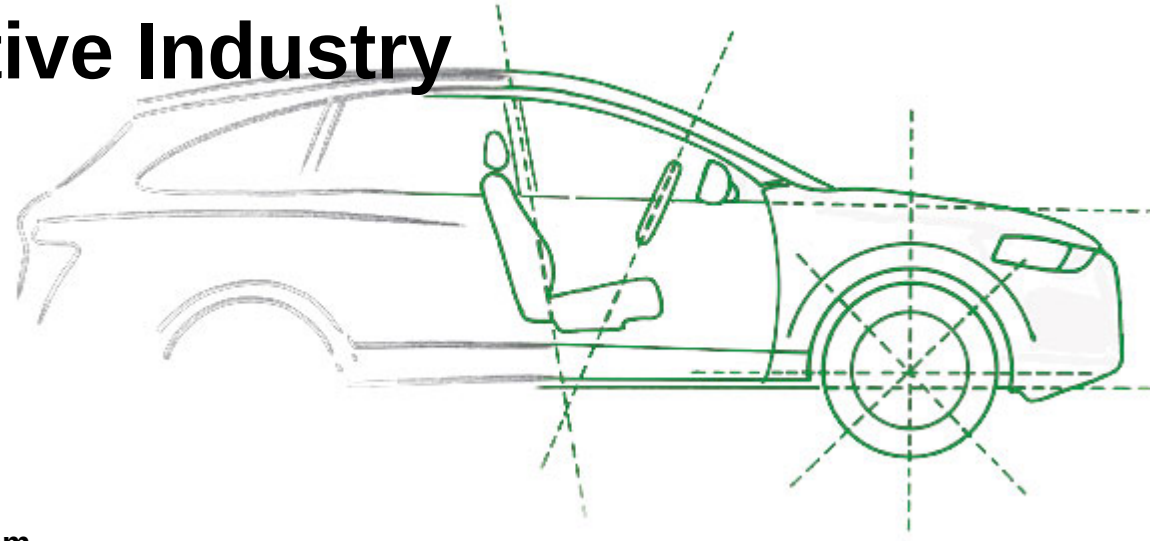
berner & mattner
optimizing your development

Black-Box Test Case Design Techniques in the Automotive Industry



Dr. Joachim Wegener

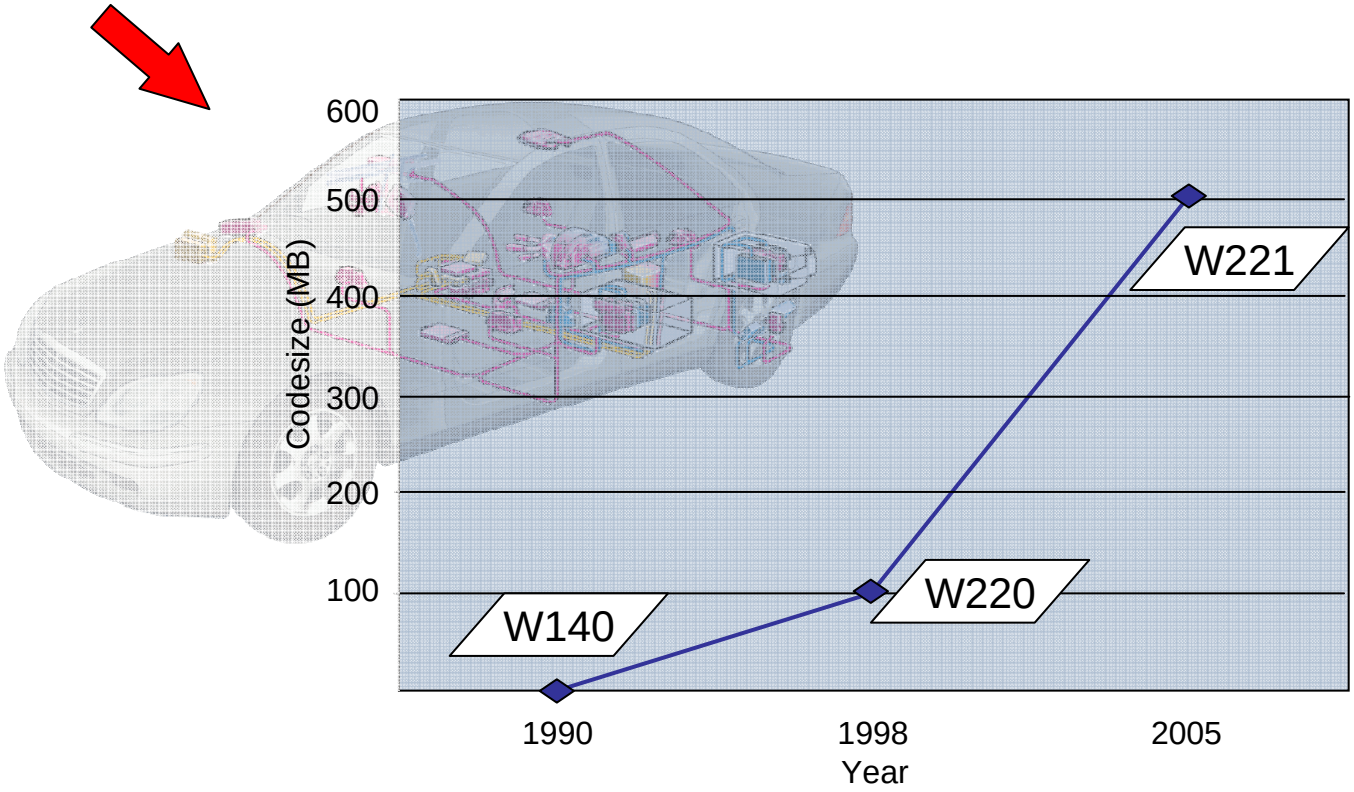
joachim.wegener@berner-mattner.com

Overview

- Motivation
- Test Case Design Requirements
- Classification-Tree Method
- Model-based Testing
- Time-Partition Testing
- Evolutionary Testing
- Conclusion

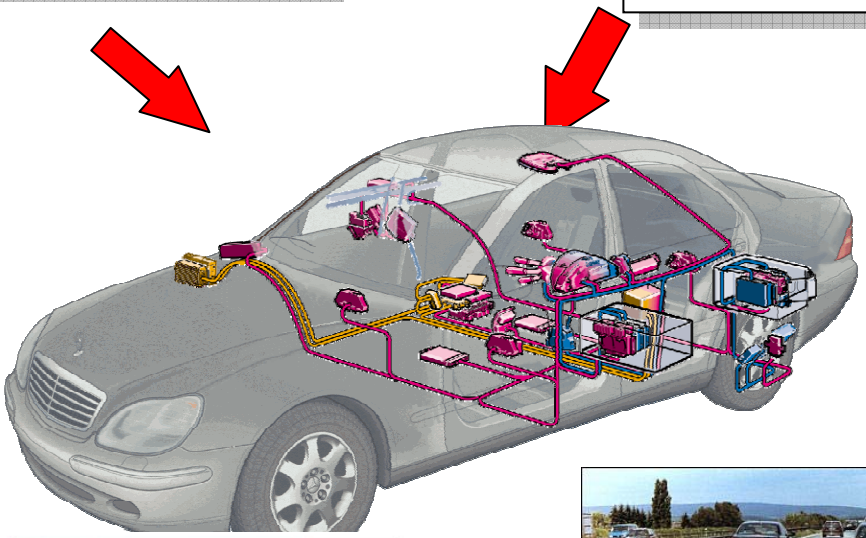
Challenges in Automotive System Development

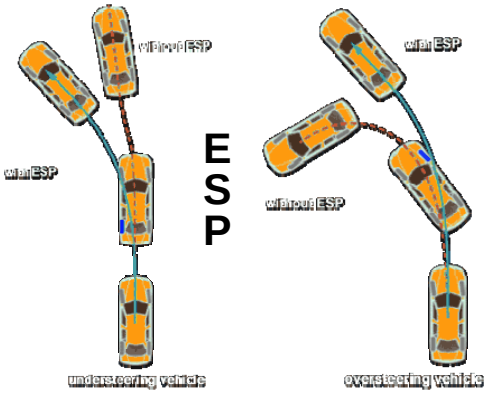
- Increasing amount of software in products in almost all areas



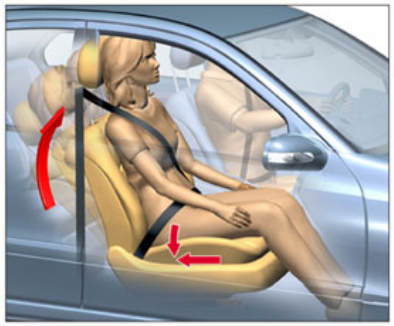
Challenges in Automotive System Development

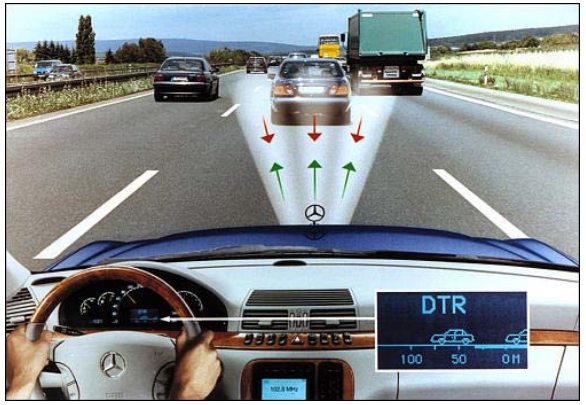
- Increasing amount of software in products in almost all areas
- Software as differentiating factor for competitive advantage





PRE-Safe

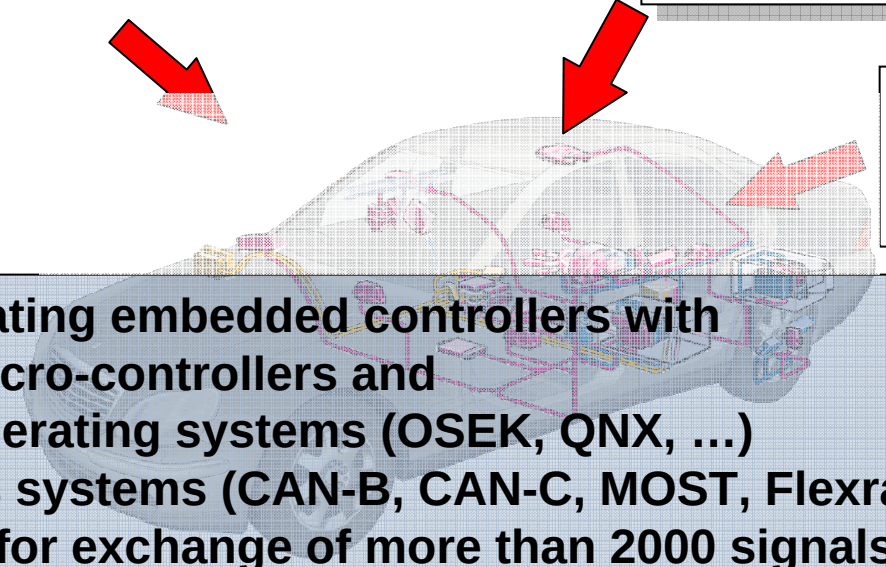




Distronic

Challenges in Automotive System Development

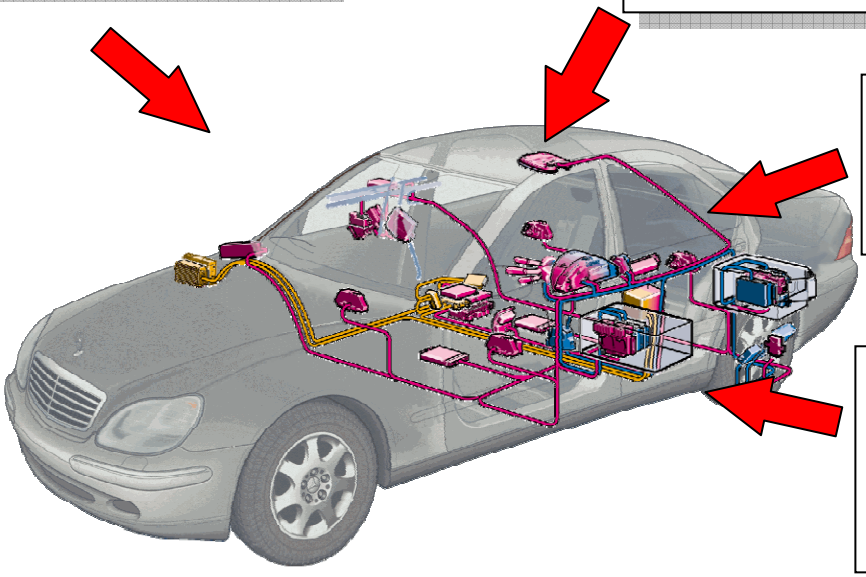
- Increasing amount of software in products in almost all areas
- Software as differentiating factor for competitive advantage
- Increasing complexity of systems



50-70 communicating embedded controllers with
  - **different micro-controllers and**
  - **different operating systems (OSEK, QNX, …)**
  - **several bus systems (CAN-B, CAN-C, MOST, Flexray, …) with different topologies for exchange of more than 2000 signals and messages**
  - **strong interactions**
- **development and production by a large number of different suppliers**
- **electrical and optical cabling (length ~2.5 km)**
- **up to 150 electronic motors**
- **more than 10,000,000 lines of software code**

Challenges in Automotive System Development

- Increasing amount of software in products in almost all areas
- Software as differentiating factor for competitive advantage



- Increasing complexity of systems
- Increasing significance of software in safety relevant areas

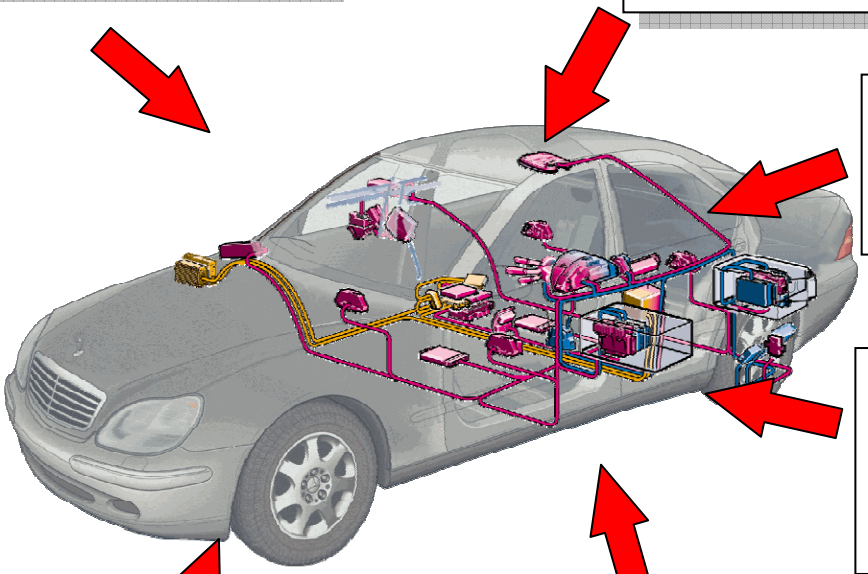
Challenges in Automotive System Development

- Increasing amount of software in products in almost all areas
- Software as differentiating factor for competitive advantage

- Increasing complexity of systems
- Increasing significance of software in safety relevant areas
- High requirements for software based systems due to norms, standards and legal regulations

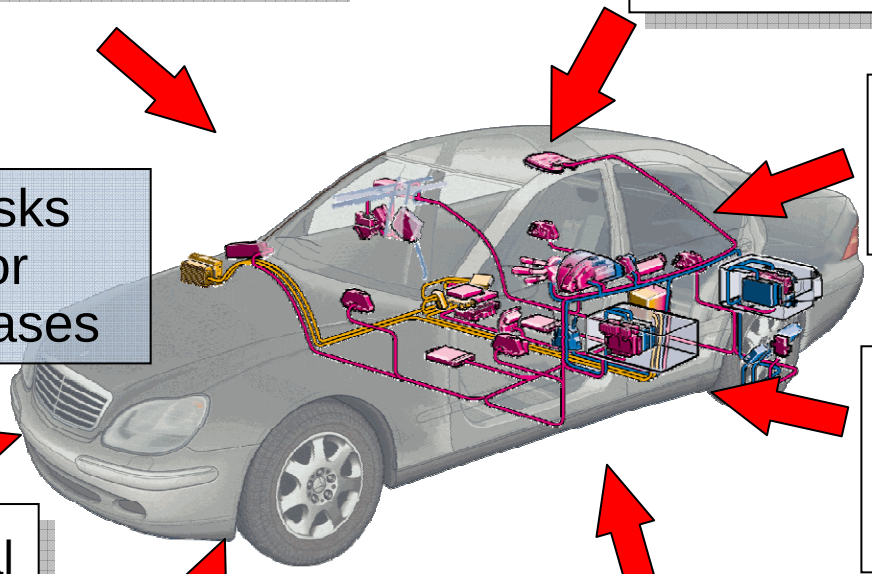
Challenges in Automotive System Development

- Increasing amount of software in products in almost all areas
- Software as differentiating factor for competitive advantage
- Increasing complexity of systems
- Increasing significance of software in safety relevant areas
- High requirements for software based systems due to norms, standards and legal regulations
- Large number of variants



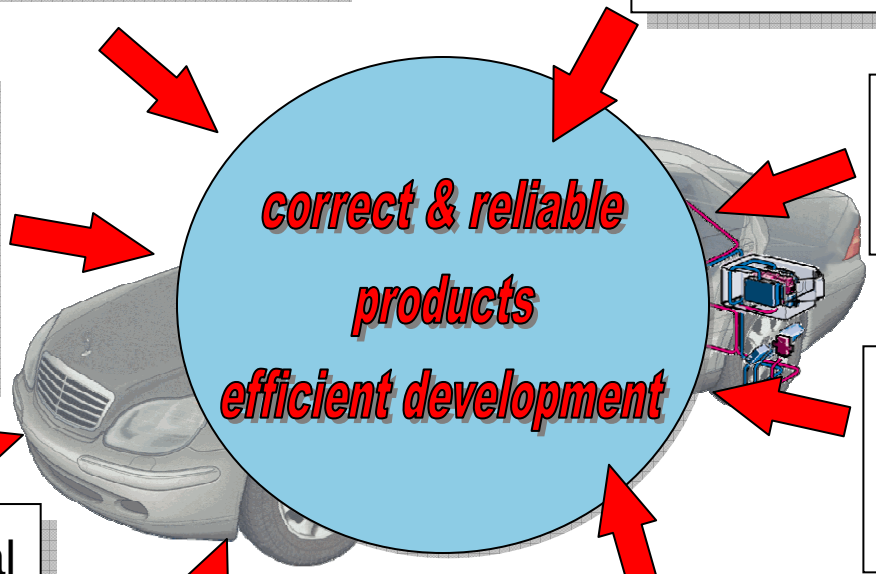
Challenges in Automotive System Development

- Increasing amount of software in products in almost all areas
- Software as differentiating factor for competitive advantage
- Increasing complexity of systems
- High economic risks due to call-back or product liability cases
- Increasing significance of software in safety relevant areas
- High consequential costs due to faults
- Large number of variants
- High requirements for software based systems due to norms, standards and legal regulations



Challenges in Automotive System Development

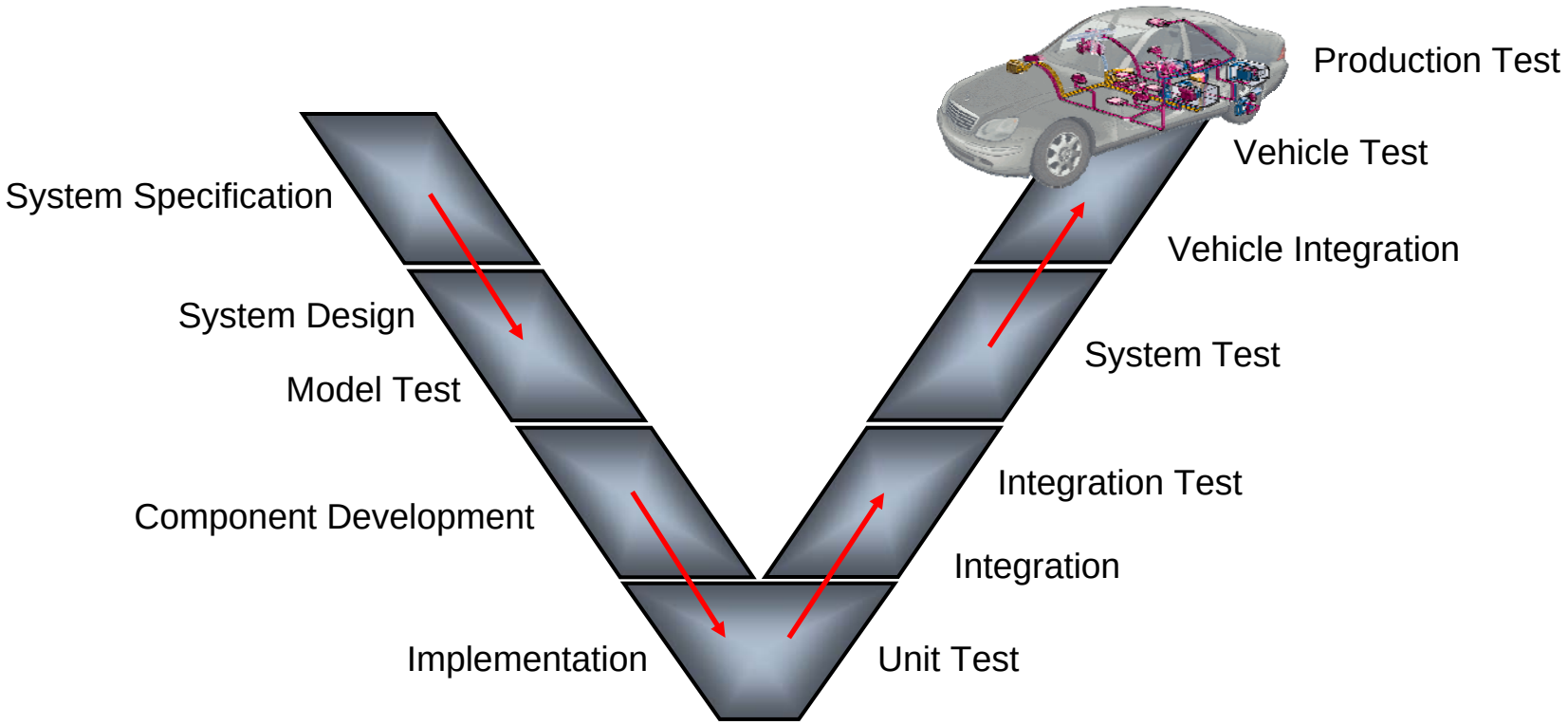
- Increasing amount of software in products in almost all areas
- Software as differentiating factor for competitive advantage
- Cost-effective development and time-to-market crucial for competitiveness
- Increasing complexity of systems
- Increasing significance of software in safety relevant areas
- High consequential costs due to faults
- Large number of variants
- High requirements for software based systems due to norms, standards and legal regulations



Challenges in Automotive System Development

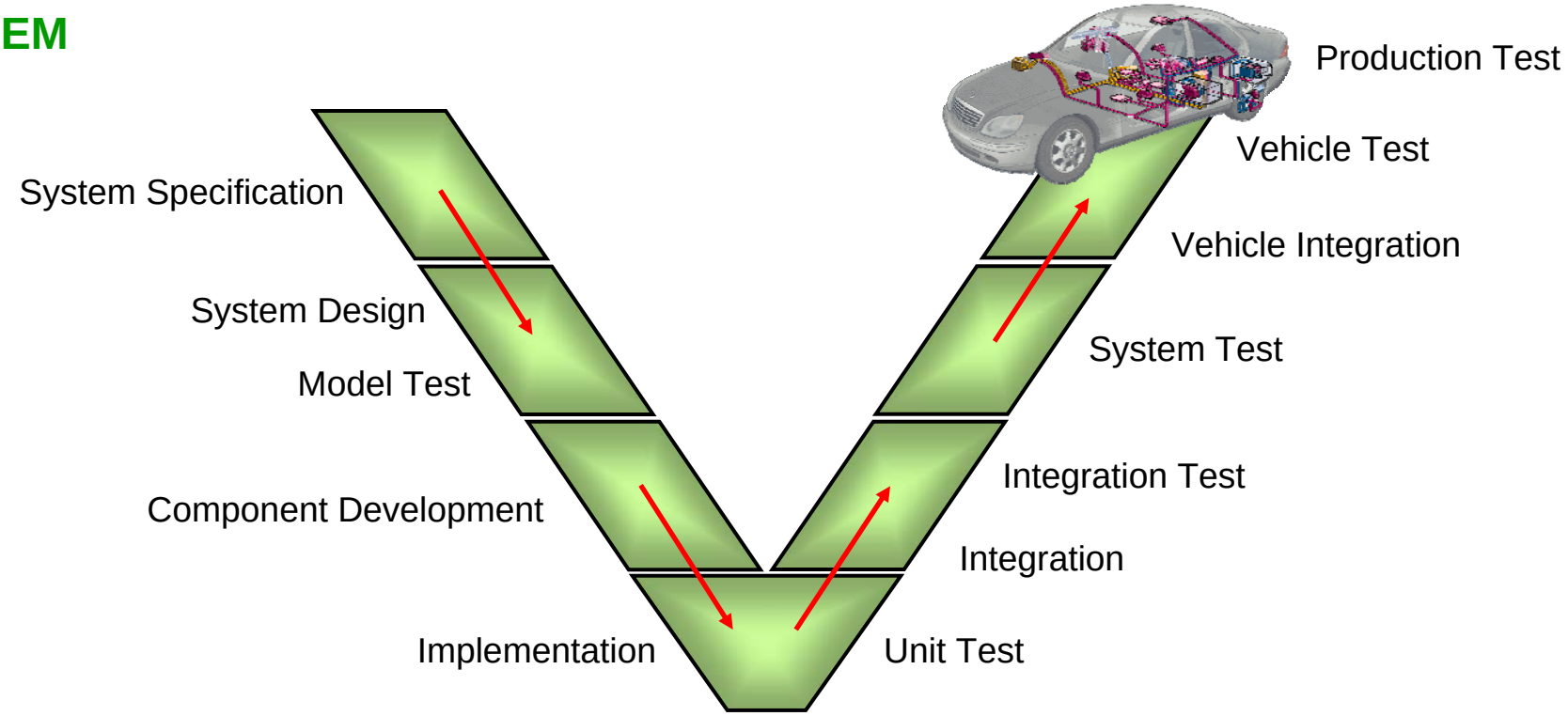
Demonstration of current systems

System Development

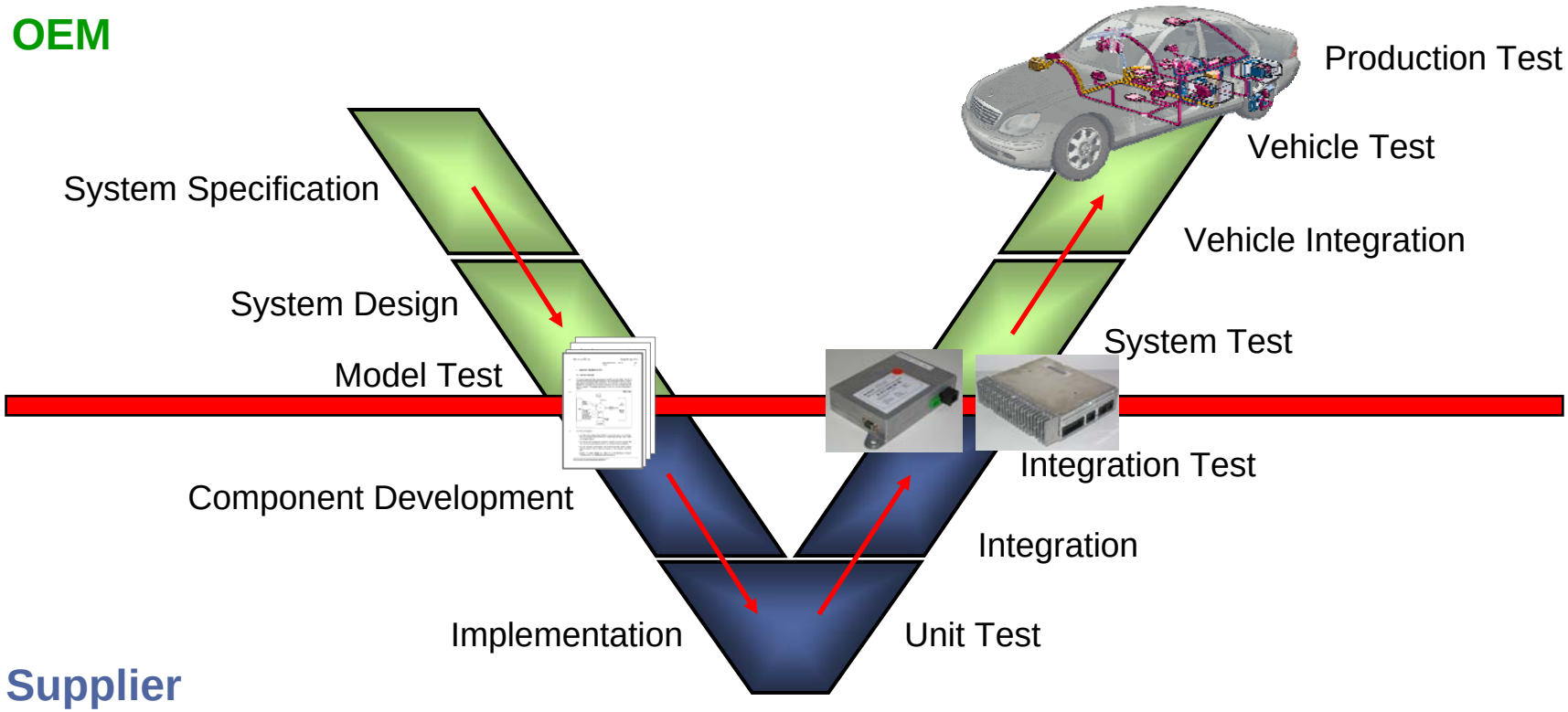


System Development

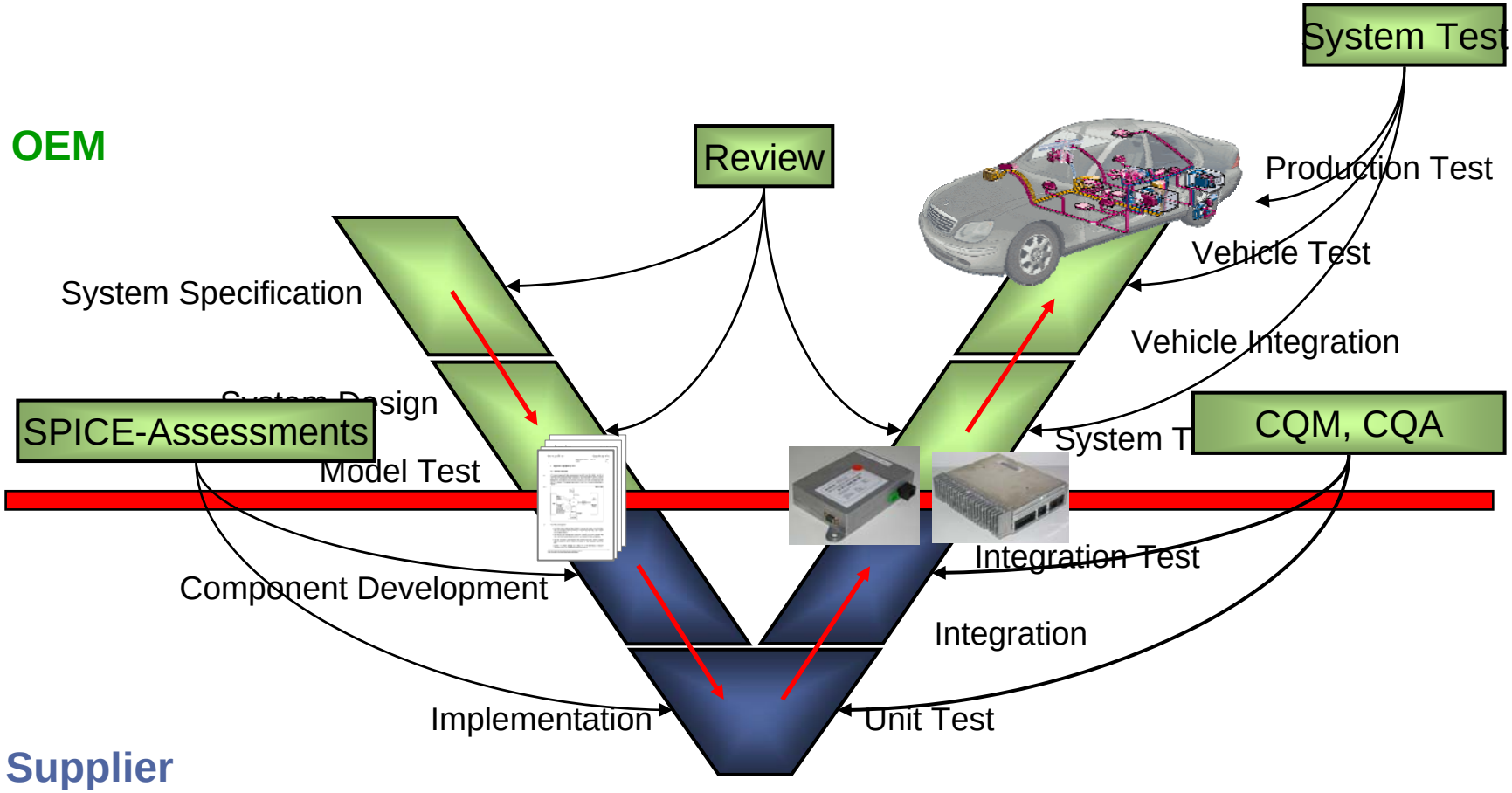
OEM



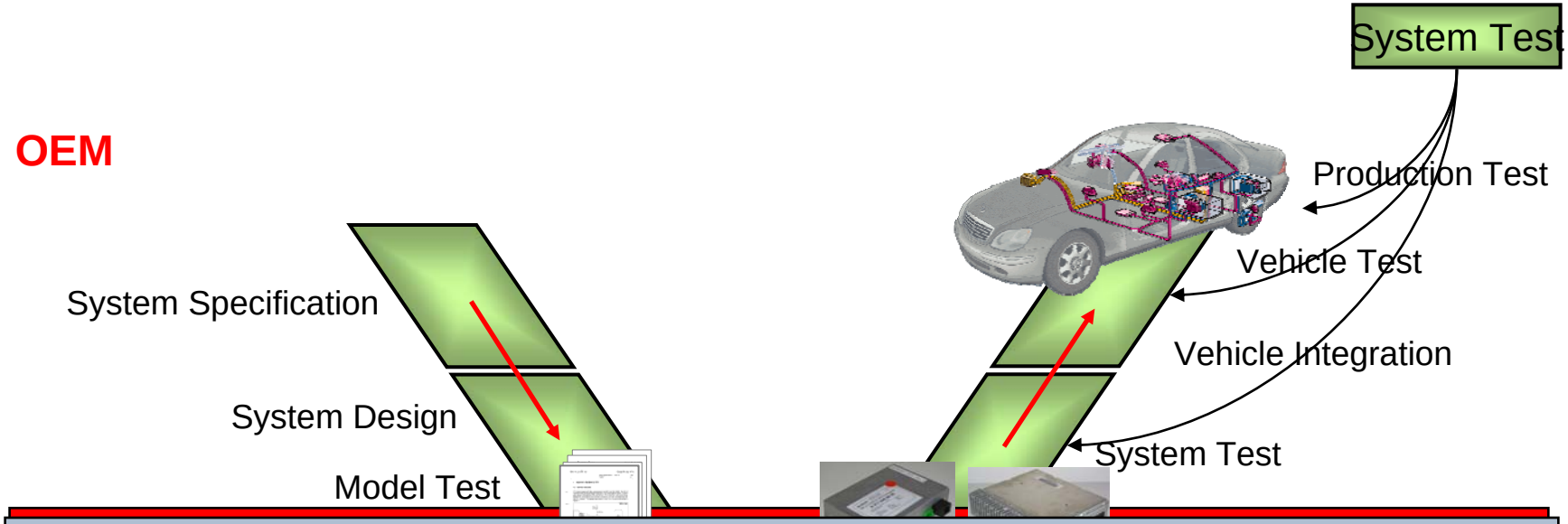
Predominat System Development



Predominat System Development



Predominat System Development



- Testing is the most important analytical quality assurance measure
- Testing is a very significant cost factor
- Pure Black-Box testing
  - no insight into implementation details
  - no structural coverage information
  - no log analysis
- Complex test objects
- Primary goal: Error detection

Test Case Design

OEM Requirements on Test Case Design Methods

- support of functional (black-box) testing
- systematic, stepwise procedure
- abstraction from concrete test data
- easy to use, easy to learn (suitable for non-programmers)
- test case descriptions using natural language and graphical representations (formal parts hidden)
- tool support, high degree of automation
- quality metrics, coverage of test relevant aspects
- comprehensive test documentation

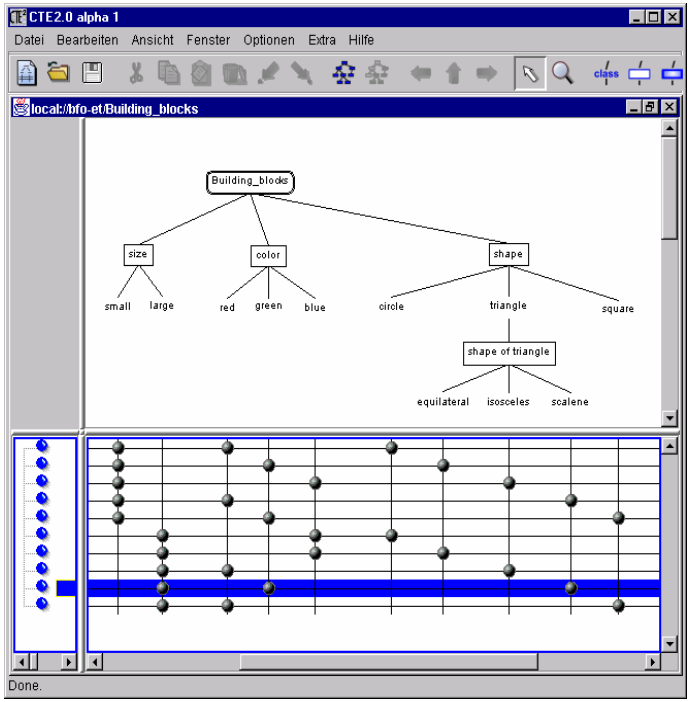
➔ traditional black-box testing techniques, such as equivalence partitioning, boundary-value analysis are not fullfilling these demands

Classification-Tree Method

efficient functional test method

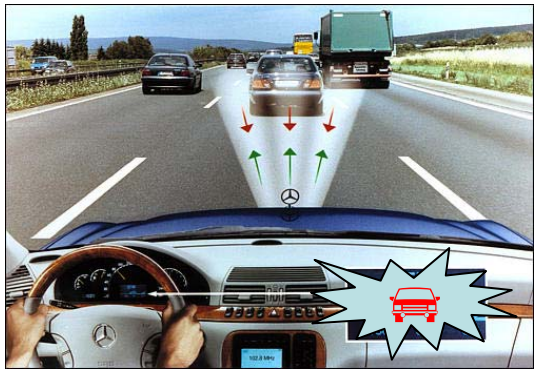
with

- systematic stepwise procedure
- easy to understand
- graphical notation with compact representation of the overall test
- extensive test documentation
- tool support (CTE XL)
- widely used in automotive industry and other domains



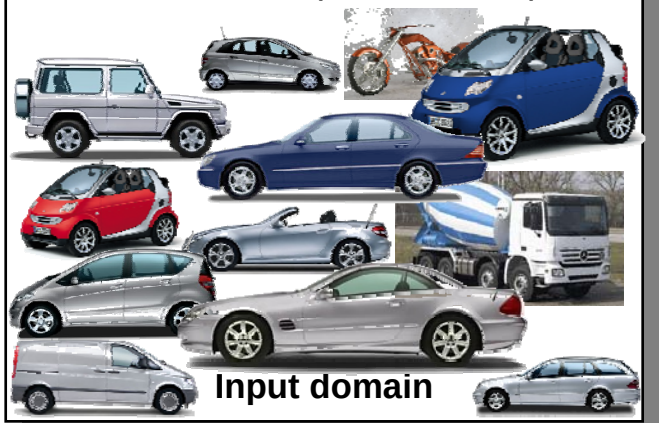
Classification-Tree Method

SUT: radar-based distance warning system

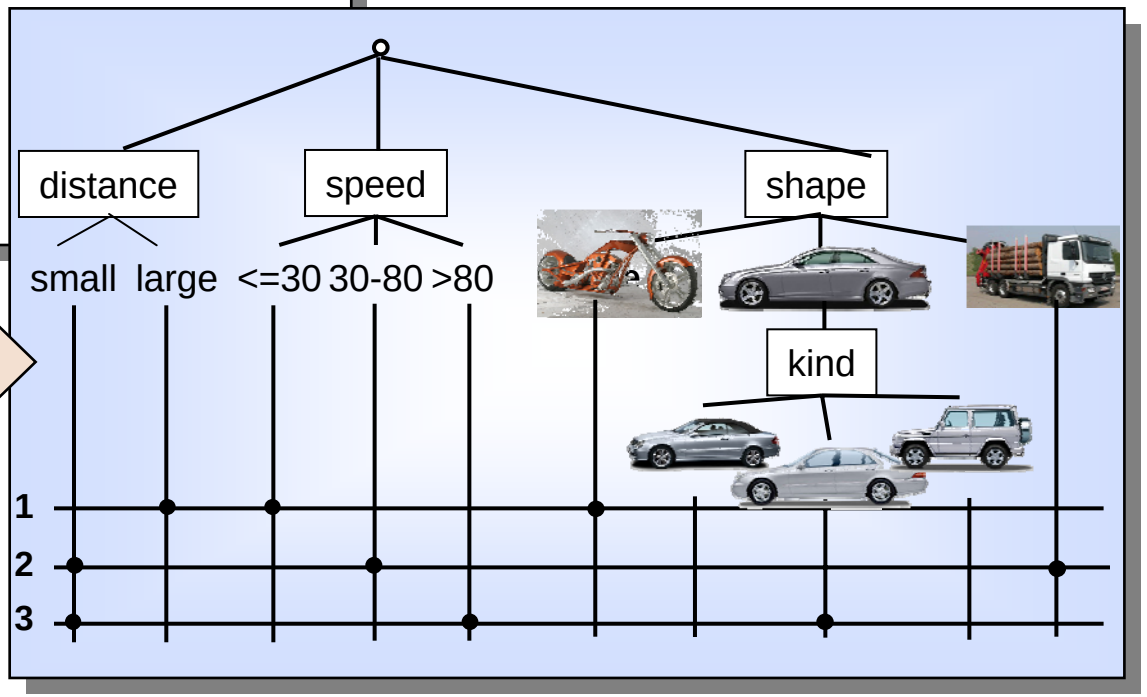


Test relevant aspects

distance speed shape

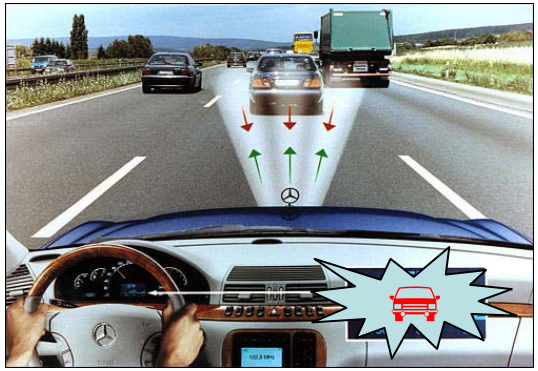


Input domain

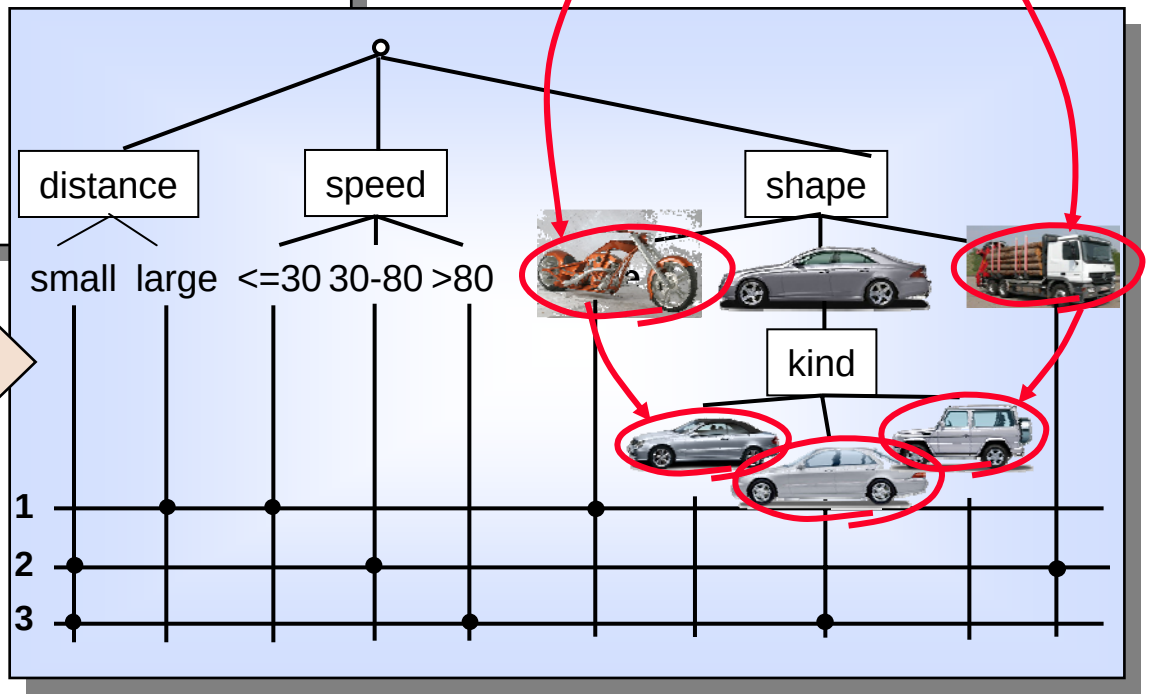


Classification-Tree Method

SUT: radar-based distance warning system



Minimum number of test cases = 5
five disjoint classes for the classification “shape”

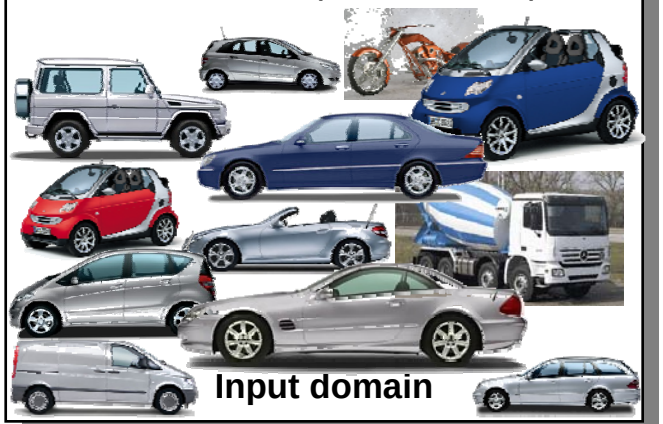


Test relevant aspects


distance

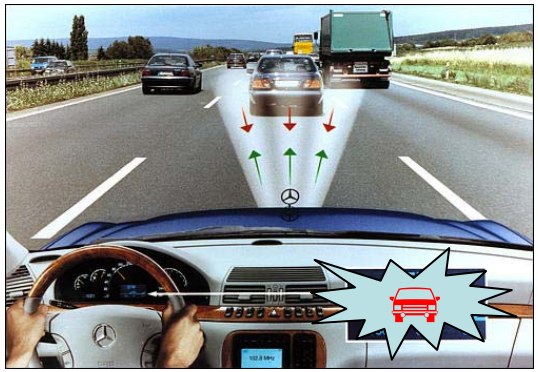

speed


shape



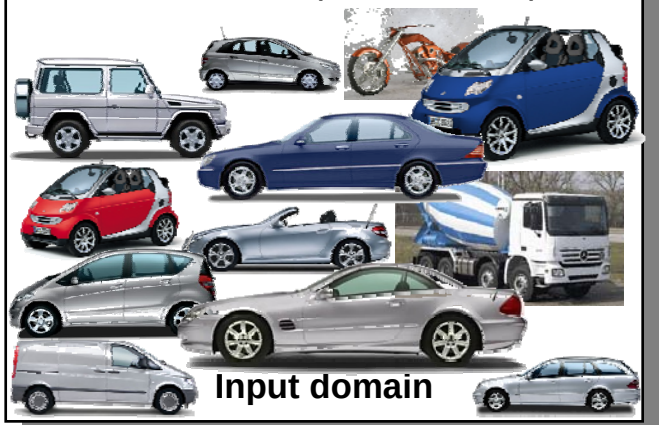
Classification-Tree Method

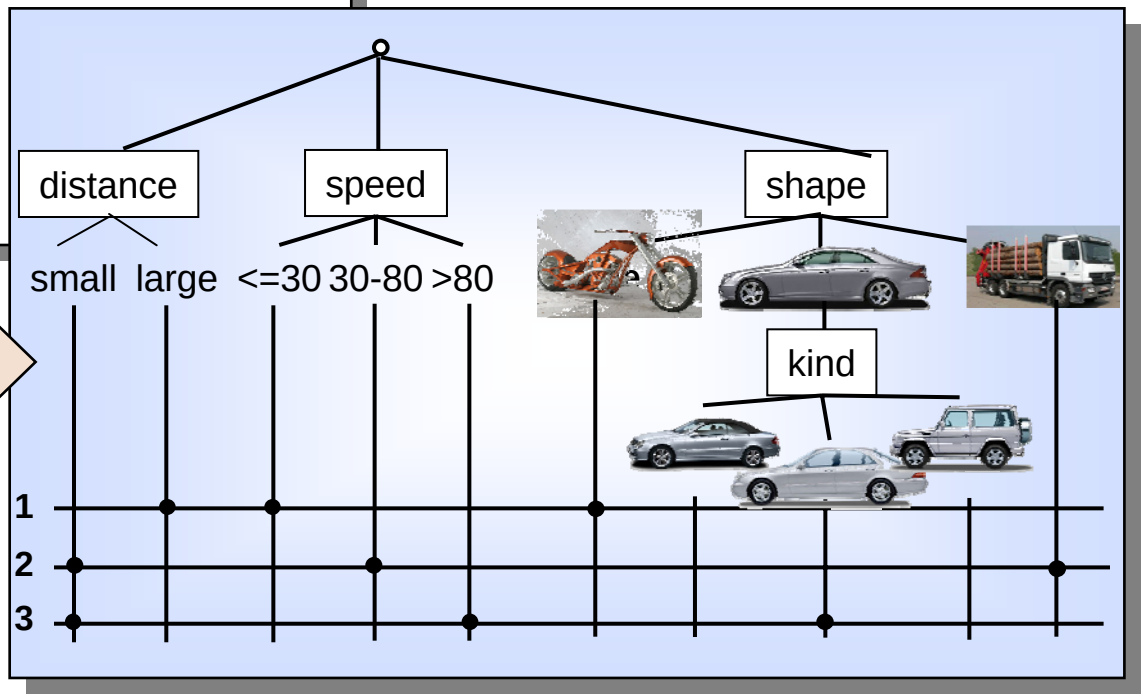
SUT: radar-based distance warning system



Test relevant aspects

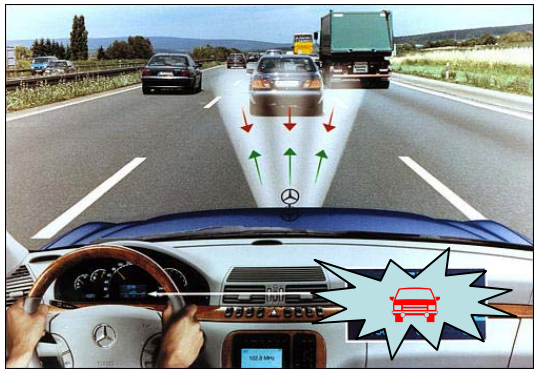
distance speed shape





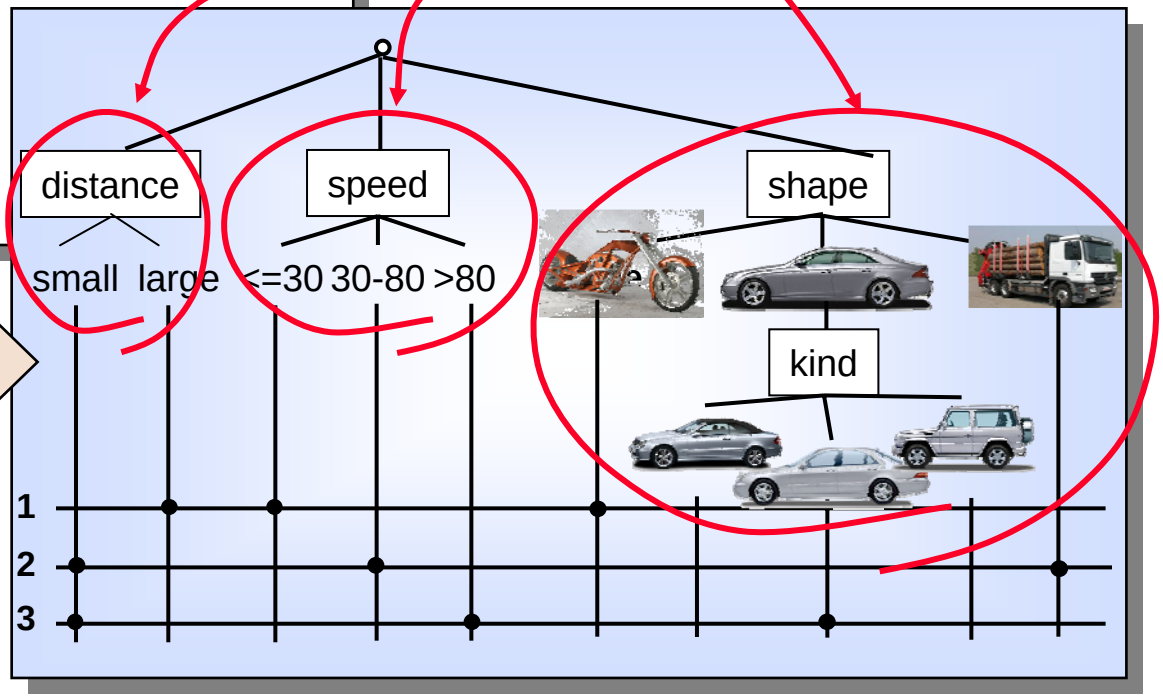
Classification-Tree Method

SUT: radar-based distance warning system

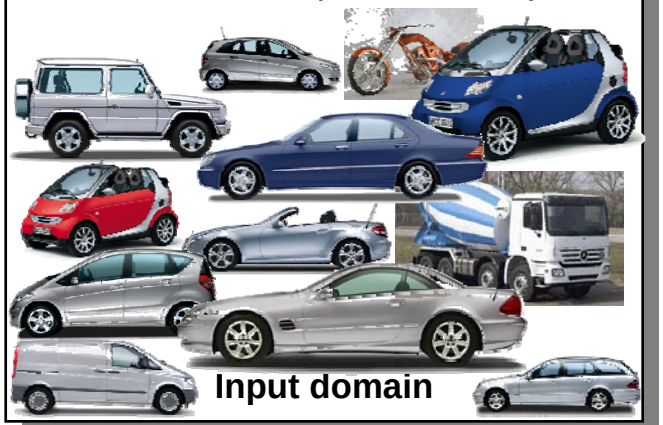


Maximum number of test cases

2 * 3 * 5 = 30

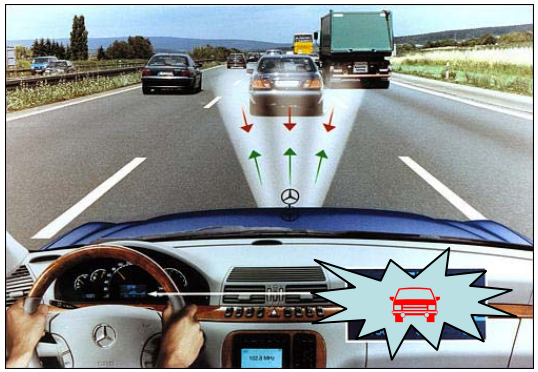


Test relevant aspects



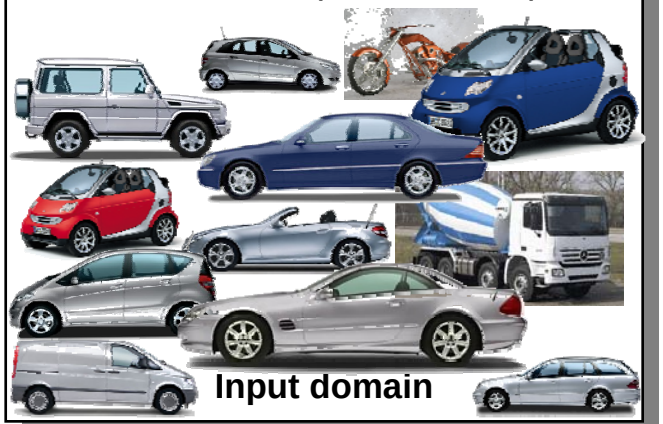
Classification-Tree Method

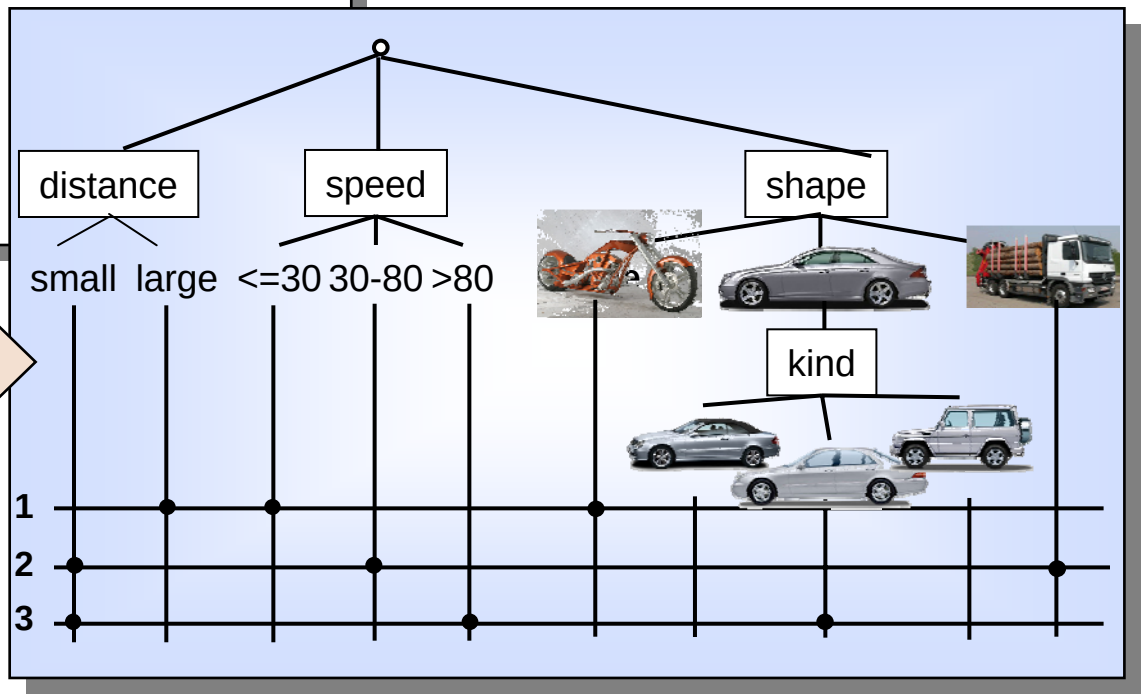
SUT: radar-based distance warning system



Test relevant aspects

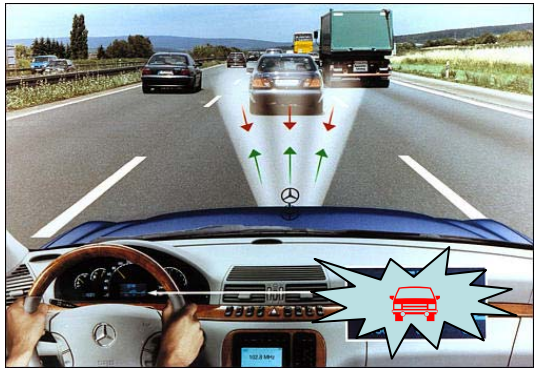
distance speed shape



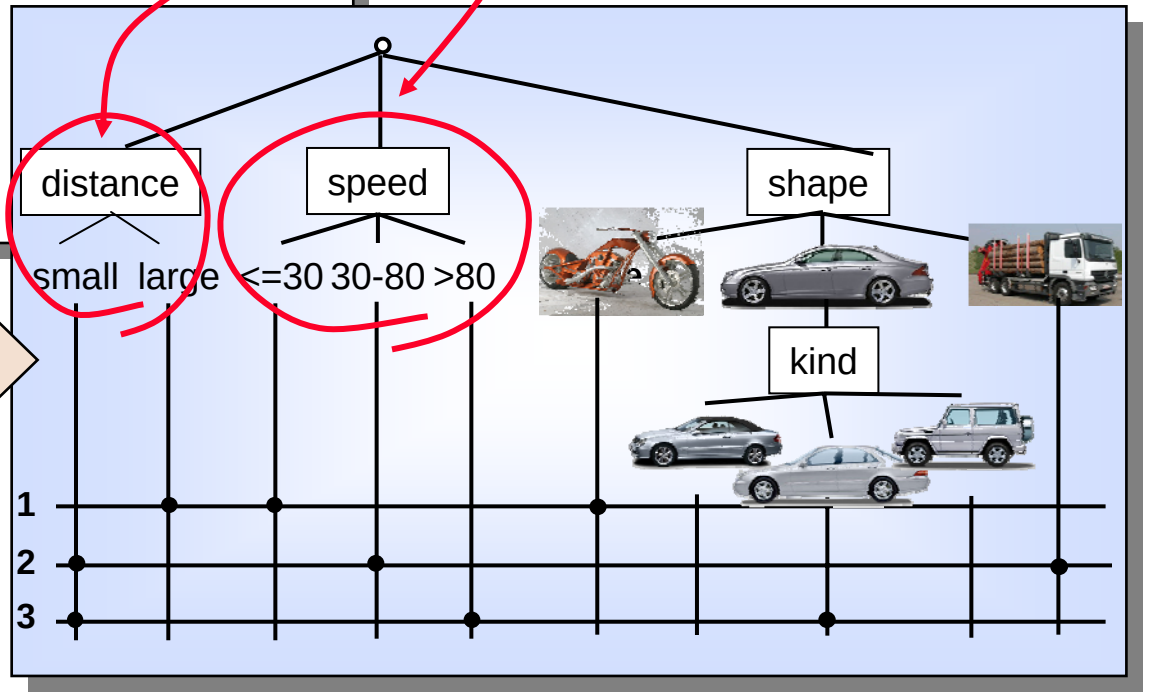


Classification-Tree Method

SUT: radar-based distance warning sys



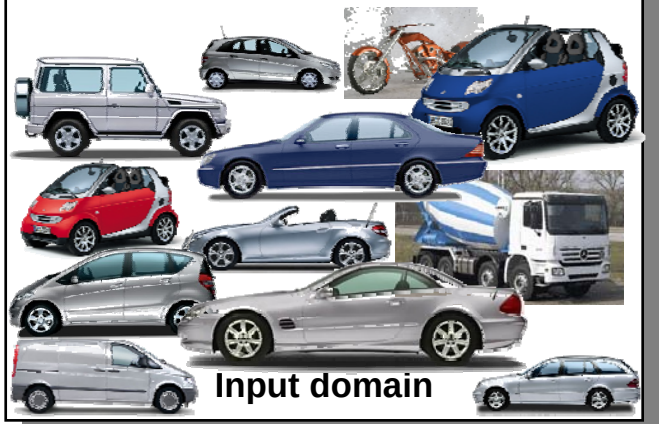
Rules for test case generation, e.g.
distance * speed => 6 test cases



Test relevant aspects



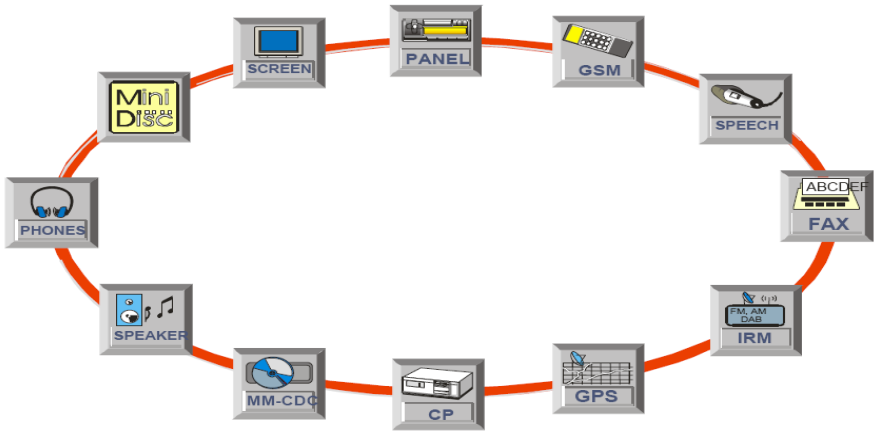
distance speed shape



Model-based Testing

Development of test models

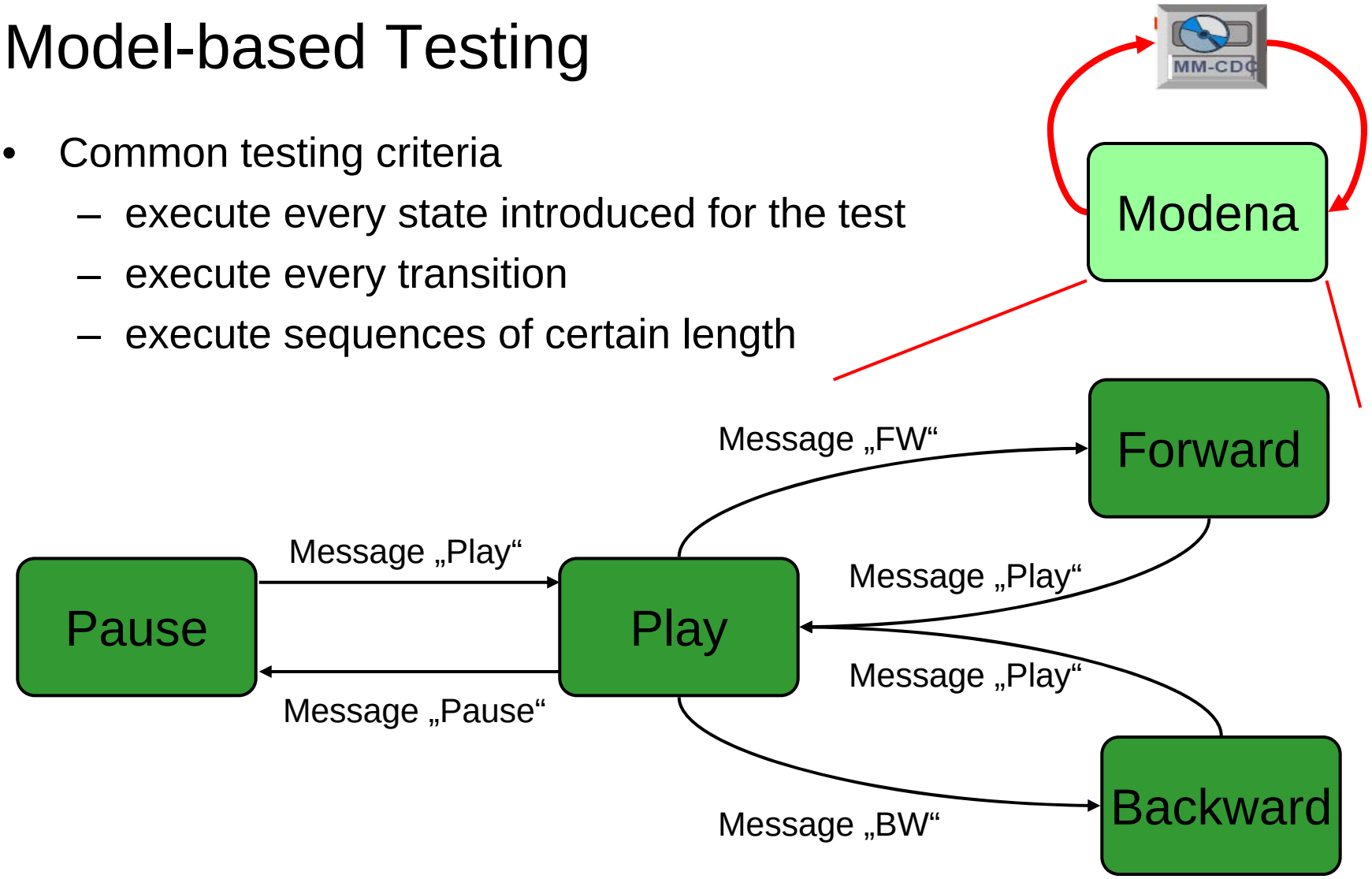
- use of common modeling languages, e.g. StateCharts, MSCs
- primarily for tests on bus protocol level, e.g. CAN, LIN, MOST, FlexRay
- easy to understand
- test documentation



- powerful tool support (providing abstractions for defined messages, e.g. Modena)
- widely used in automotive industry and other domains

Model-based Testing

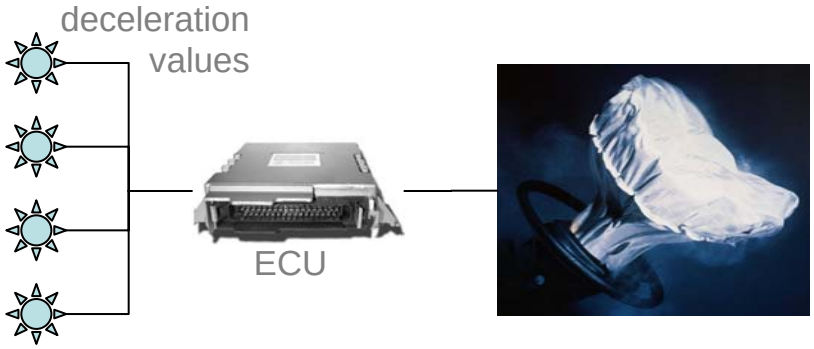
- Common testing criteria
  - execute every state introduced for the test
  - execute every transition
  - execute sequences of certain length



Why testing continuous behavior is different...

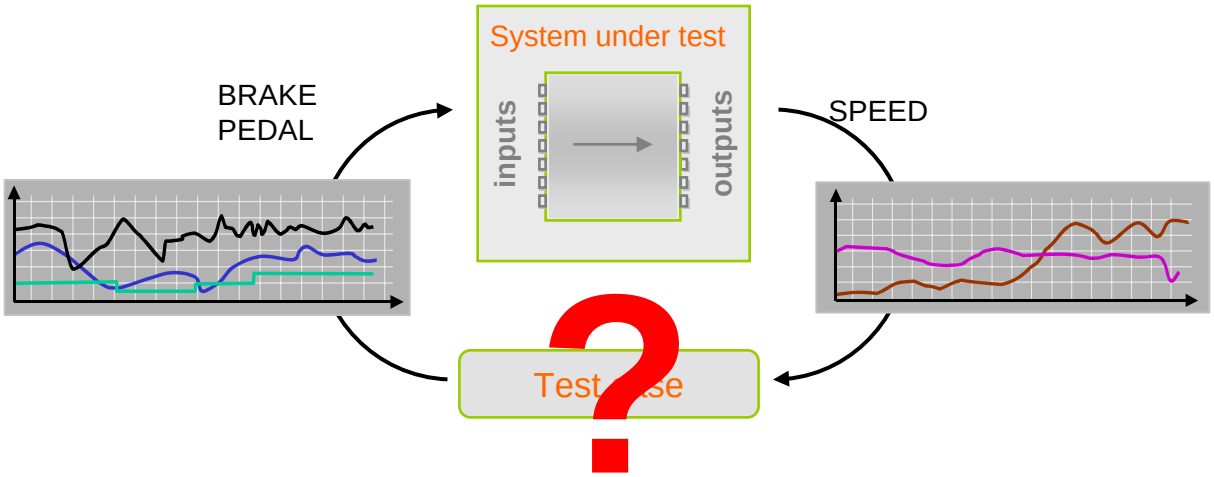
systems under test are

- signal driven and/or event driven
- functional complex due to data complexity (“large interfaces”)
- functional complex due to timing complexity (sequences, temporal conditions, signal processing etc.)
  - Noise
  - Monotony
  - Sequences (off ⇨ on ⇨ off)
- hybrid systems (mixture of continuously changing and static inputs/control and information systems)



Difficult to cope with conventional test methods

Time-Partition-Testing



- Test cases stimulate the system under test by continuously defining input quantities for the system under test
- Test cases react on the system behavior by observing the output quantities

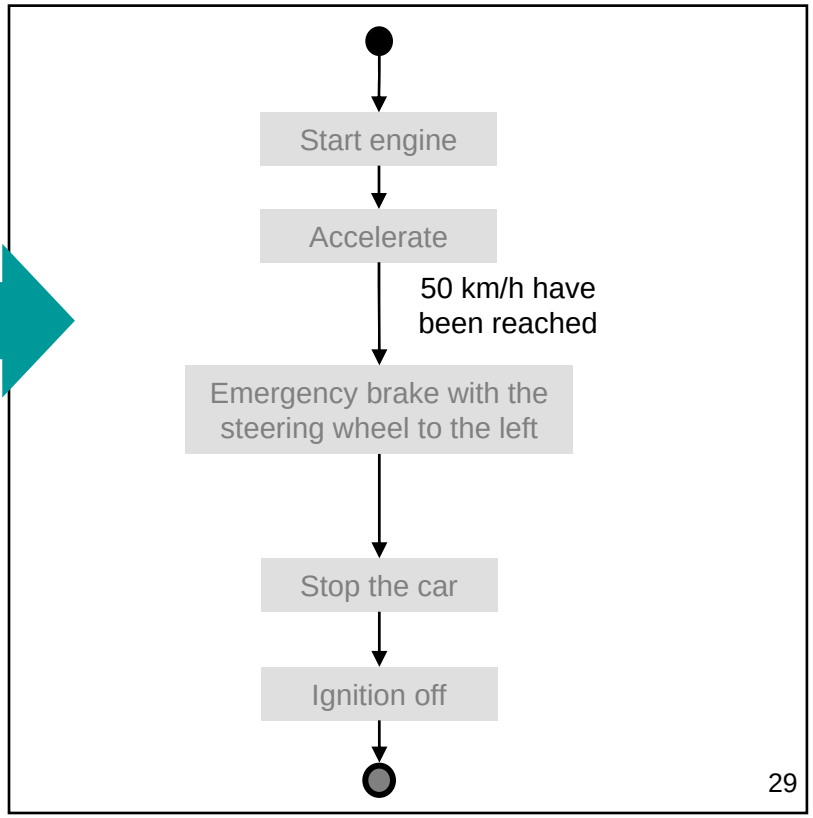
Time-Partition-Testing

Test Modeling

System Testing Scenarios often consist of a sequence of logical phases

1. Start engine
2. Accelerate until speed 50 km/h has been reached
3. Emergency brake with steering wheel as far as it will go left-hand
4. Stop the car
5. Ignition off

Such sequences are described with TPT using naturally readable state machines

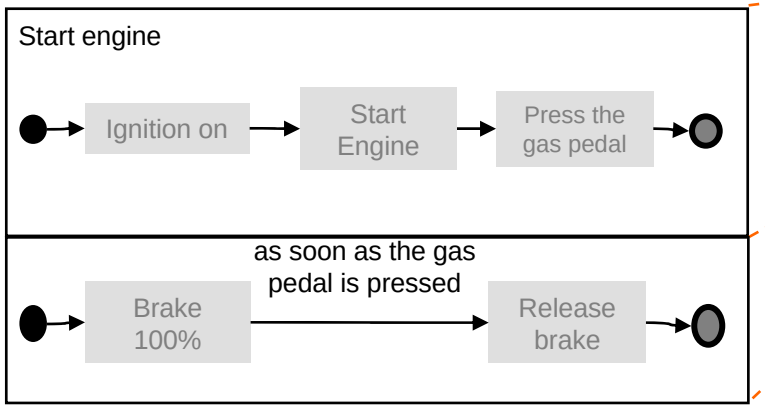


Time-Partition-Testing

Test Modeling

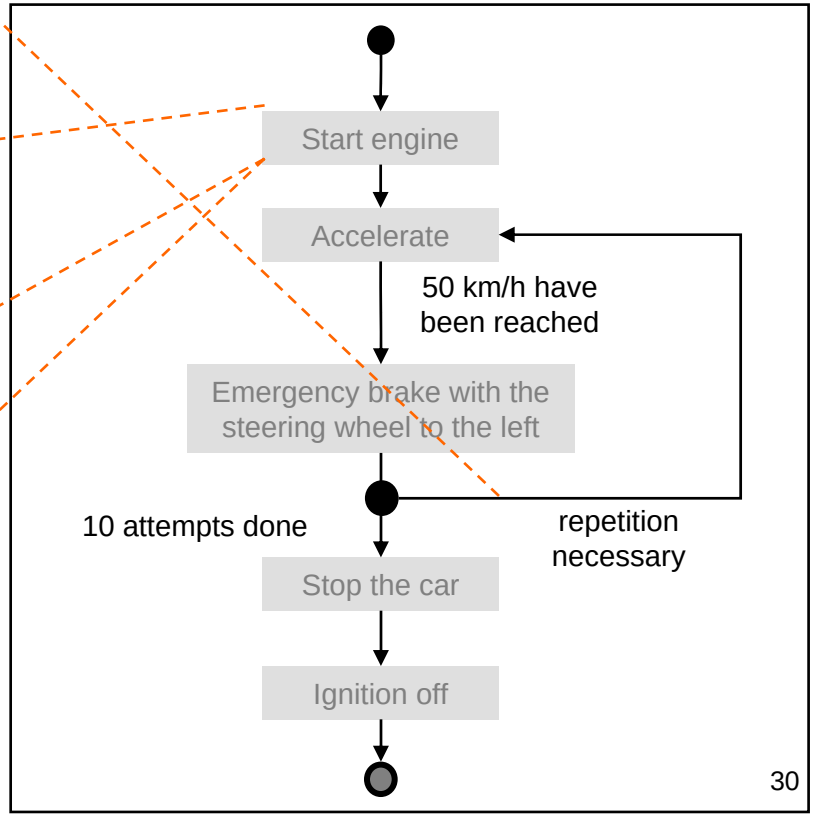
Possibility to model more complex situations (e.g., branches and loops)

Details of the sequences can be hidden by hierarchical state machines



Parallel state machines allow intuitive and powerful test models of more complex sequences

Such sequences are described with TPT using naturally readable state machines



Time-Partition-Testing

Test Modeling

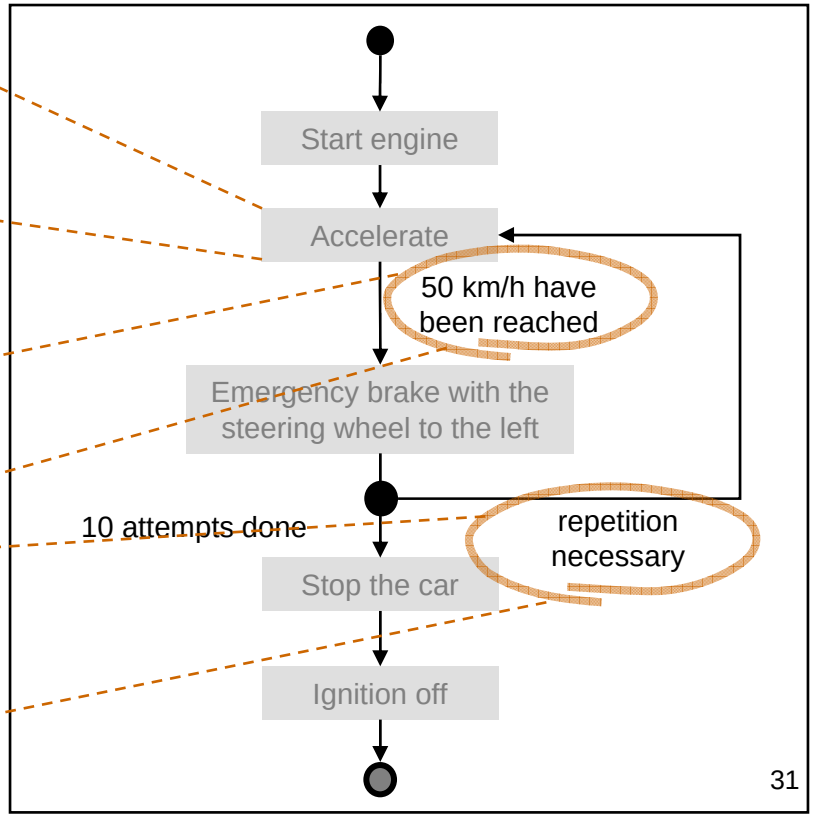
Equations with C-like syntax are used for executable signal definitions on the lowest level

```
pedal(t)      = min(10 + 10 * t, 100)
brake(t)      = 0.0
handbrake(t)  = 0.0
```

Transition conditions are precisely defined by expressions

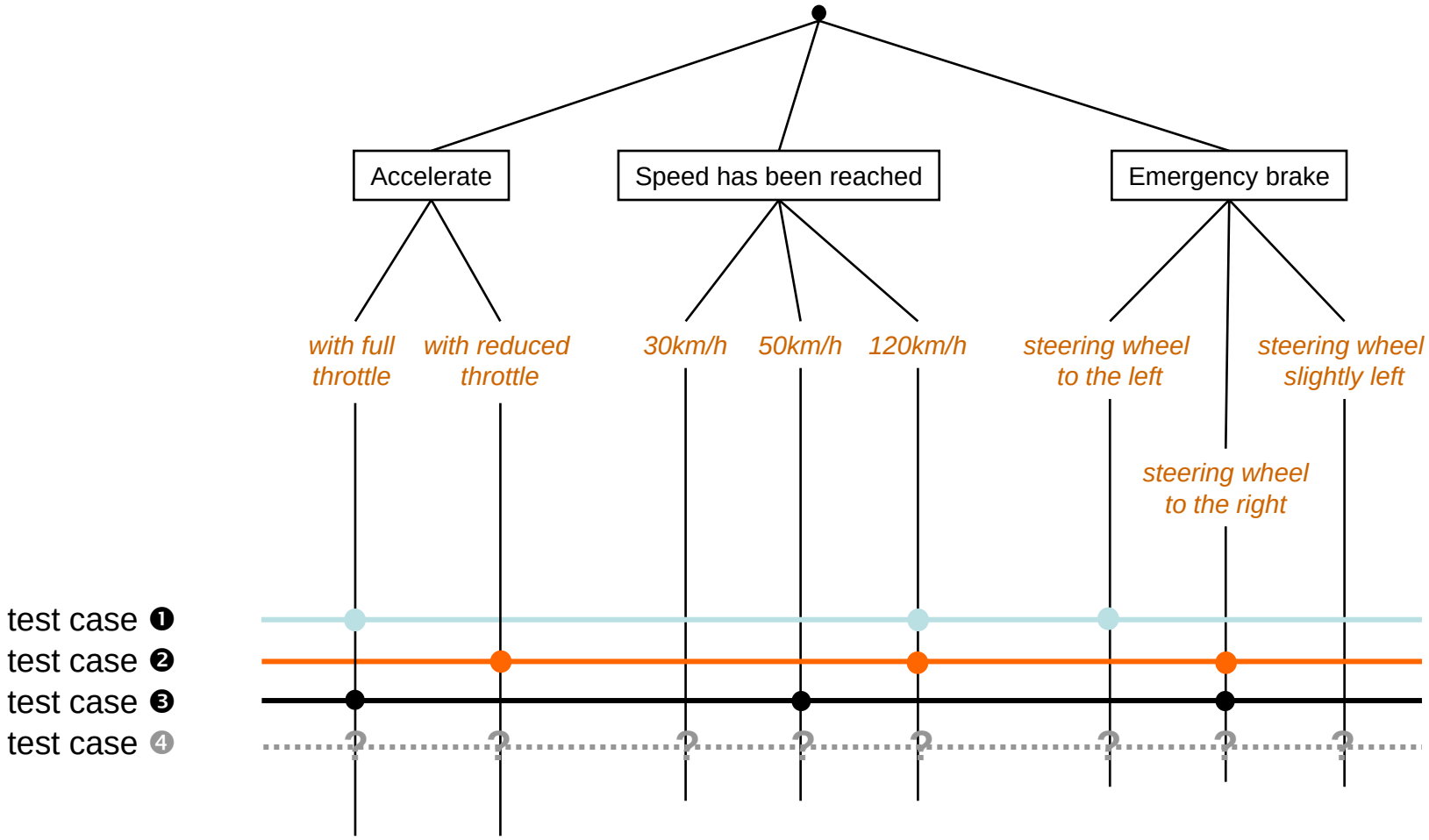
```
speed(t) >= 50.0
```

```
count <= 9          /* Condition */
-------------------------------------
count = count + 1;  /* Action */
```



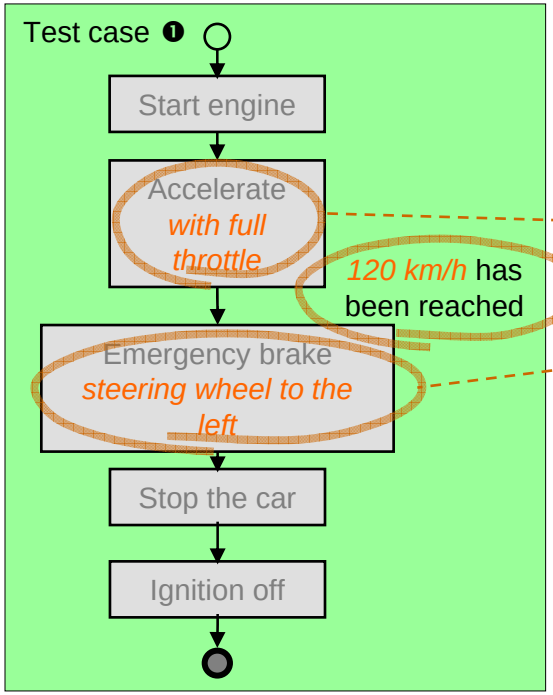
Time-Partition-Testing

Combination of Scenarios

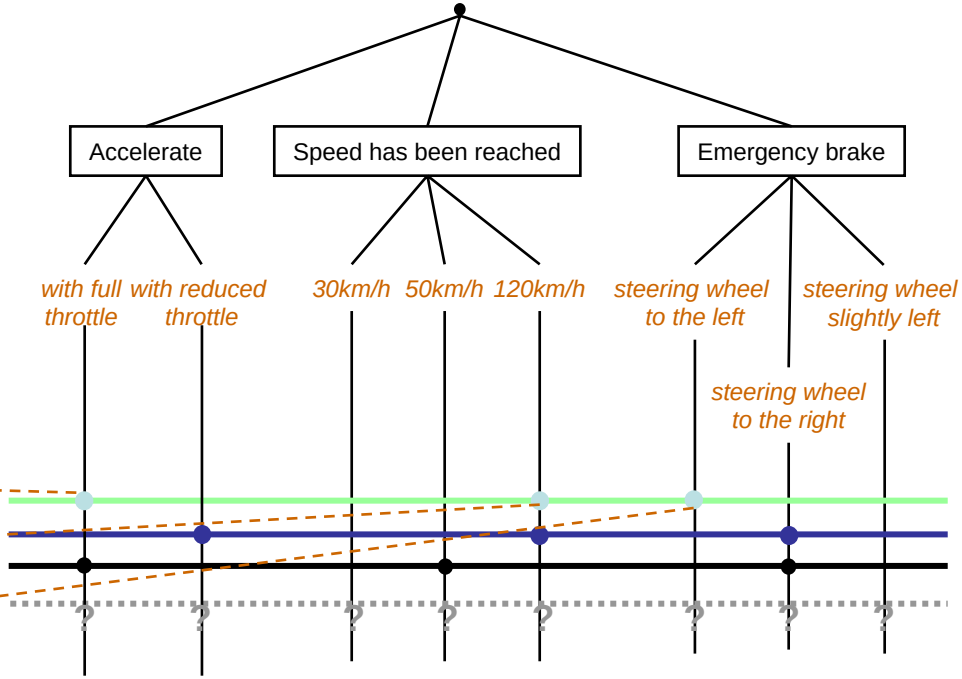


Time-Partition-Testing

Combination of Scenarios

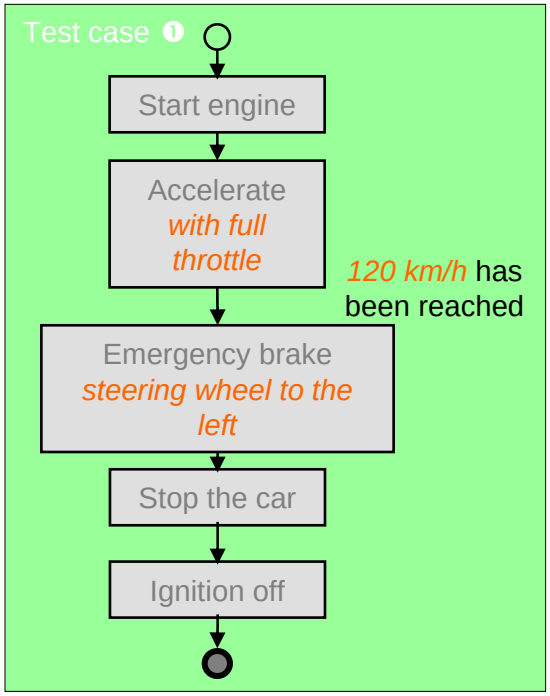


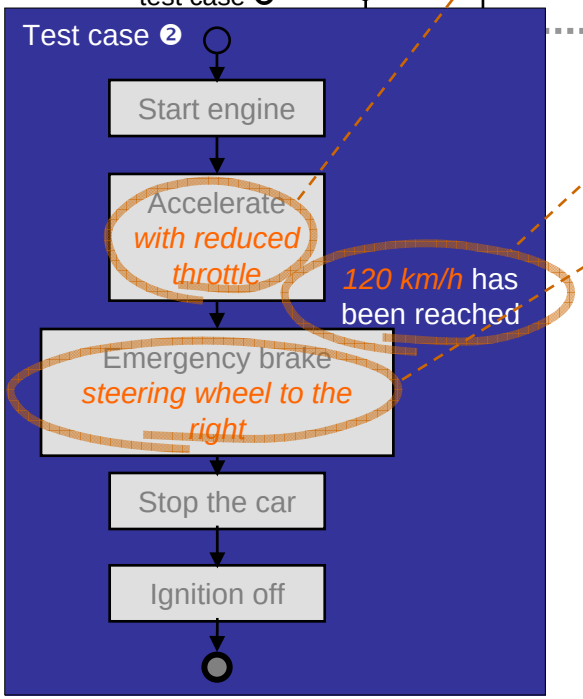
120 km/h has been reached

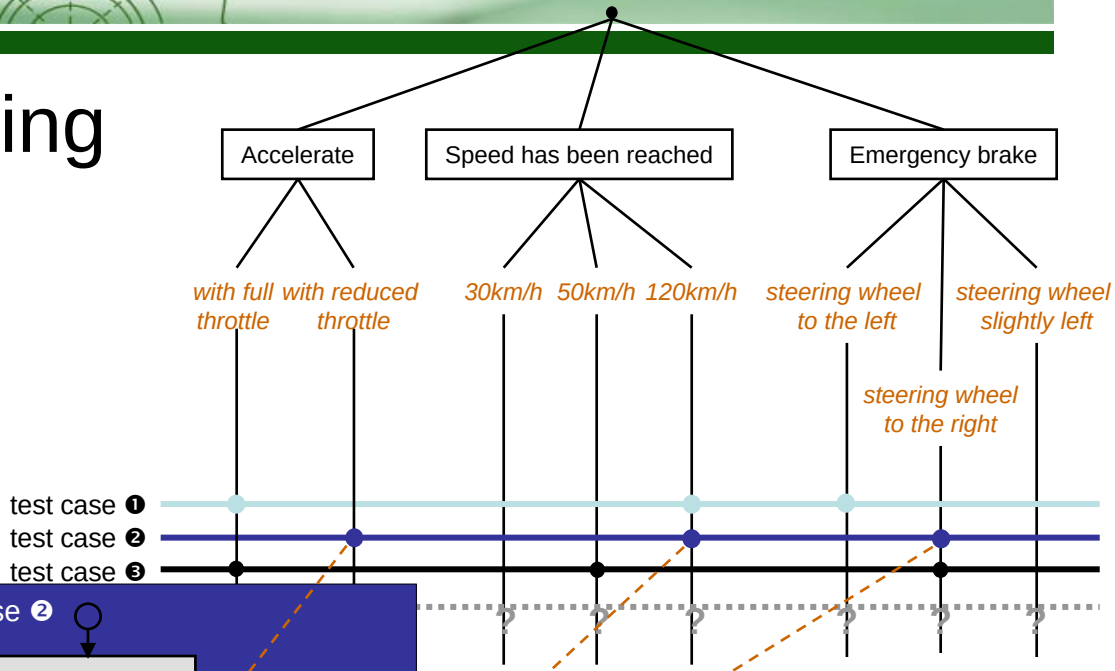


Time-Partition-Testing

Combination of Scenarios







Time-Partition-Testing

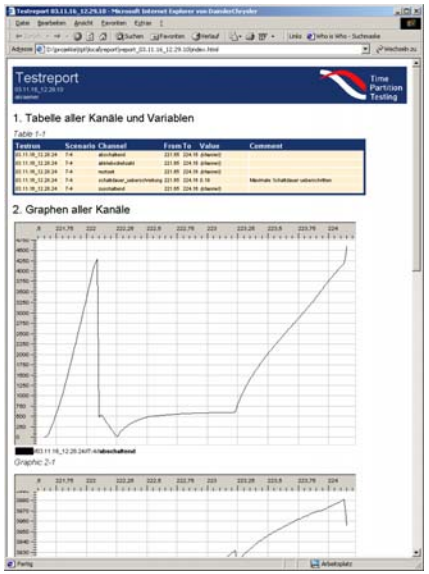
Test Execution

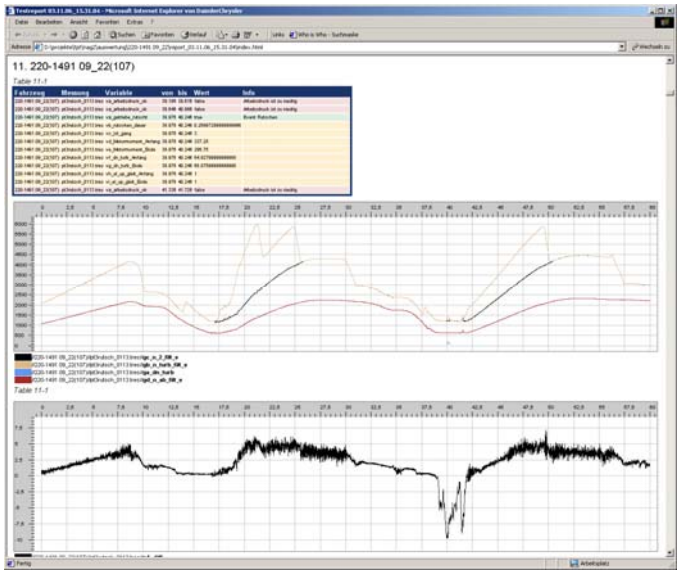
- Fully automated test execution
- TPT virtual machine for test execution (small and efficient real-time execution engine)
- TPT VM available for different test and simulation environments, e.g.
  - Software-In-The-Loop (Matlab/Simulink)
  - Hardware-In-the-Loop

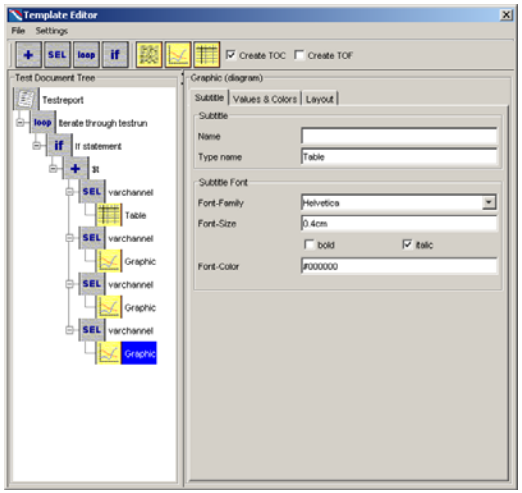
Time-Partition-Testing

Test Execution

- Assessment language based on Python scripts
- Generated test documentation of analyzed test results
- Based on configurable templates
- Generates documents in HTML, PDF and RTF







Evolutionary Testing

Search for interesting test data fully automatically by

- transforming the test problem into an optimisation problem,
- interpreting the test object‘s input domain as search space
- applying meta-heuristic search techniques, such as evolutionary algorithms to solve this problem

- representation of individuals/test data
- test objective has to be defined numerically (suitable fitness function)
- fitness assessment for generated test data based on monitoring results
- iterative procedure, combining good test data to achieve better test data

Evolutionary Testing

Transforming Test Objectives into Search Problems

Different test objectives require different fitness functions

- Functional testing ⇨ search for test datum causing logical error
- Real-time testing ⇨ search for test datum with longest and shortest execution time
- Safety testing ⇨ search for test datum violating system safety constraints
- Robustness testing ⇨ search for test datum stressing fault-tolerance mechanisms
- Structural testing ⇨ search for test datum executing particular program construct
- Mutation testing ⇨ search for test datum which detects the injected fault

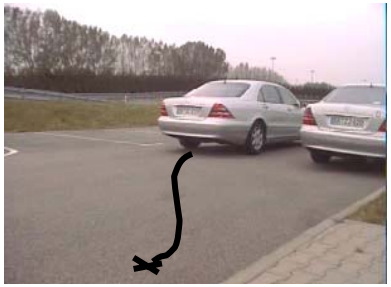
Evolutionary Testing
of Autonomous Parking System

System description

- Measuring the size of the parking space using environmental sensors and parking space model
- Signaling sufficient sized parking spaces to the driver

- If parking is committed by the driver:

  - Determine the position of the car with respect to the parking space
  - Plan the trajectory path for the parking maneuver



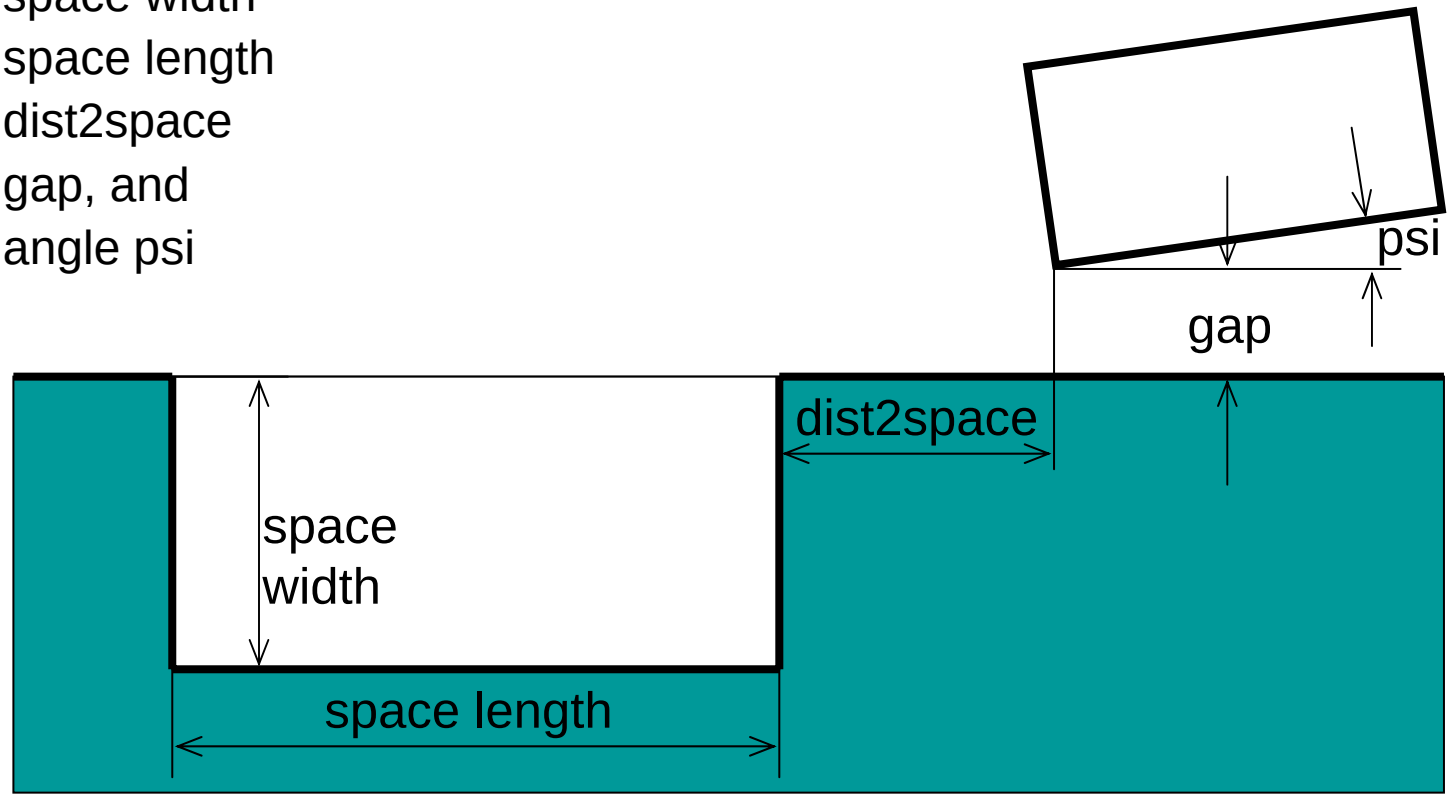
  - Drive the car into the parking space autonomously

Evolutionary Testing

of Autonomous Parking System

Generation of parking scenarios by evolutionary algorithms varying

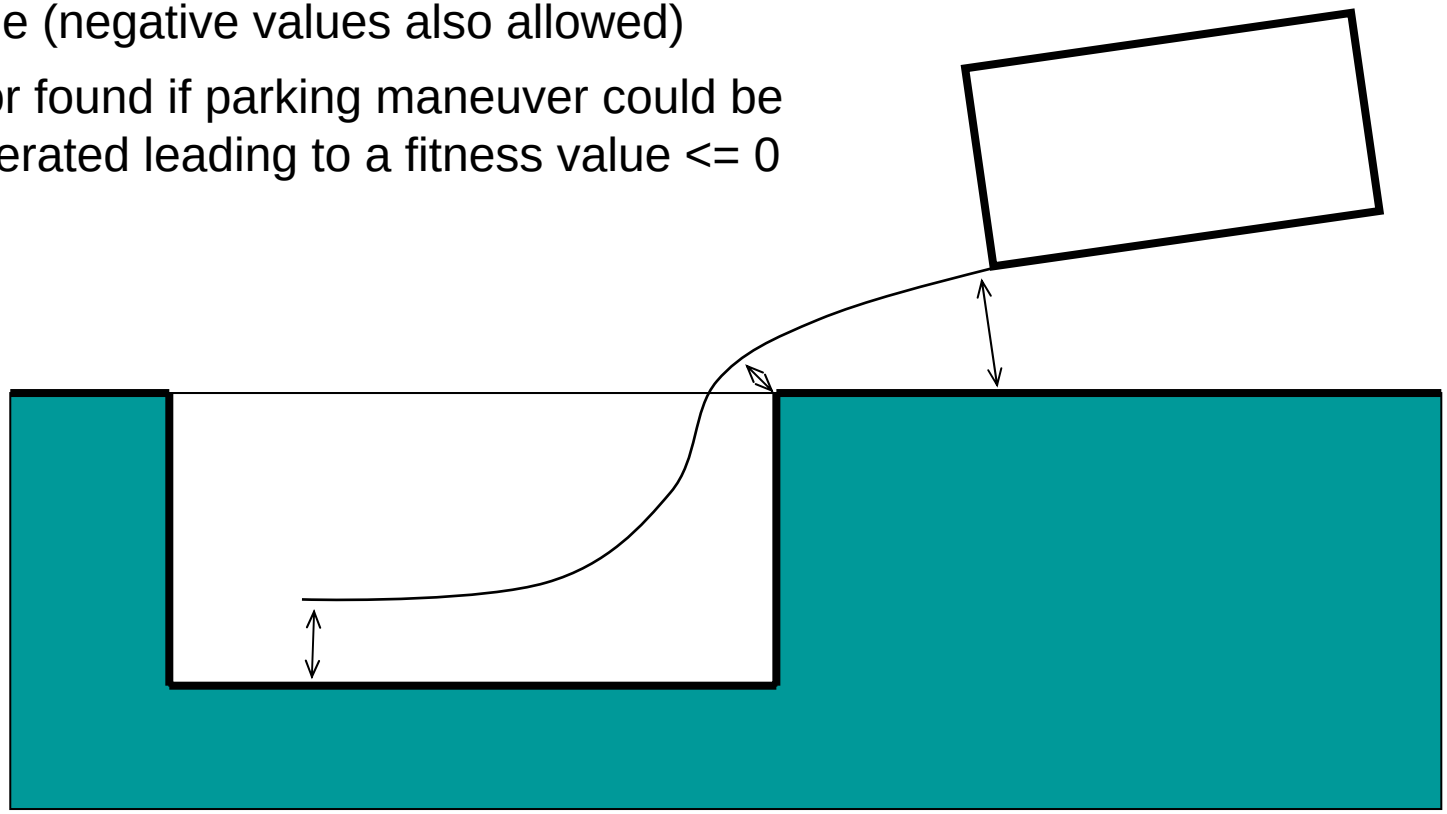
- space width
- space length
- dist2space
- gap, and
- angle psi



Evolutionary Testing

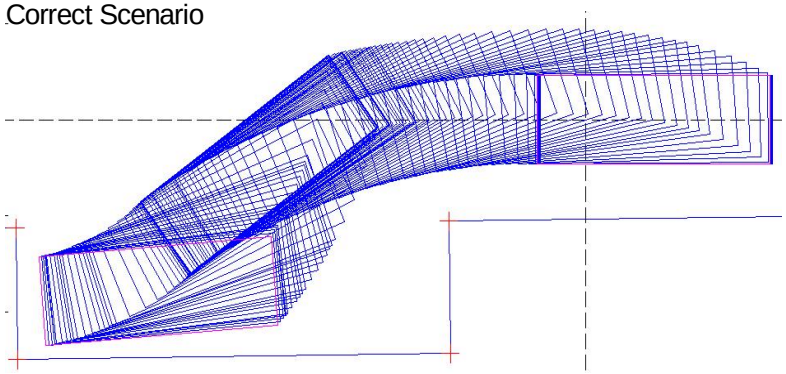
of Autonomous Parking System

- Selection of smallest distance between car and collision area as fitness value (negative values also allowed)
- Error found if parking maneuver could be generated leading to a fitness value <= 0



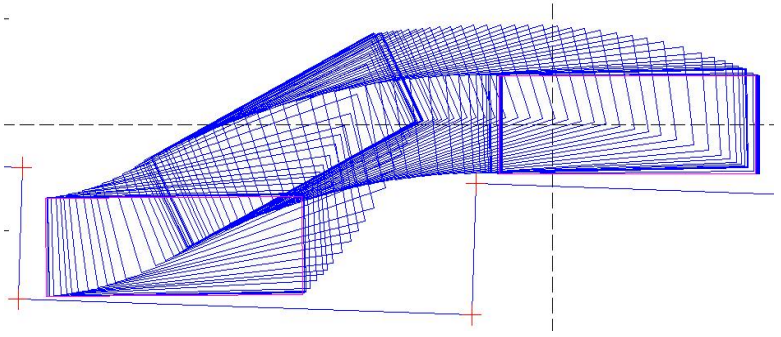
Evolutionary Testing
of Autonomous Parking System

Correct Scenario



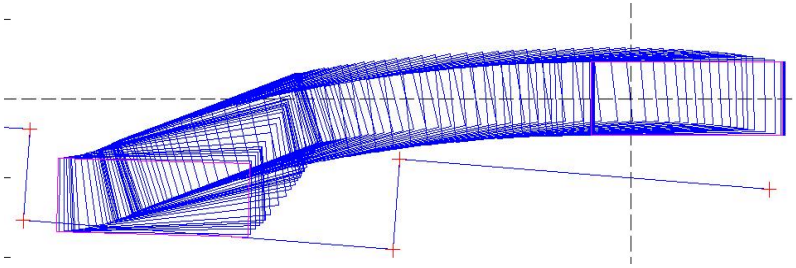
Generation 01 / Individual 13

Critical Scenario



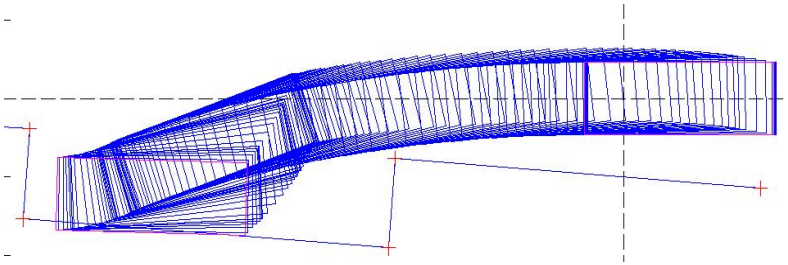
Generation 10 / Individual 02

Scenario leading to erroneous system behavior (edge entered collision area)



Generation 20 / Individual 06

Scenario leading to erroneous behavior (end-position in collision area)



Generation 20 / Individual 05

Conclusion

- Most testing is black-box testing
- Methods necessary to support systematic test case design
  - graphical methods preferred
  - applicable without programming background
- Test automation important for test efficiency and high test coverage
- Ideally, tools support both issues

- Vision: Berner & Mattner Messina platform capable of executing test cases defined by different methods on different target systems integrating various system models

References

Wegener, J. (2005): Systematic Black-Box Test Case Design. TTCN-3 User Conference, France.

Video on Automatic Braking System, Zweites Deutsches Fernsehen, 2002.

- CTE – www.systematic-testing.com
- Modena – www.berner-mattner.com
- TPT – www.piketec.com
- Messina – www.berner-mattner.com