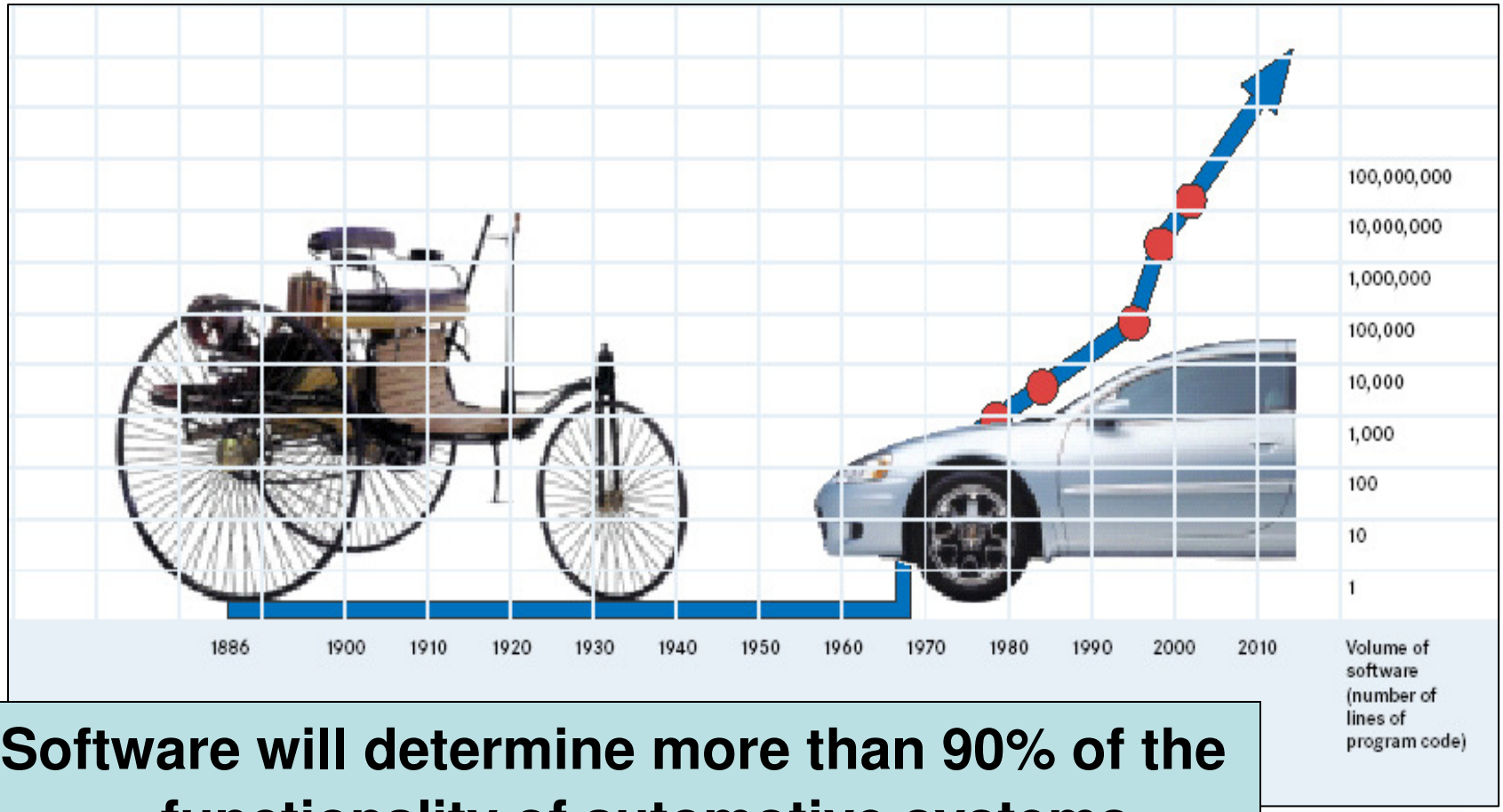

Multidisciplinary Business



Supply Chain of Components



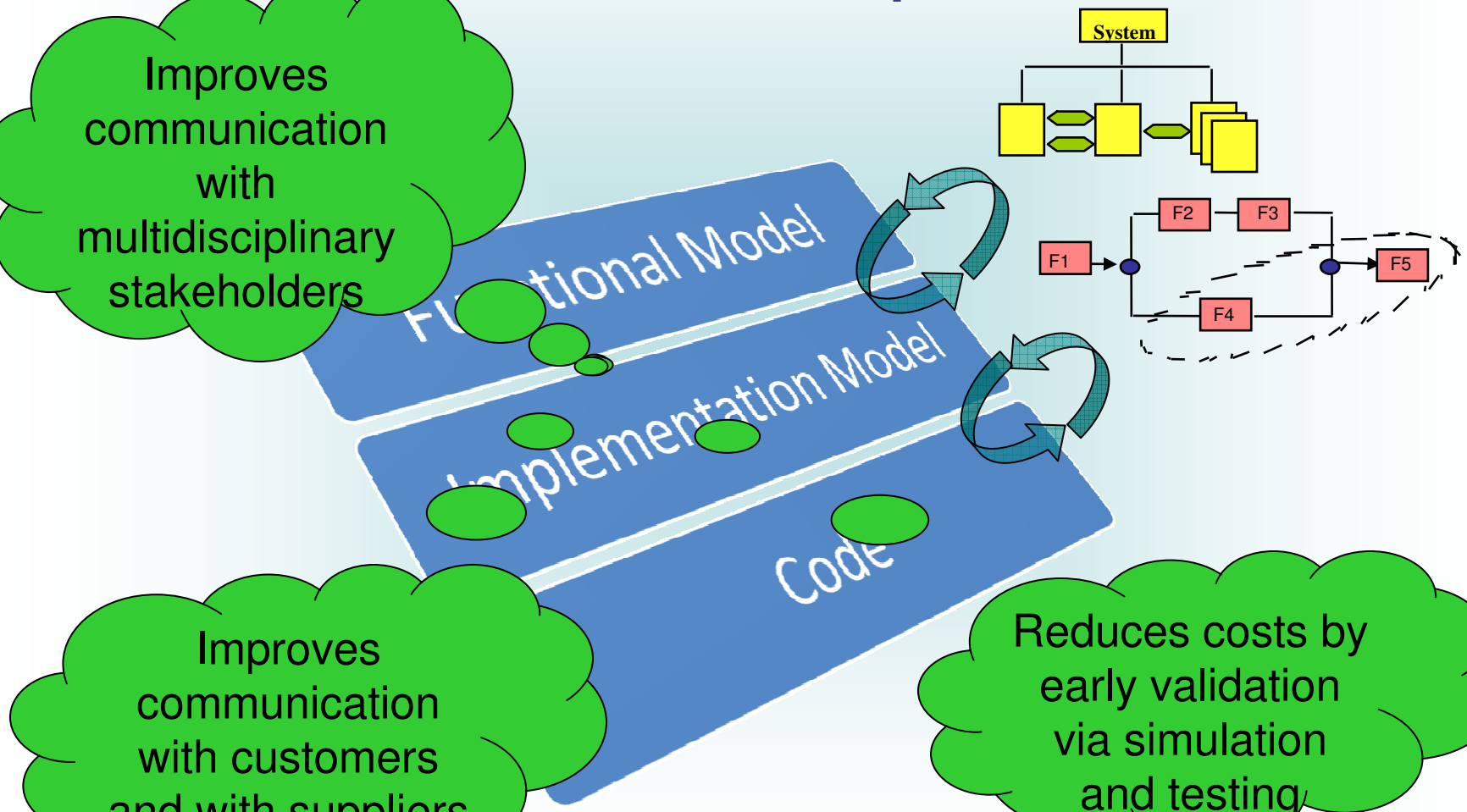
Innovation is Driven by Software



Software will determine more than 90% of the functionality of automotive systems

Model-based Development Became Popular

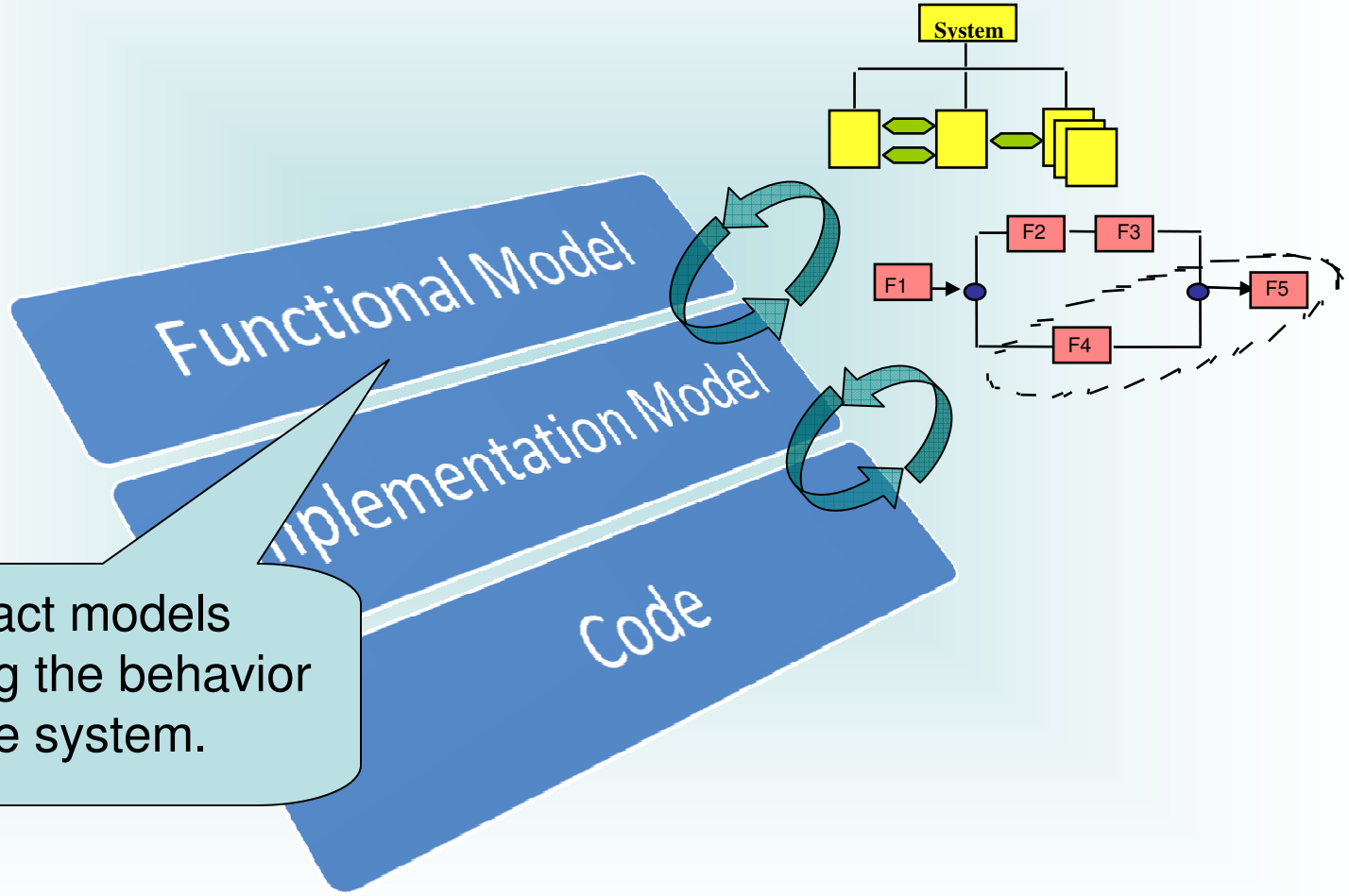
Improves communication with multidisciplinary stakeholders



Improves communication with customers and with suppliers

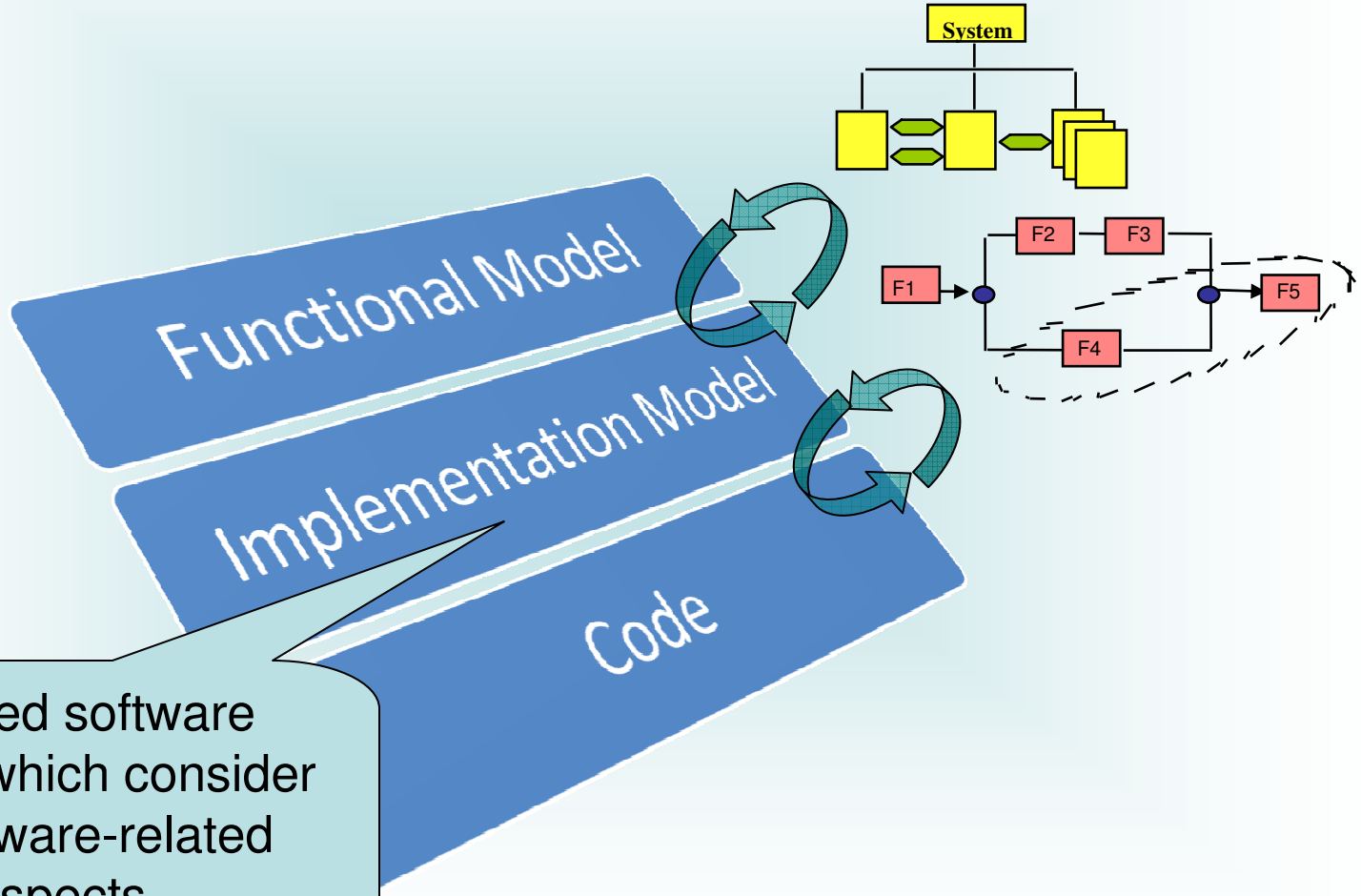
Reduces costs by early validation via simulation and testing

Model-based Development Became Popular



Abstract models describing the behavior of the system.

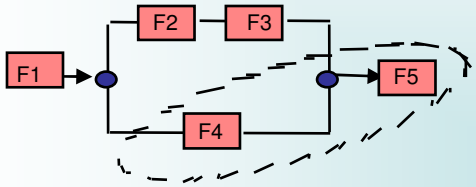
Model-based Development Became Popular



Detailed software models which consider all software-related aspects

Validation Stages

MiL: Model in the loop

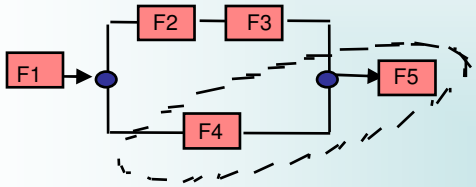


Models run in a virtual environment

- *Functional models* – at the system level
- *Implementation models* – at an individual module and a whole system levels

Validation Stages

MiL: Model in the loop



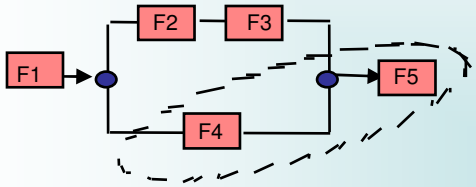
SiL: Software in the loop

```
int main(){
  bind_sut_inputs_to_vm;
  bind_sut_outputs_to_vm;
  bind_sut_parameters_to_vm;
  while testrun {
    sut_code
    tpt_vm_run_one_cycle
  }
}
```

Software runs in
a virtual environment,
without any hardware

Validation Stages

MiL: Model in the loop

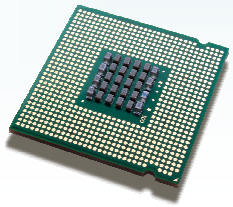


SiL: Software in the loop

```
int main(){
  bind_aut_inputs_to_vm;
  bind_aut_outputs_to_vm;
  bind_aut_parameters_to_vm;
  while testrun {
    aut_code
    tpt_vm_run_one_cycle
  }
}
```

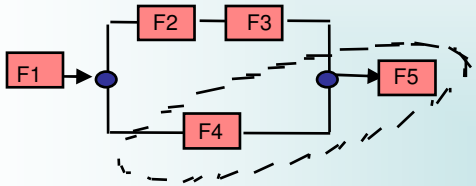
Software runs on a target processor or an emulator, but with a proprietary hardware (ECU)

PiL: Processor in the loop



Validation Stages

MiL: Model in the loop



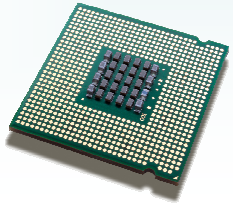
HiL: Hardware in the loop

SiL: Software in the loop

```
int main(){
  bind_sut_inputs_to_vm;
  bind_sut_outputs_to_vm;
  bind_sut_parameters_to_vm;
  while testrun {
    sut_code
    tpt_vm_run_one_cycle
  }
}
```

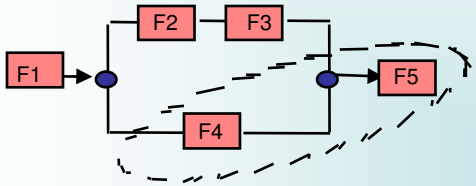
Final ECU with simulated environment

PiL: Processor in the loop



Validation Stages

MiL: Model in the loop



HiL: Hardware in the loop

SiL: Software in the loop

```
int main(){
  bind_sut_inputs_to_vm;
  bind_sut_outputs_to_vm;
  bind_sut_parameters_to_vm;
  while testrun {
    sut_code
    tpt_vm_run_one_cycle
  }
}
```

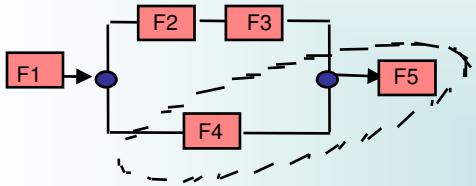
Test Rig

PiL: Pro

The environment consist of physical components (electrical, mechanical, hydraulic)

Validation Stages

MiL: Model in the loop



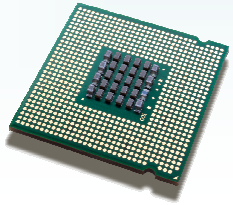
HiL: Hardware in the loop

SiL: Software in the loop

```
int main(){
  bind_sut_inputs_to_vm;
  bind_sut_outputs_to_vm;
  bind_sut_parameters_to_vm;
  while testrun {
    sut_code
    tpt_vm_run_one_cycle
  }
}
```

Test Rig

PiL: Processor in the loop



Car

Requirements for Testing

1. *Automation* to support multiple intermediate releases due to software hardware co-design
2. *Portability* between validation stages
3. *Systematic test case selection* to ensure coverage and lack of redundancies
4. *Readability* – "lingua franca" for multiple stakeholders
5. *"Closed loop" behavior* – the execution of the test depends on the behavior of the tested system
6. *Real-time behavior*
7. *Continuous signals*

Outline

1. Motivation: challenges of in-car software development process
2. Requirements for testing
3. TPT tool for automotive model-based testing
4. Experience report
5. Future work
6. Summary and discussion

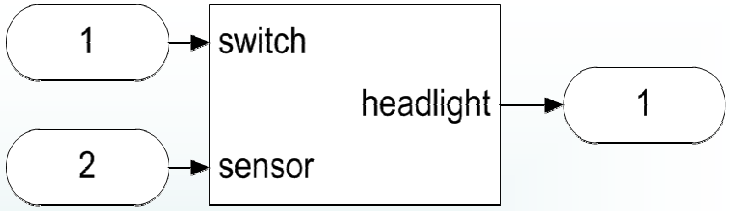
TPT for Automotive Model-Based Testing

1. Test language for describing test cases

2. A method to represent a test suite for a system under test (SUT)

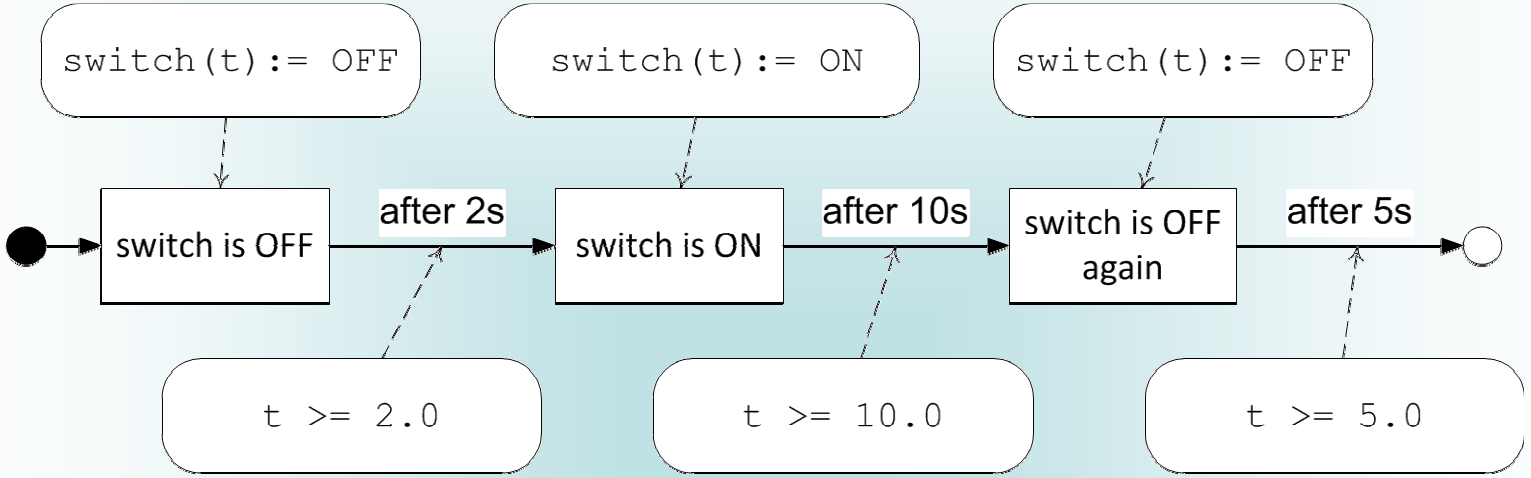
3. A set of adapters for development environments at different lifecycle stages

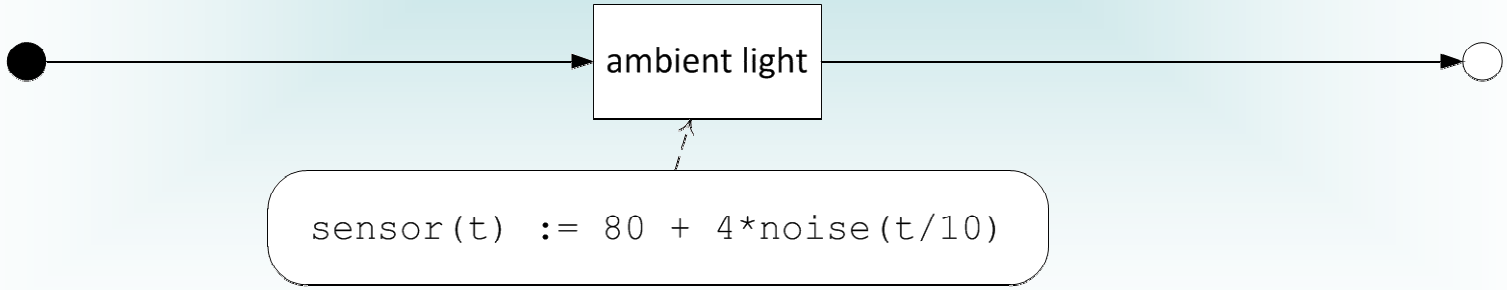
Example: Exterior Headlight Controller



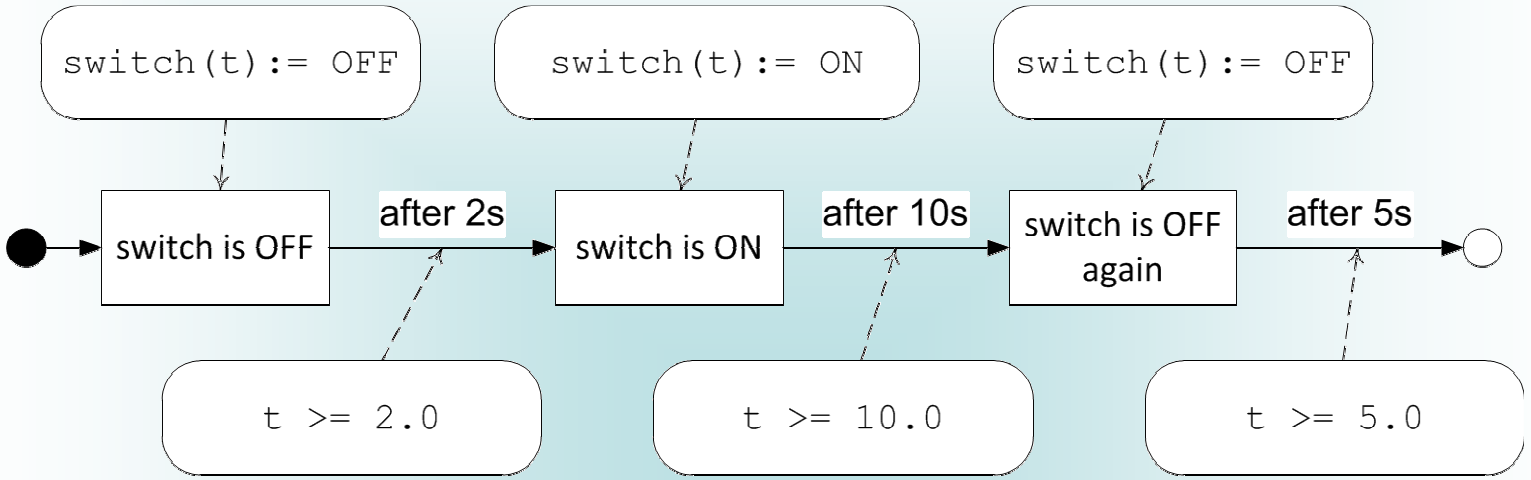
1. A switch with three states: ON, OFF, and AUTO
2. ON: turn on
3. OFF: turn off
4. AUTO: turn on or off depending on the *ambient light,* acquired by a light sensor with range 0% … 100%
   - when the system starts, turned on if ambient light is below 70% and off otherwise.
   - turn on if the ambient light around the car falls below 60% for at least 2 seconds
   - turn off if the ambient light around the car exceeds 70% for at least 3 seconds

1. Test Language





1. Test Language



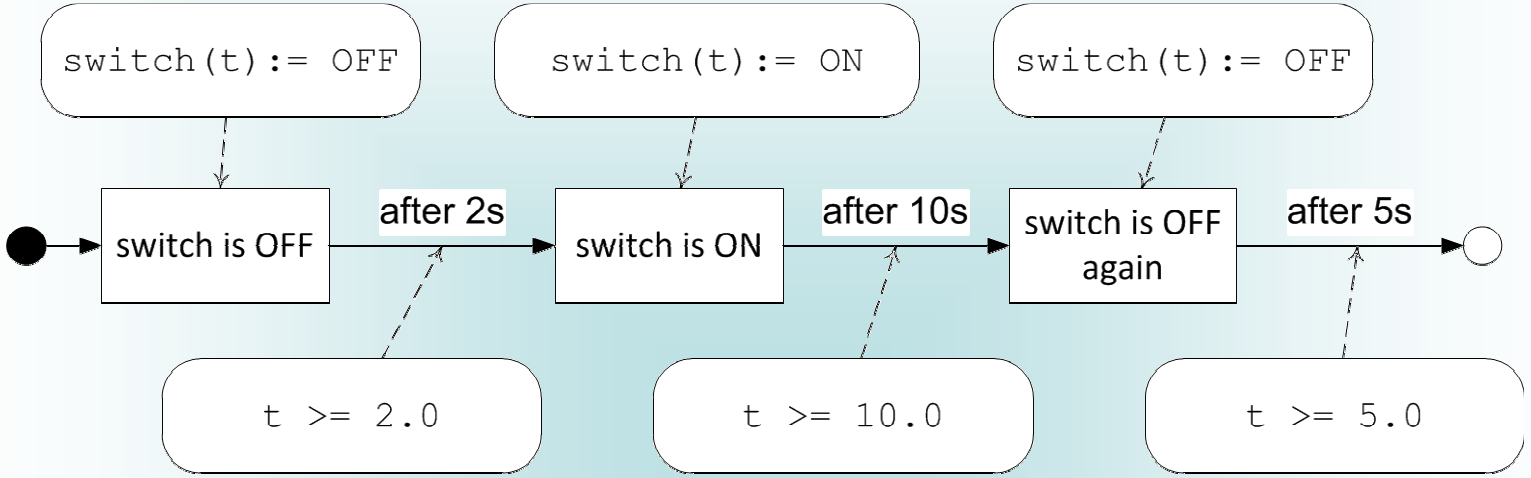
More features (hierarchical state machines, junctions, actions on transitions, …) are available

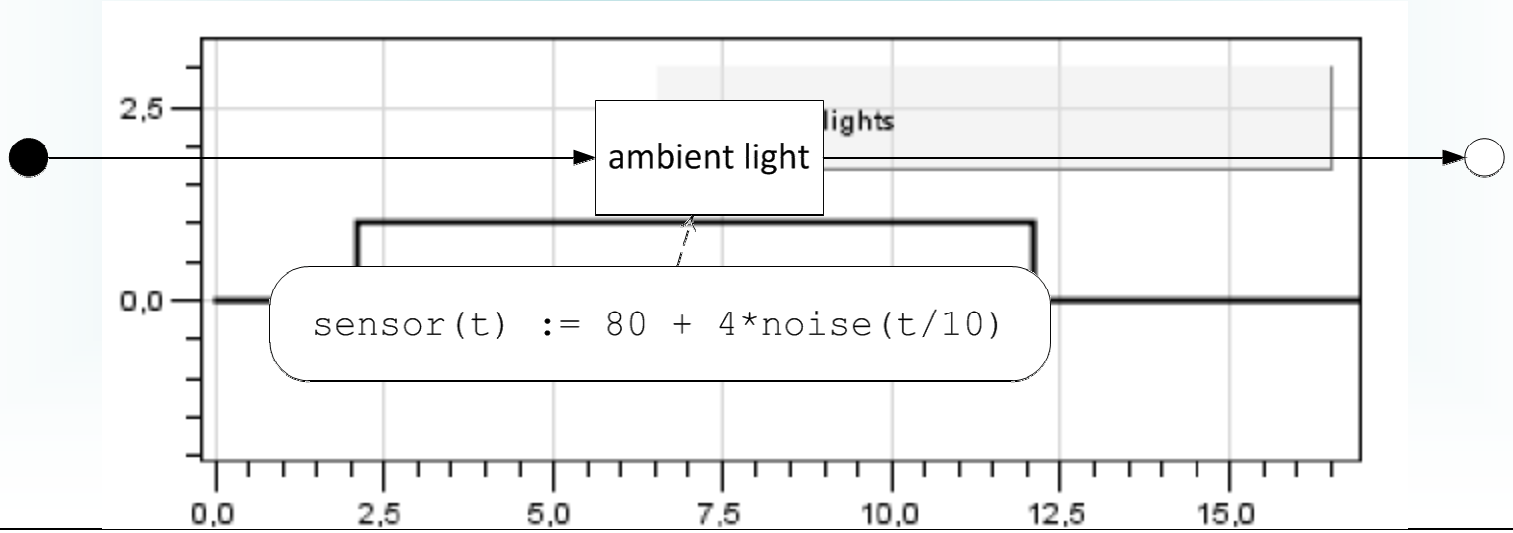
bient light

+ 4*noise

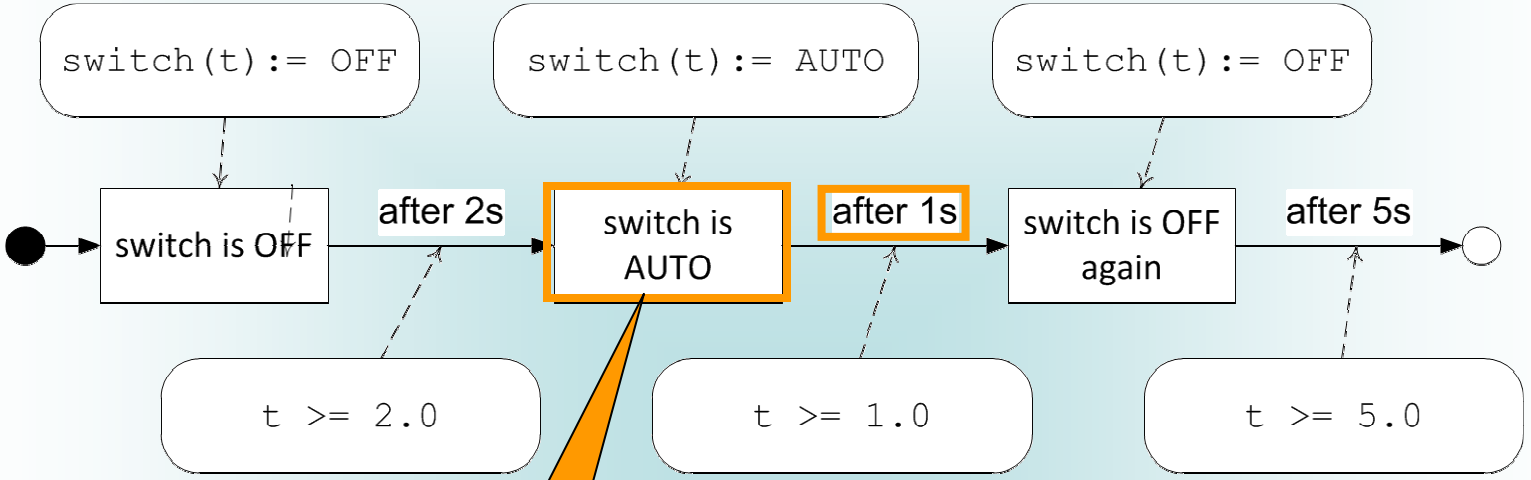
No dependency on software implementation or ECU ➔ can run on an arbitrary platform, given the appropriate adapter

1. Test Language

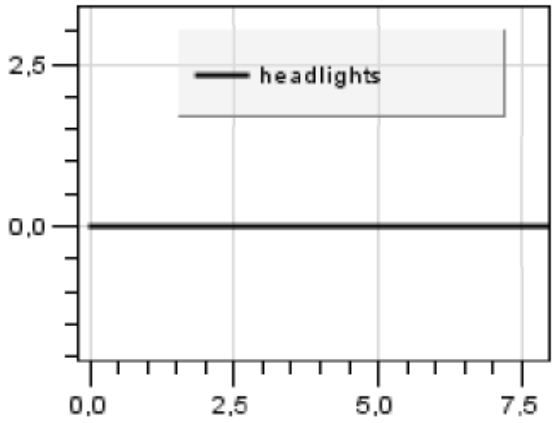




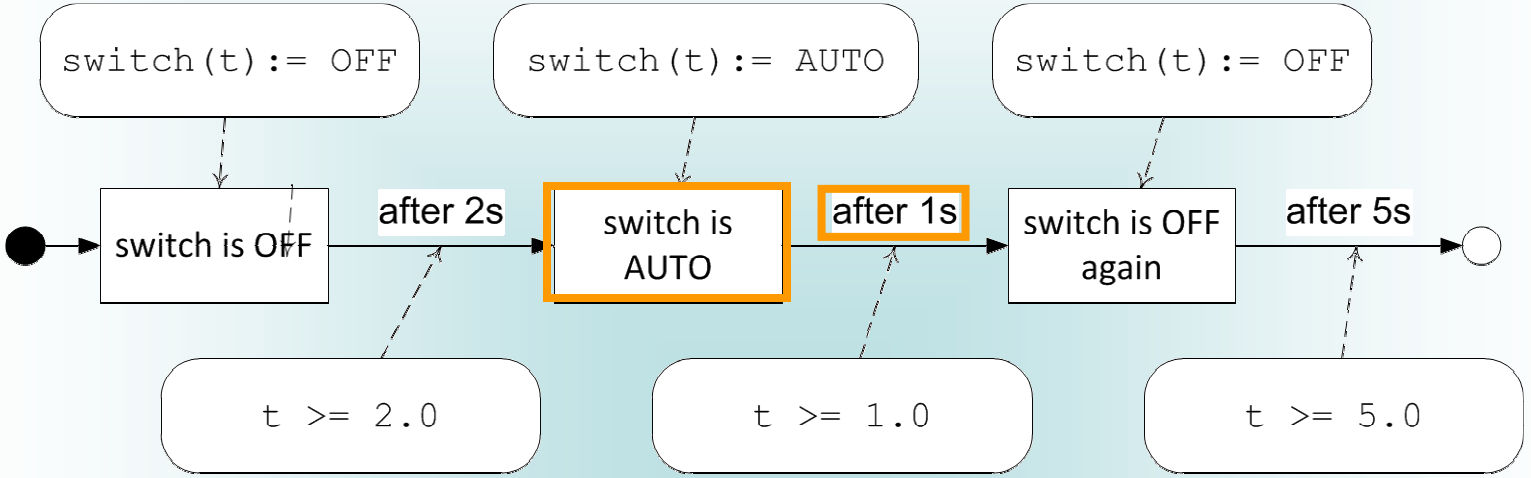
Another Test Case

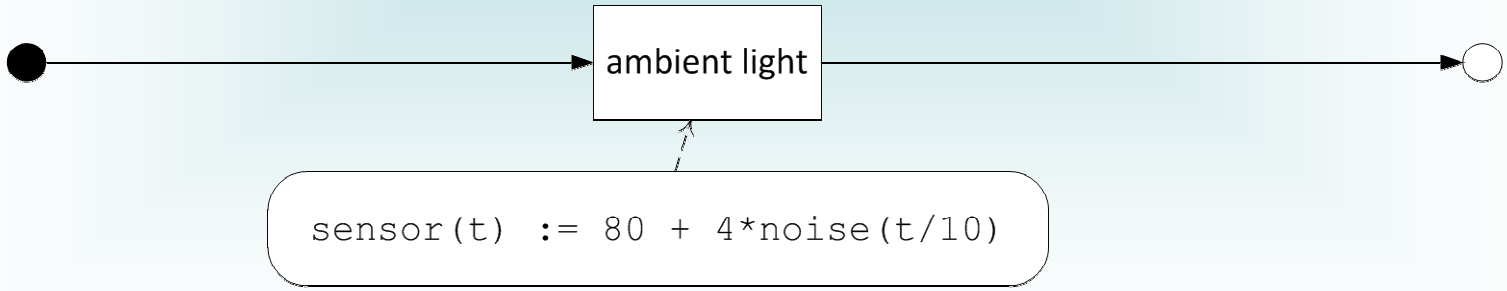


Headlight signal remains off

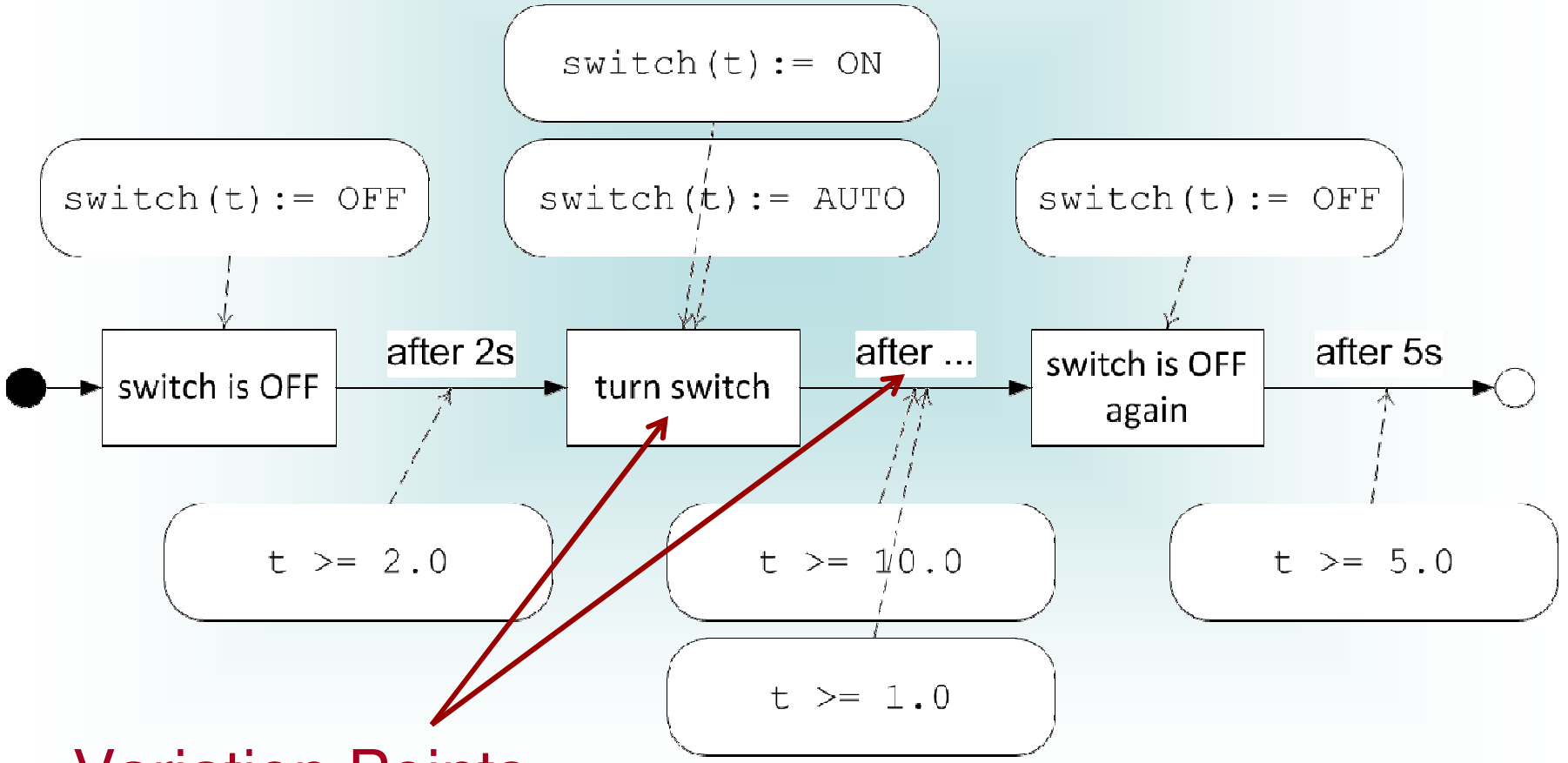


Another Test Case

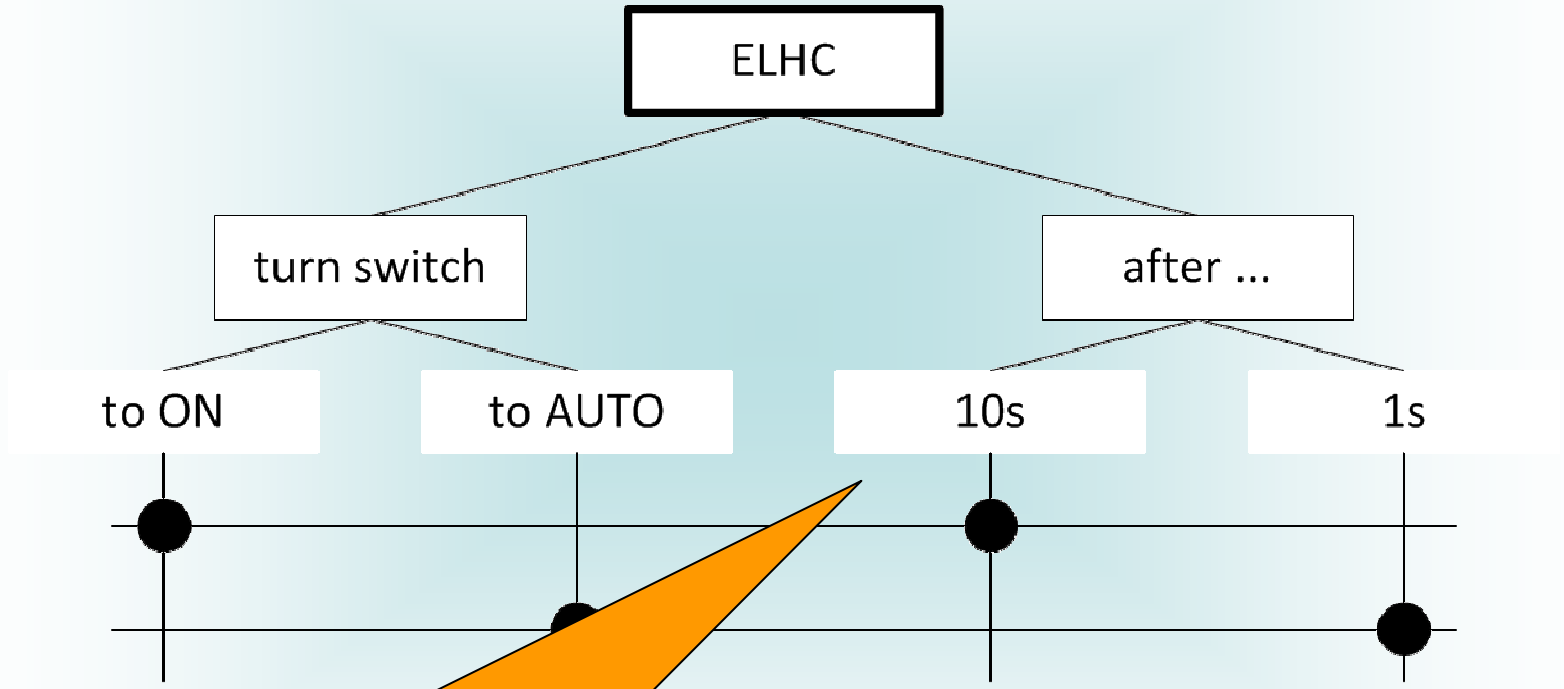




2. Representation of a Test Suite:
all TPT Tests are Integrated into a Single Test Model

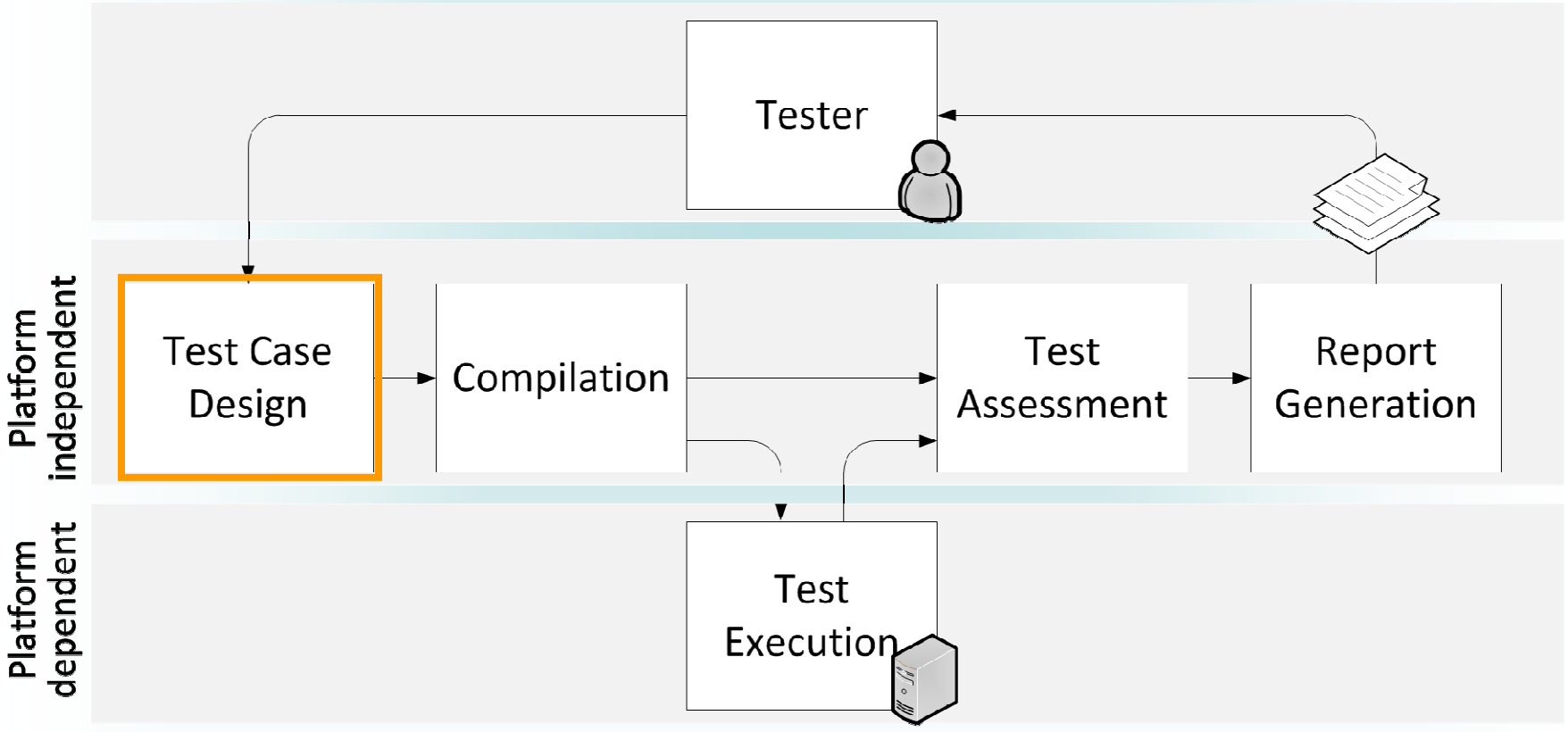


Deriving Test Instances: Classification Trees

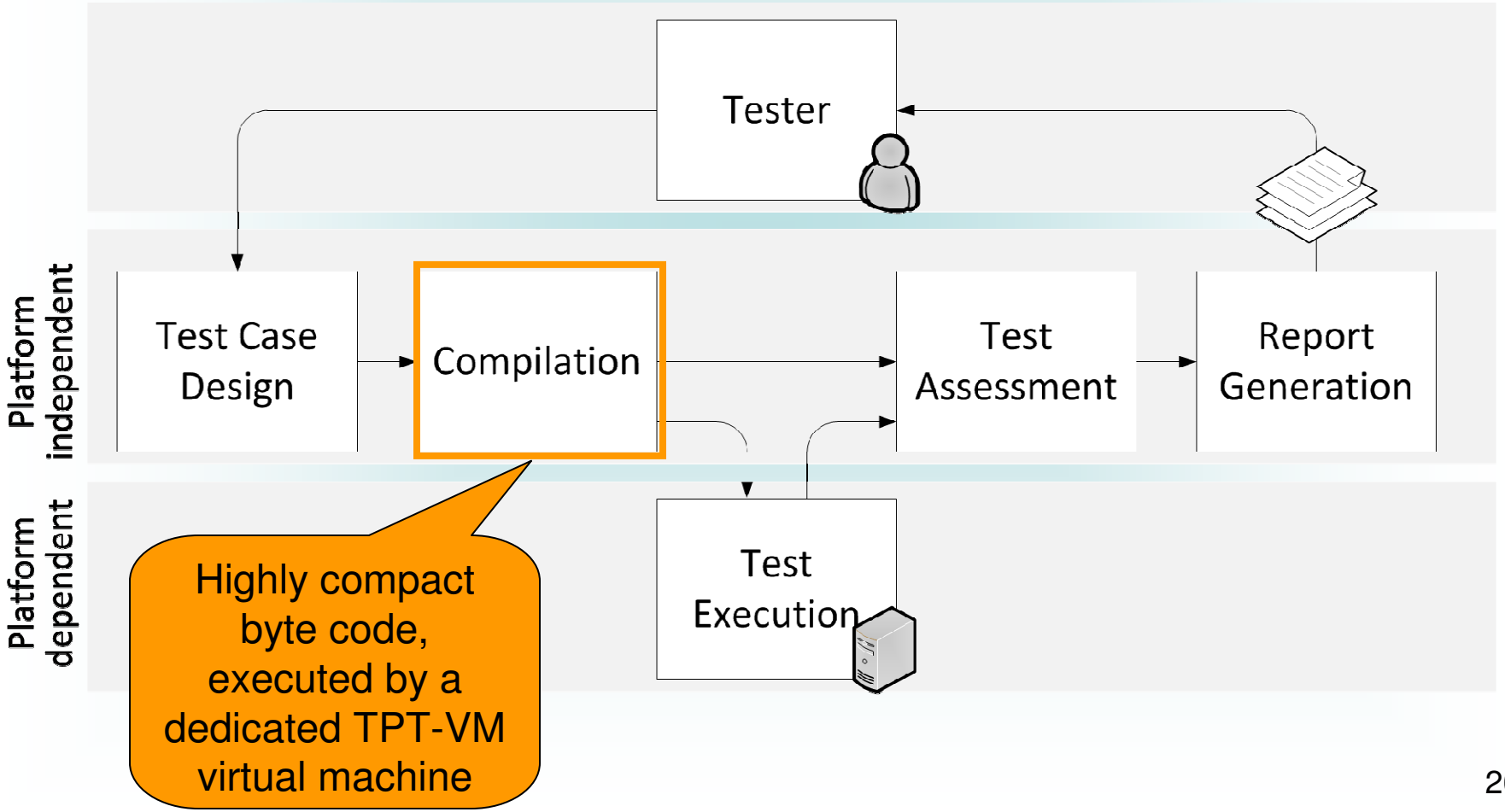


Can express ***logical constraints*** between variants and describe the desired ***coverage*** in the space of all possible combinations

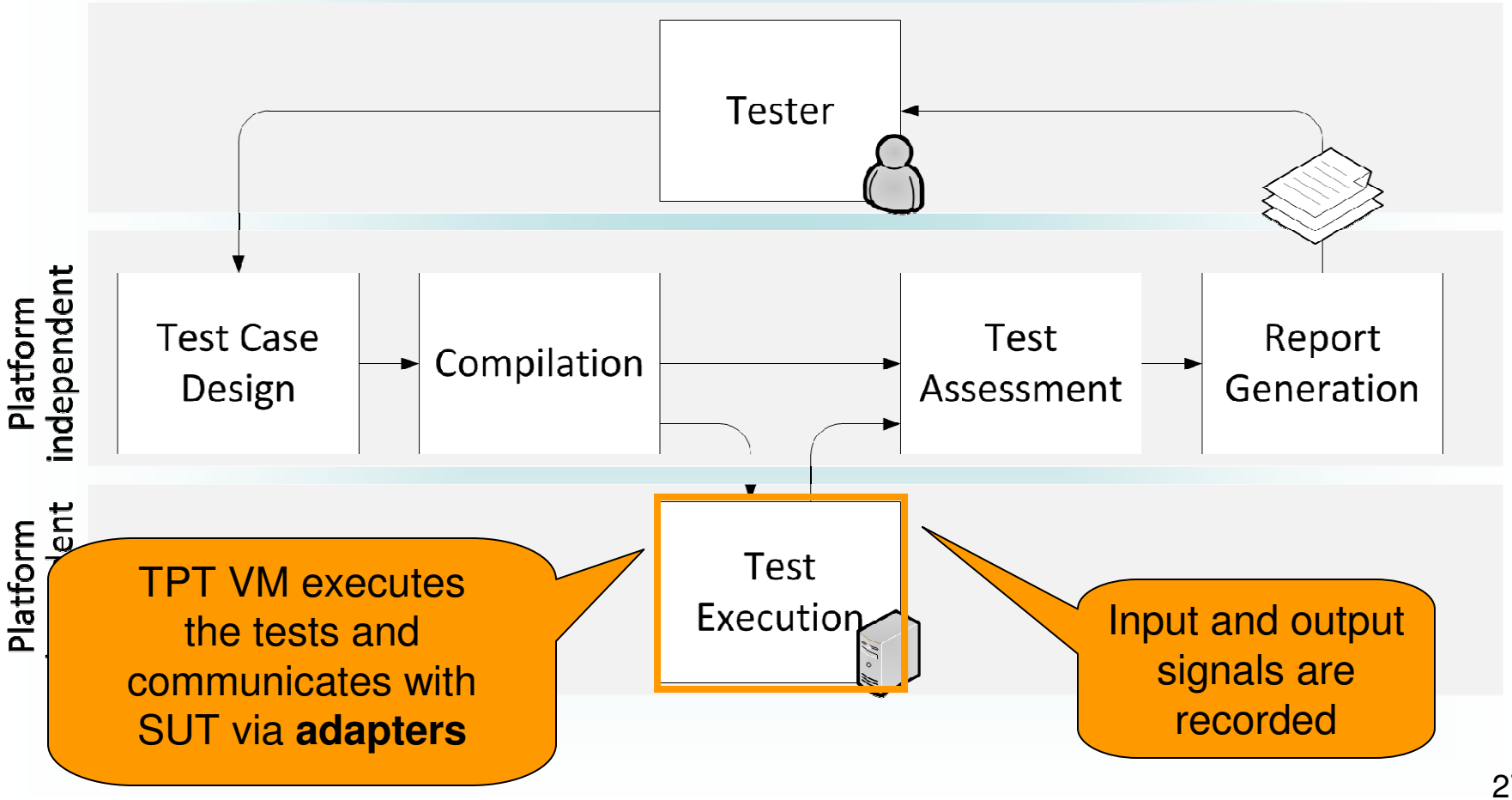
Test Process Using TPT



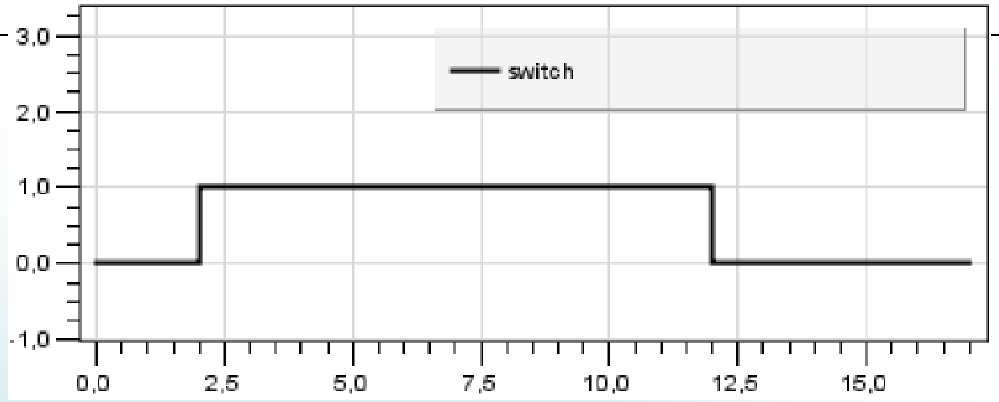
Test Process Using TPT

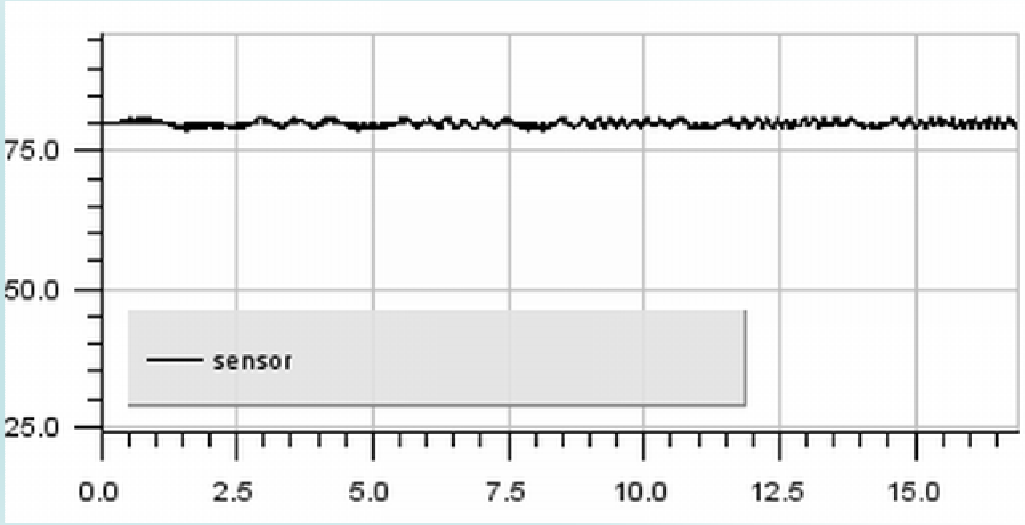


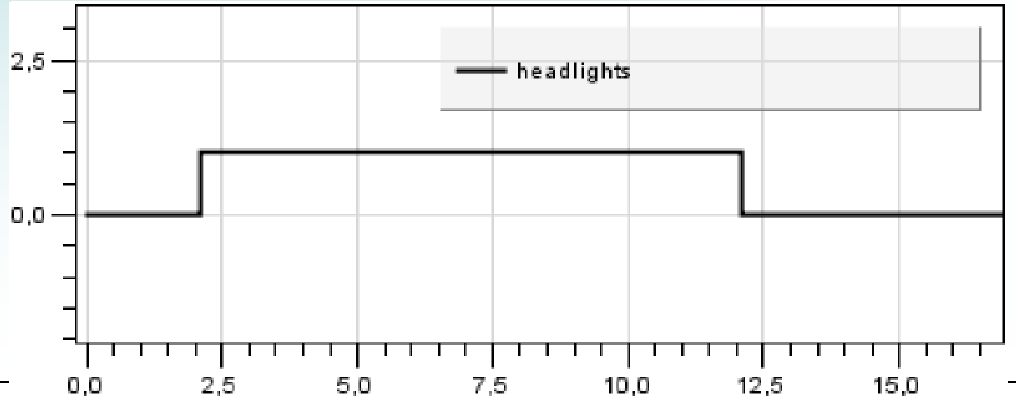
Test Process Using TPT



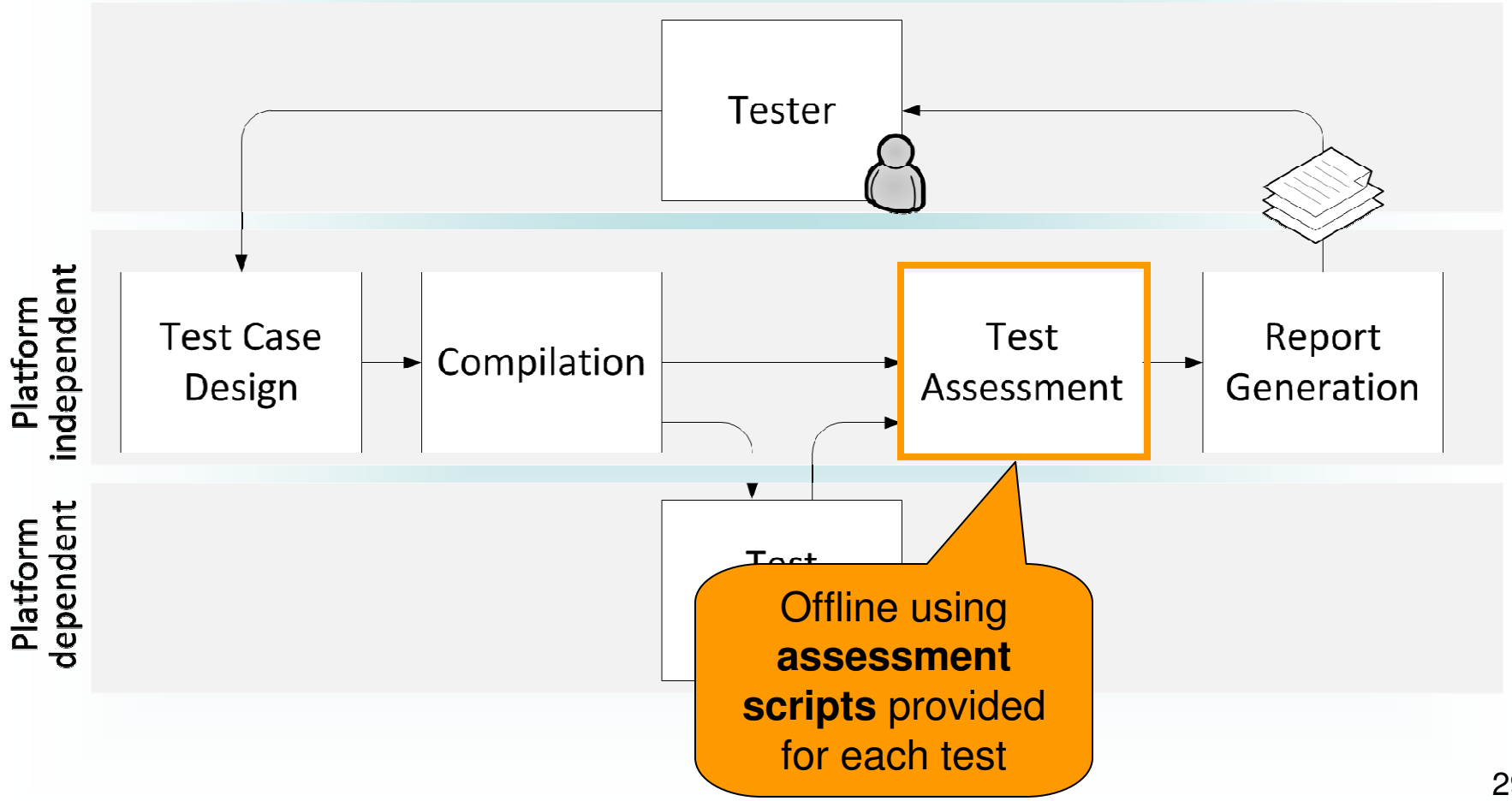
Recorded Test Signals



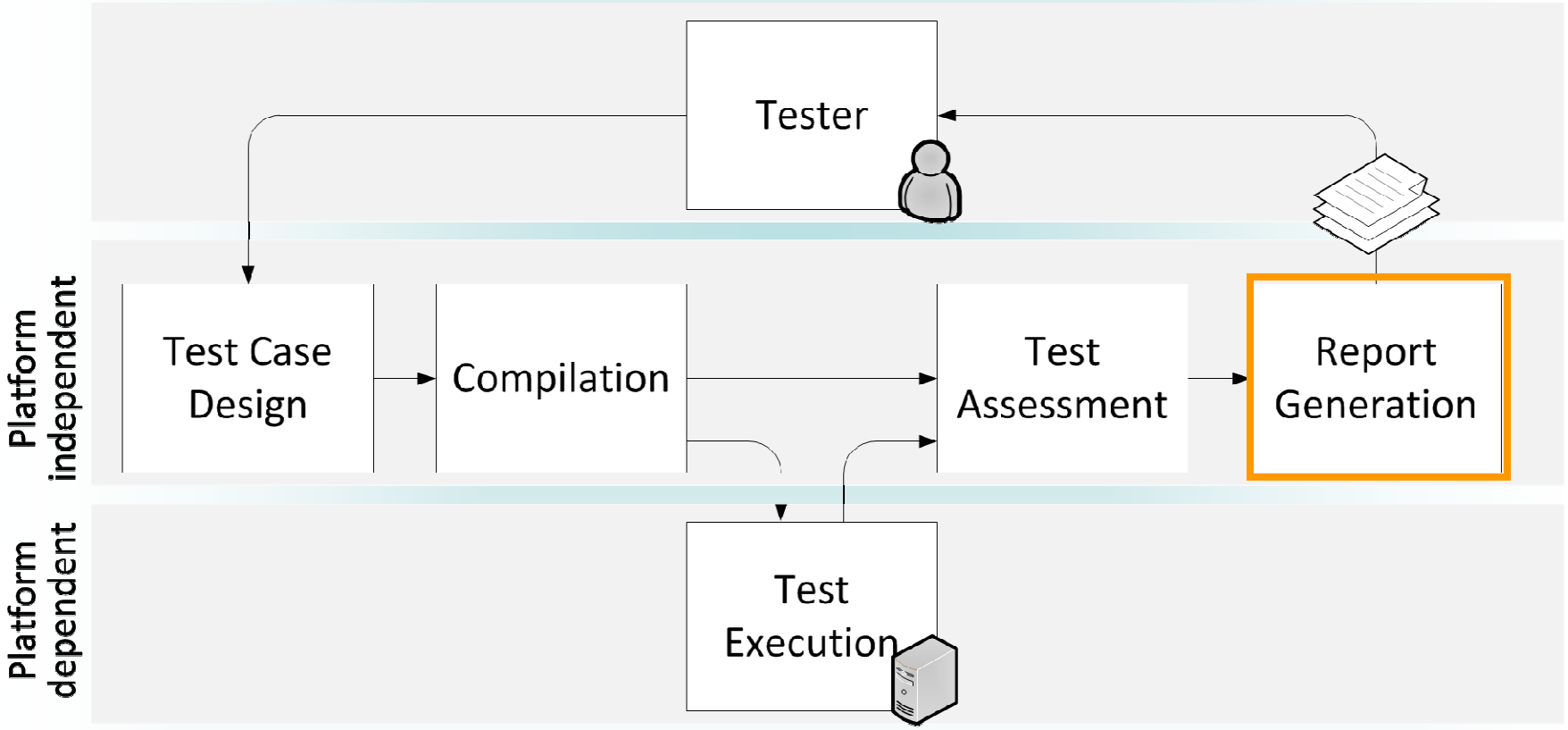




Test Process Using TPT



Test Process Using TPT



Requirements for Testing

✓ *1.* *Automation* to support multiple intermediate releases due to software hardware co-design

✓ *2.* *Portability* between validation stages

✓✗ *3.* *Systematic test case selection* to ensure coverage and lack of redundancies

✓ *4.* *Readability* – "lingua franca" for multiple stakeholders

? *5.* *"Closed loop" behavior* – the execution of the test depends on the behavior of the tested system

✓ *6.* *Real-time behavior*

✓ *7.* *Continuous signals*

Outline

1. Motivation: challenges of in-car software development process
2. Requirements for testing
3. TPT tool for automotive model-based testing
4. Experience report
5. Future work
6. Summary and discussion

Experience

- Initially developed in Daimler Software Technology Research
  - In cooperation with many production vehicle development projects
- Used by all interior production vehicle projects in Daimler

- Used by car manufactures and suppliers as a basis for acceptance tests
- The idea to concentrate all test scenarios into a single model has been proven to be scalable

Future Work

- Interaction with version and configuration management
- Interaction with test management
- Product families
- Model-based testing on integration levels
  - Relationship to component models
  - Utilization of architectural models

Summary

Multidisciplinary Business

2

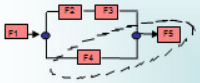
Requirements for Automated Model-based Testing (MBT)

1. Automation, to support multiple intermediate releases due to software hardware co-design
2. Portability between integration levels
3. Systematic test case selection to ensure coverage and lack of redundancies
4. Readability – "lingua franca" for multiple stakeholders
5. Real-time behavior (HiL level)
6. Continues signals

13

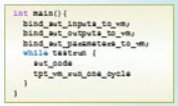
Validation Stages

MiL: Model in the loop

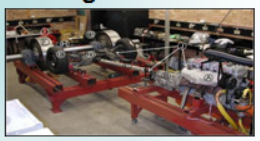


HiL: Hardware in the loop

SiL: Software in the loop



Test Rig

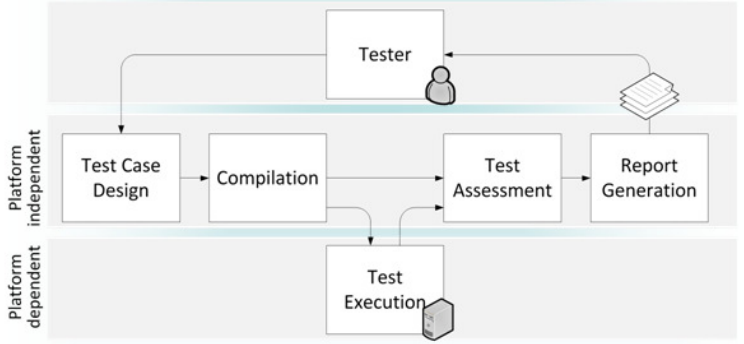


PiL: Processor in the loop

Car

12

Test Process Using TPT



28

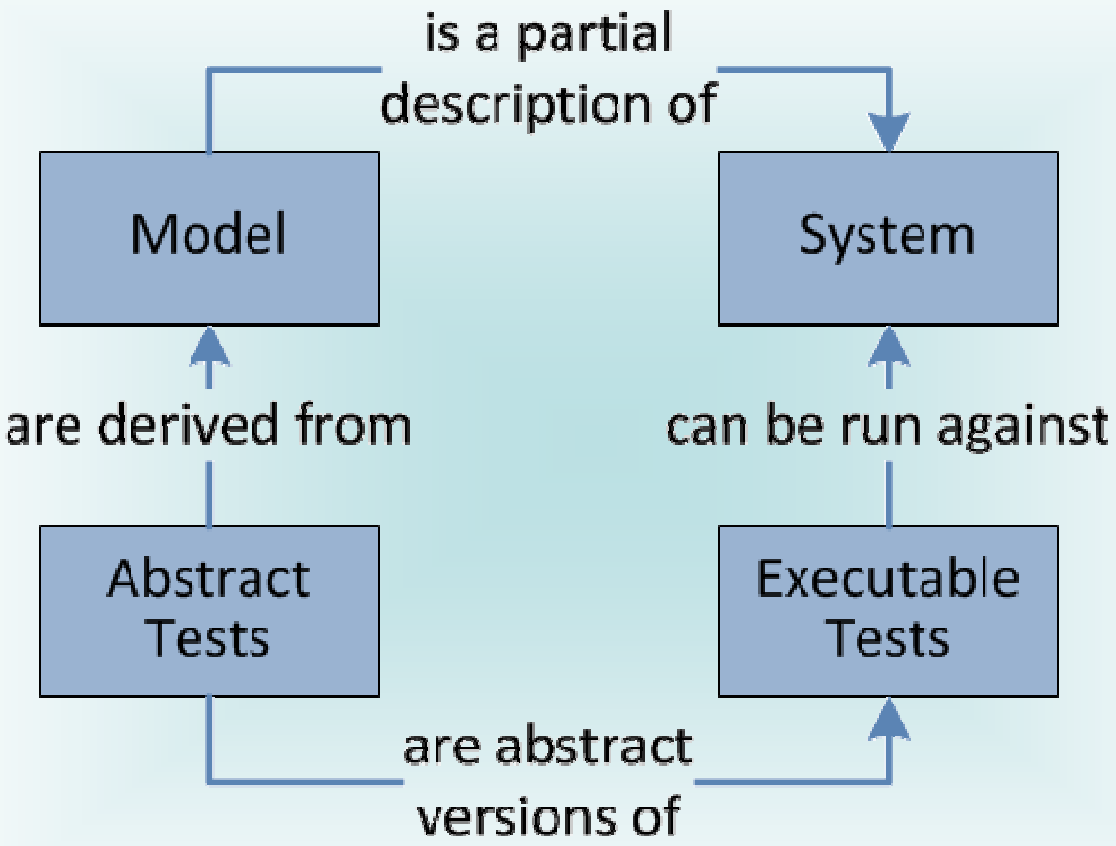
Discussion

- Seems to be very useful in practice
- Some important details are omitted, e.g.,
  - How to design and manage assessment scripts?
  - How difficult is it to build adapters?
- Would be interesting to see how scalable the representation of “all test cases in one model” is
- No related work is discussed
  - Most of the provided references are not cited in the text
- What makes the tool specific to automotive?
  - Can it be used in other domains, e.g., A&D

Thank You!

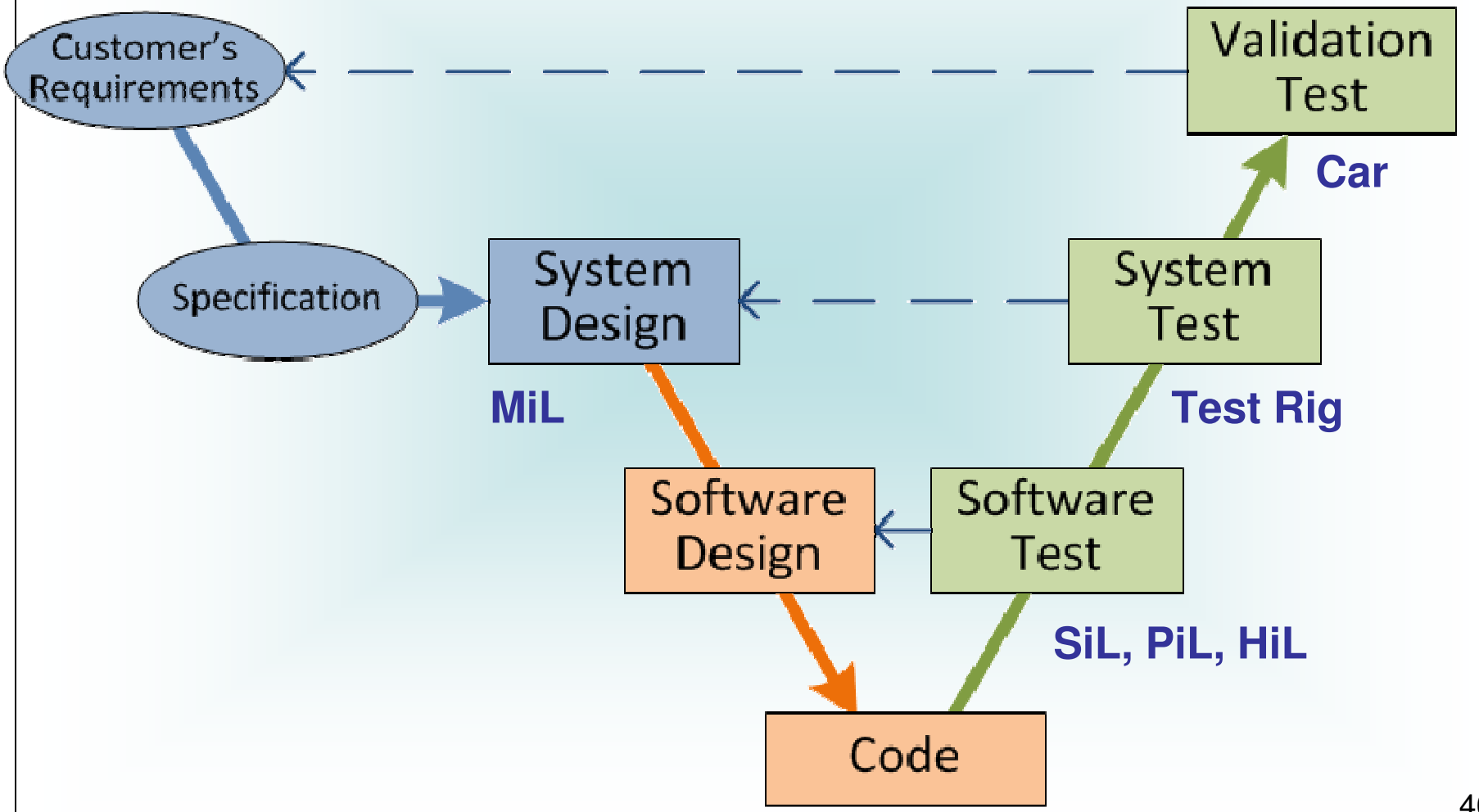
Backup

What is Model-Based Testing?



"In the automotive industry [the term MBT] is normally used to describe all testing activities in the context of model-based development projects"

The V Development Model



Deutsch English

PIKETEC
software engineering

Home Company Products Services Download Support Contact Career

Company

- Introduction
- Founder
- ISO 9001:2008
- Partner
- Publications

Andreas Krämer, Jens Lüdemann, Eckard Bringmann

The PikeTec founders Dr. Eckard Bringmann, Andreas Krämer und Dr. Jens Lüdemann share a ten year working experience at DaimlerChrysler. Their research focussed on software technology and tool development for the software development of technical systems.

With their work they supported the development group at DaimlerChrysler AG especially with software development by defining and designing development processes. This support included also the technical and scientific steering of development projects in the automotive divisions passenger compartment, driving dynamics, telematics and powertrain.