Mercedes-Benz

Model-based Development of a Dual-Clutch Transmission using Rapid Prototyping and SiL

Holger Brückmann, Jochen Strenkert, Dr. Uwe Keller, EP/MAG, Daimler AG

Benno Wiesner-Tittes, Dr. Andreas Junghanns, QTronic GmbH

July 1, 2009

Slides presented at the VDI Congress Transmissions in Vehicles 2009, Friedrichshafen, Germany, 30.06.-01.07.2009

Outline of the talk

1 History and Motivation

2 DCT Development

3 Rapid Prototyping

4 Automated Testing

5 Code Coverage Analysis

6 Outlook

Outline of the talk

1 History and Motivation

2 DCT Development

3 Rapid Prototyping

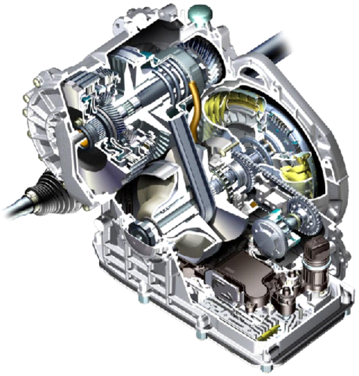
4 Automated Testing

5 Code Coverage Analysis

6 Outlook

History and Motivation

Software-in the-Loop simulation is used at Daimler transmission development since many years:



Autotronic since 1998

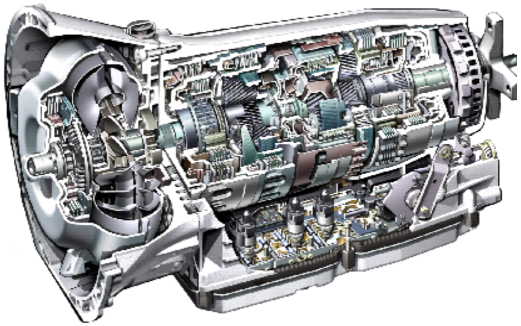
- Rapid-prototyping via A-Muster
- Simulink-SiL with floating-point code
- Module- and system-tests in Simulink
- Continuous operation simulations with fix-point code

7G-Tronic since 1998

- Rapid-Prototyping via Backbone
- Fix-point code simulation
- System tests
- Continuous operation simulations with fix-point code

- many different tools
- many of them developed in-house

➔ Objective for new projects:
+ simplify tool chain
+ use of „standard software“
+ minimize in-house customization of tools



➔ first application of the new tool chain:
dual clutch tranmission (DCT) development

Outline of the talk

1 History and Motivation

→ 2 DCT Development

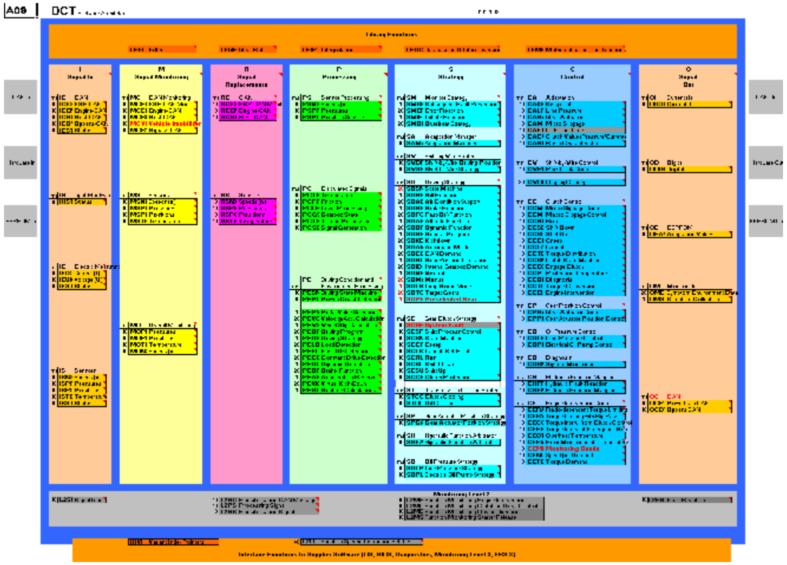
3 Rapid Prototyping

4 Automated Testing

5 Code Coverage Analysis

6 Outlook

Function tool box



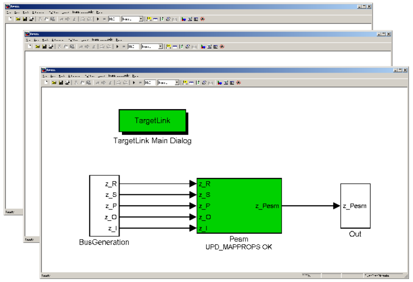
- software functions with ca. 150 modules
- developed using MatLab/Simulink/Stateflow
- and dSpace TargetLink with DataDictionary
- 100% autocode

Objective of SiL:

- integrated tool chain
- cover software-in the-loop and rapid prototyping
- support software validation and automated test

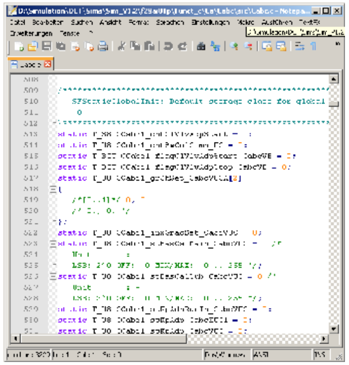
Workflow for software development

150 modules from MatLab/Simulink



dSpace code generator

C-code



cross compiler

object code for Infineon TriCore

- software for control unit
- A2L and application parameter
- objects for all 150 modules

Microsoft Visual Studio Compiler

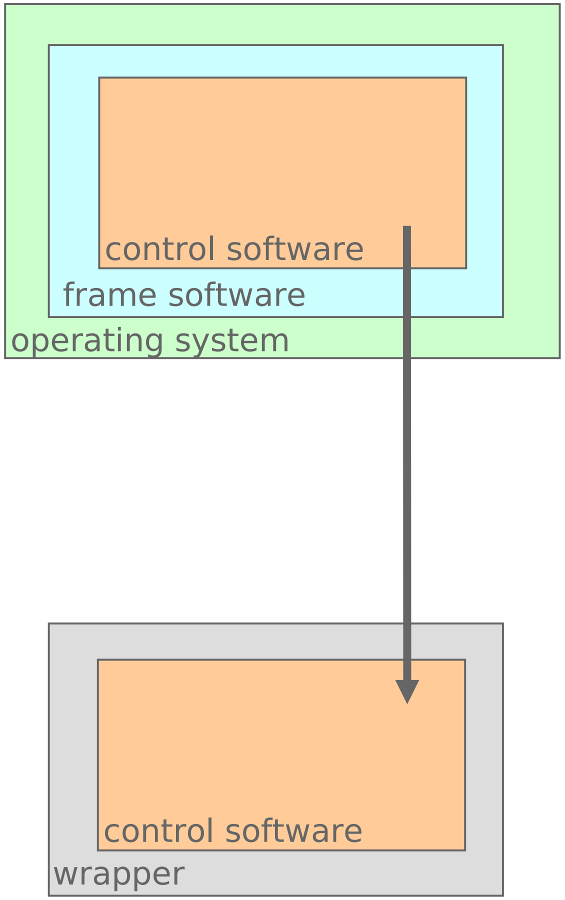
object code for x86

- software for SiL
- A2L database, application paramter
- DLL for simulation
- objects for all 150 modules

Advantages:

- no adaption of Simulink modules required
- same code for ECU und SiL (fix-point integer)
- **ECU and SiL use the same sources**

Structure of the ECU software



operating system

- device driver
- memory
- …

frame software

- EEPROM
- CAN data
- …

control software

- 150 modules

Structure of the SiL software

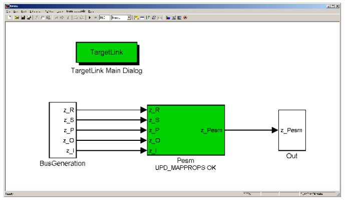
The wrapper emulates the functions of the frame software.

Many wrapper functions simply return default values.

wrapper

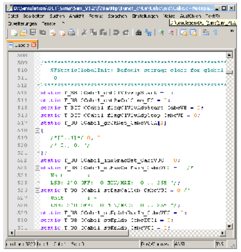
+ complete control software
+ CAN Data
+ EEPROM

- no error code memory
- no diagnostic functions



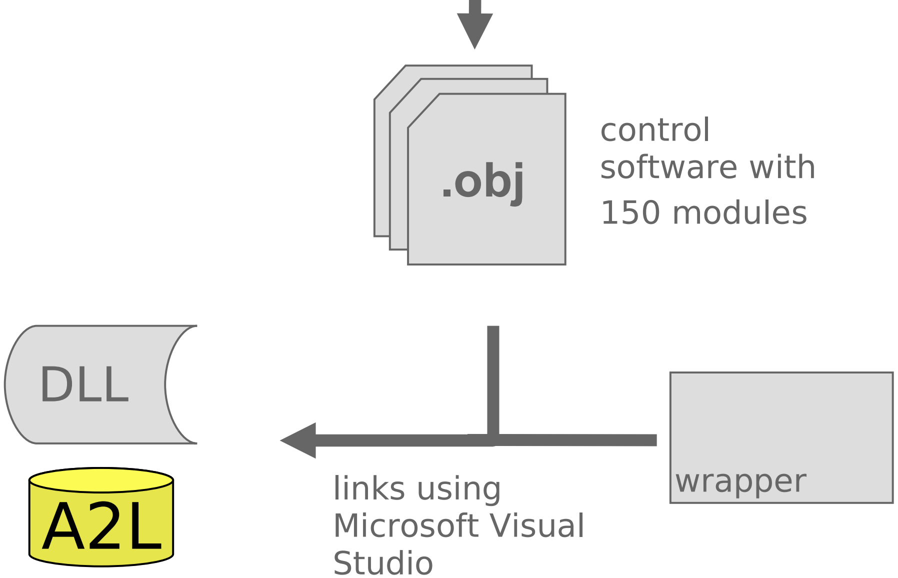
Software developer edits his module

generates code



gets the other 149 modules as object code

compiles his module using Microsoft Visual Studio



control software with 150 modules

links using Microsoft Visual Studio

software for SiL

Working results:

- DLL for x86 PC containing the entire control software
- A2L database with adresses of the DLL
- Build process within minutes, because only 1 module was changed
- Every developer can test his modules at once in system context
- No access to all module sources required during the build process

Simulation environment

Tools:

- Simulation: Silver (QTronic)
- Measurement: Canape (Vector)
- Debugging: Visual Studio (Microsoft)
- Automated Test: TestWeaver (QTronic)
- Code Coverage: Testwell CTC++ (Verifysoft)

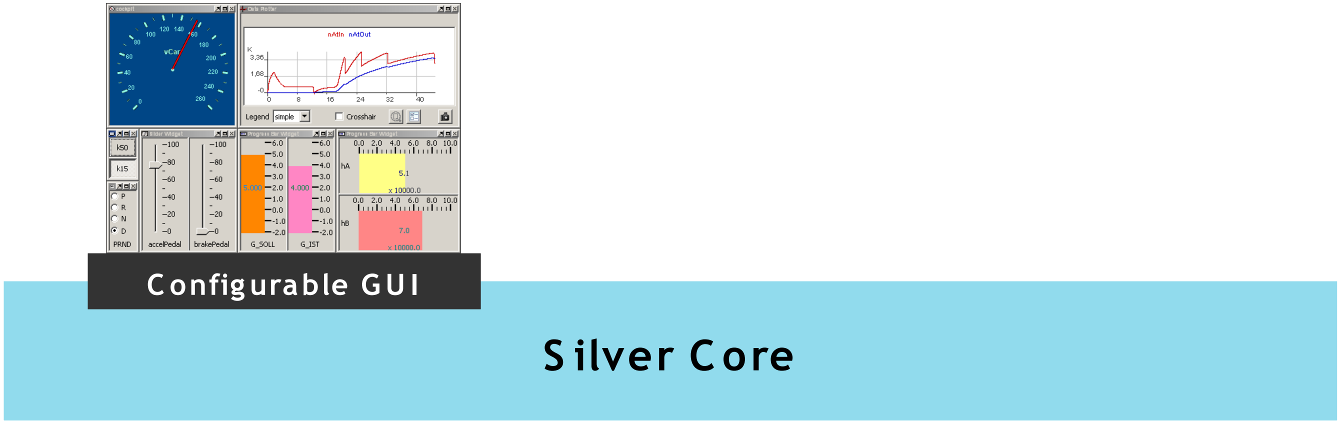
Simulation environment

Tools:

- Simulation: Silver (QTronic)
- Measurement: Canape (Vector)
- Debugging: Visual Studio (Microsoft)
- Automated Test: TestWeaver (QTronic)
- Code Coverage: Testwell CTC++ (Verifysoft)

Graphical user-interface (GUI) to SiL with Silver:

- Interaction of driver/user with simulated car
- Accel pedal, brake padel, ignition, temperature, ... can be controlled
- All inputs and outputs can be directly manipultated



Configurable GUI

Silver Core

Simulation environment

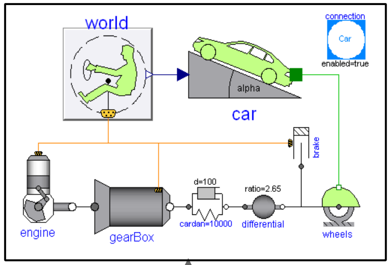
Tools:

- Simulation: Silver (QTronic)
- Measurement: Canape (Vector)
- Debugging: Visual Studio (Microsoft)
- Automated Test: TestWeaver (QTronic)
- Code Coverage: Testwell CTC++ (Verifysoft)

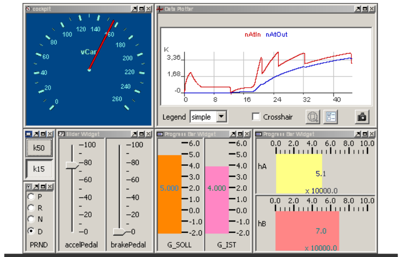
hardware DLL:

- simulated vehicle, engine and transmisssion
- developend in-house using Dymola

hardware-model



Dymola DLL



Configurable GUI

Silver Core

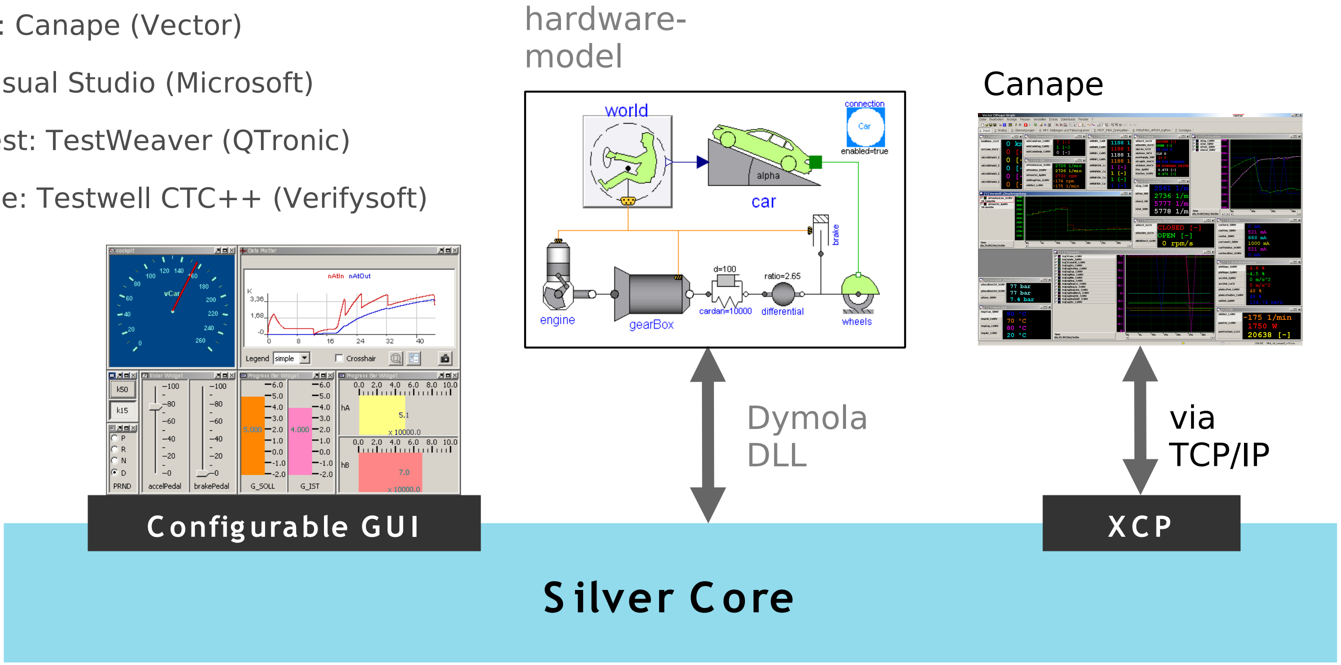
Simulation environment

Tools:

- Simulation: Silver (QTronic)
- Measurement: Canape (Vector)
- Debugging: Visual Studio (Microsoft)
- Automated Test: TestWeaver (QTronic)
- Code Coverage: Testwell CTC++ (Verifysoft)

XCP with Canape:

- XCP measurements via TCP/IP and Gigabit-Ethernet
- no limitation of bandwith as with CAN
- online calibration of parameters


hardware-model

Canape

Dymola DLL

via TCP/IP

Configurable GUI

XCP

Silver Core

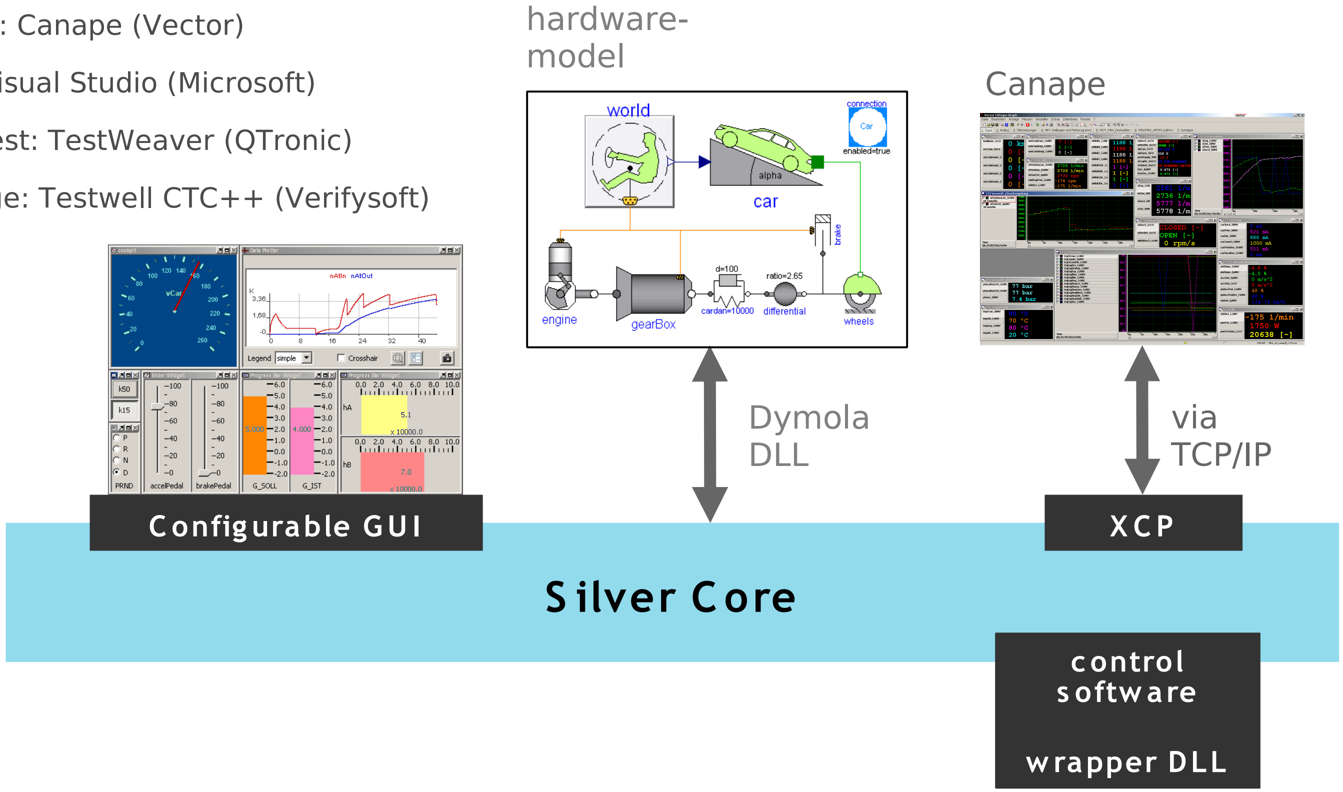
Simulation environment

Tools:

- Simulation: Silver (QTronic)
- Measurement: Canape (Vector)
- Debugging: Visual Studio (Microsoft)
- Automated Test: TestWeaver (QTronic)
- Code Coverage: Testwell CTC++ (Verifysoft)

Control software with wrapper DLL:

- entire TCU control software (all 150 modules)
- frame software software emulated by wrapper


hardware-model

Canape

Dymola DLL

via TCP/IP

Configurable GUI

XCP

Silver Core

control software

wrapper DLL

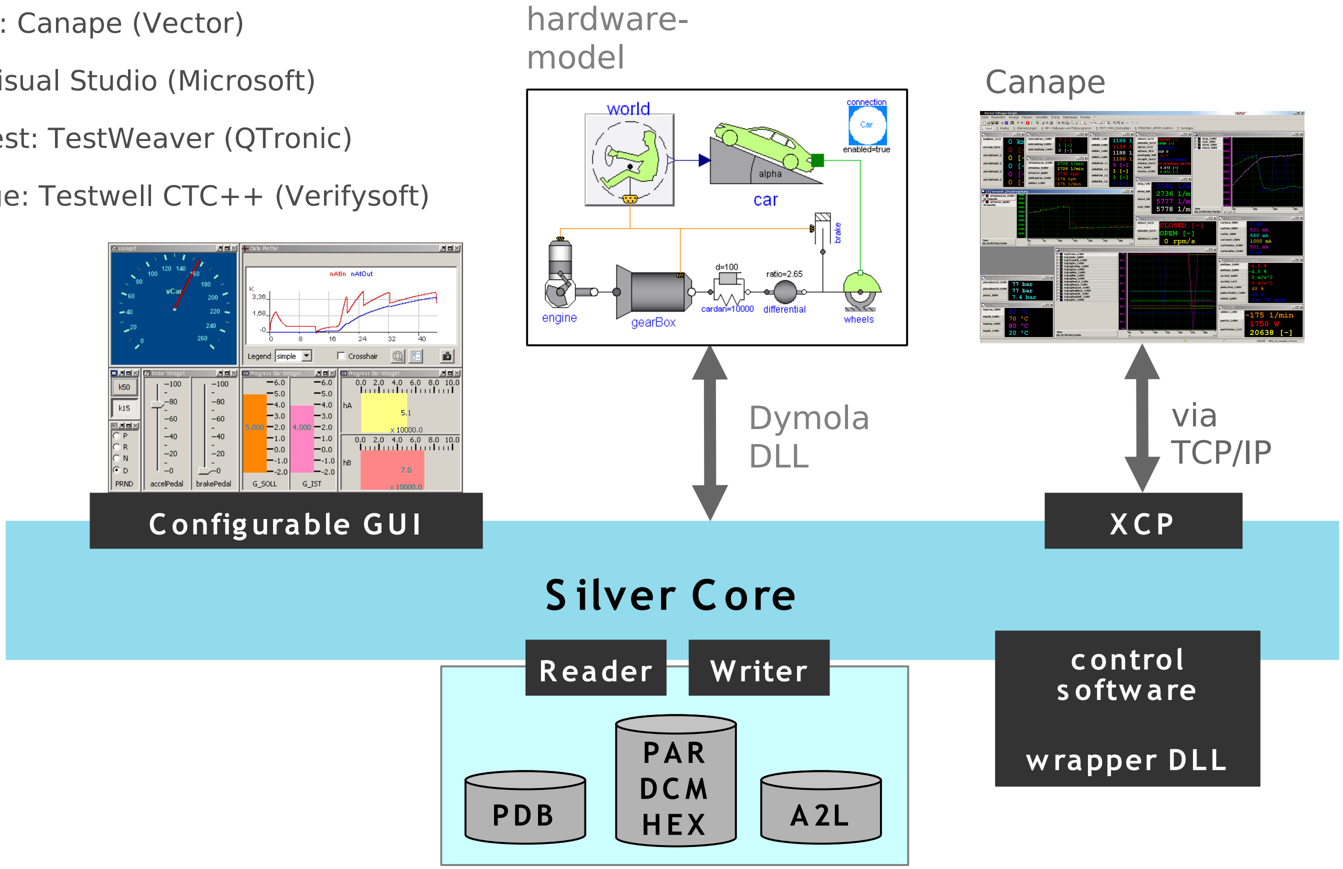
Simulation environment

Tools:

- Simulation: Silver (QTronic)
- Measurement: Canape (Vector)
- Debugging: Visual Studio (Microsoft)
- Automated Test: TestWeaver (QTronic)
- Code Coverage: Testwell CTC++ (Verifysoft)

A2L and parameter:

- A2L with address infromation adapted to the DLL
- complete and latest parameter values loaded at simulation start



hardware-model

Canape

Dymola DLL

via TCP/IP

Configurable GUI

XCP

Silver Core

Reader

Writer

PAR DCM HEX

PDB

A2L

control software wrapper DLL

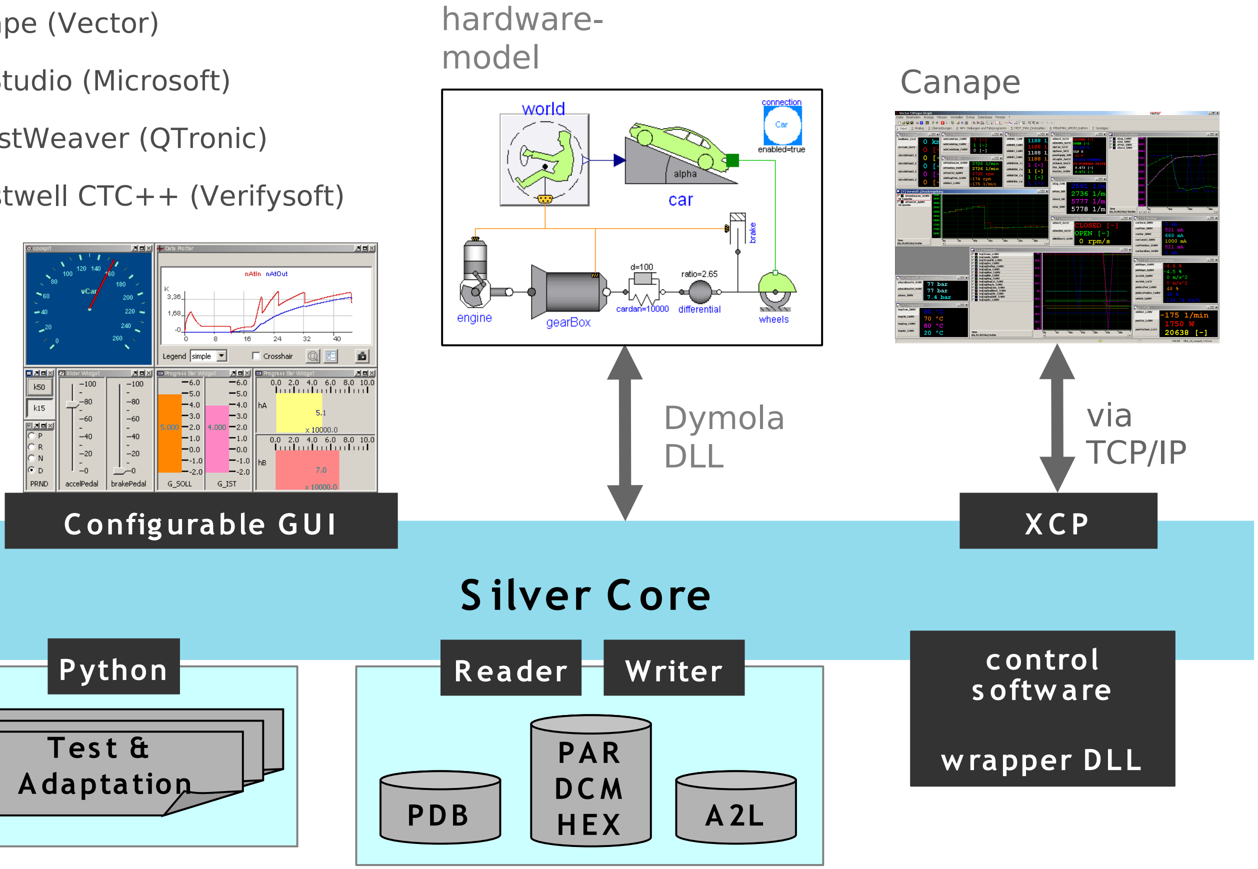
Simulation environment

Tools:

- Simulation: Silver (QTronic)
- Measurement: Canape (Vector)
- Debugging: Visual Studio (Microsoft)
- Automated Test: TestWeaver (QTronic)
- Code Coverage: Testwell CTC++ (Verifysoft)

Scripting with Python:

- frequently used procedures can be automated using scripting (e. g. engine start, adaptation procedure)



hardware-model

Canape

Dymola DLL

via TCP/IP

Configurable GUI

XCP

Silver Core

Python

Test & Adaptation

Reader

Writer

PDB

PAR DCM HEX

A2L

control software wrapper DLL

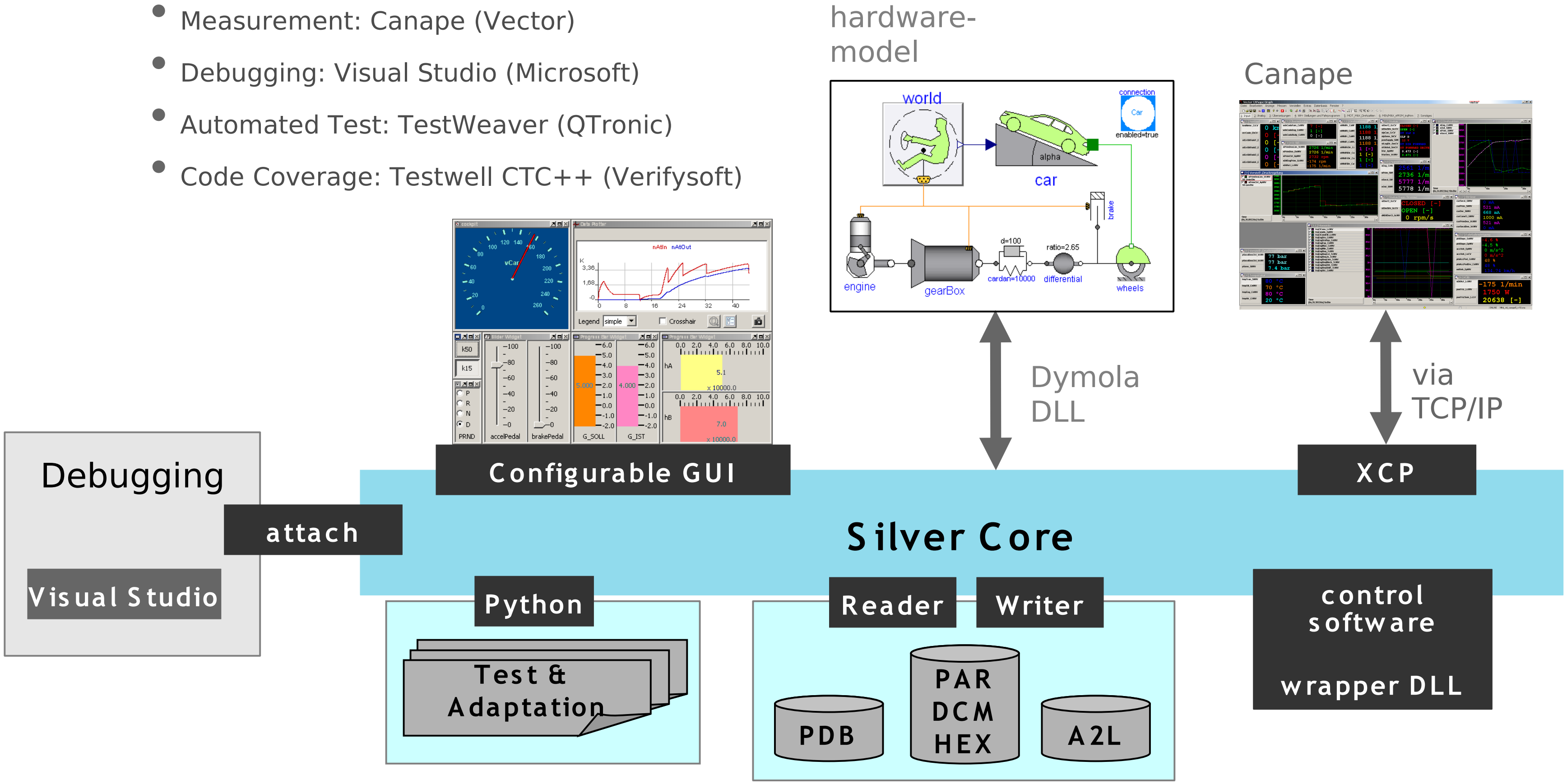
Simulation environment

Tools:

- Simulation: Silver (QTronic)
- Measurement: Canape (Vector)
- Debugging: Visual Studio (Microsoft)
- Automated Test: TestWeaver (QTronic)
- Code Coverage: Testwell CTC++ (Verifysoft)

Debugging with Visual Studio:

- Simulation can be suspended at any time
- Visual Studio Debugger can be attached to the Silver simulation process.



hardware-model

Canape

Dymola DLL

via TCP/IP

Debugging

attach

Visual Studio

Configurable GUI

XCP

Silver Core

Python

Test & Adaptation

Reader

Writer

PDB

PAR DCM HEX

A2L

control software wrapper DLL

Advantages of SiL

- Accelerated and early detection of errors because every developer can test his module in the context of all 150 modules
- Measurement as in a real vehicle (same measurement config. file)
- Fault simulation
  - sensor faults, gear jumps, overheating
  - convenient test environment for fault protection, detection and recovery strategies
- Support for EEPROM and adaptation procedures
- Scripting with Python
  - automated computation of adaptation values
- Debbuging
  - Every module developer can test and debug his module
    in closed-loop system context

Outline of the talk

1 History and Motivation

2 DCT Development

3 Rapid Prototyping

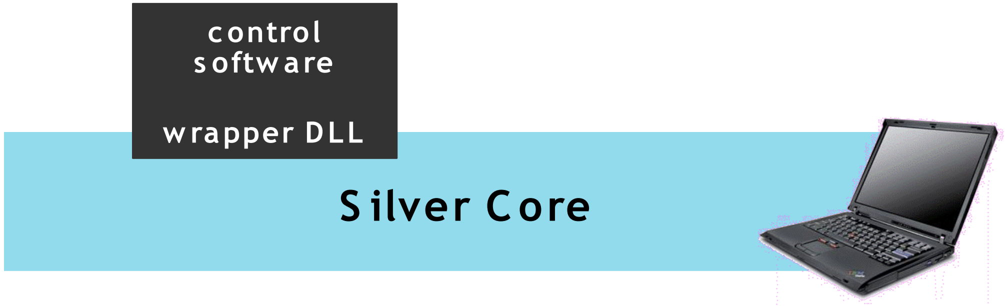
4 Automated Testing

5 Code Coverage Analysis

6 Outlook

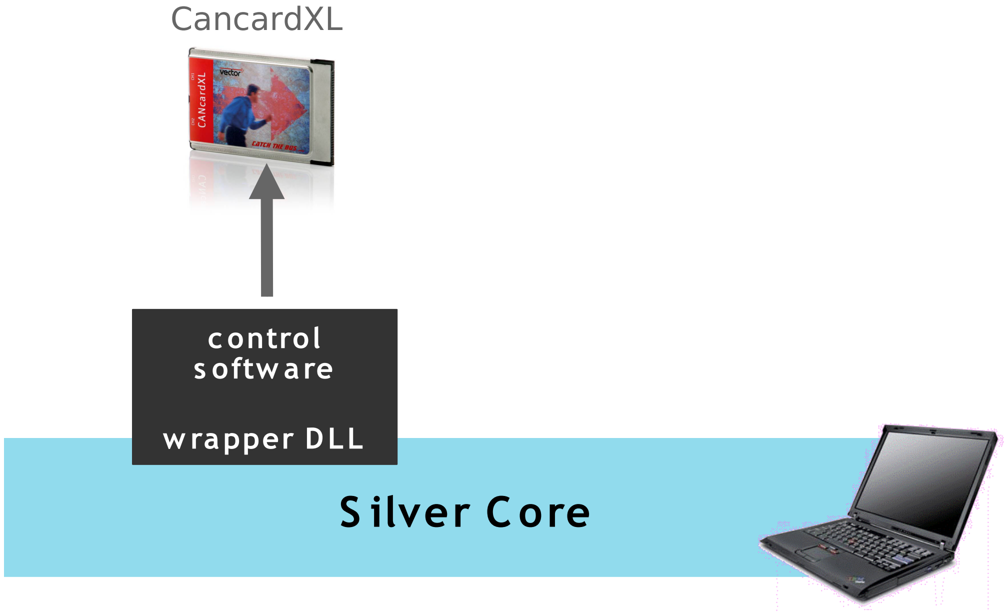
Rapid Prototyping

1 Silver simulation runs on a standard laptop:
- without graphical user-interface
- without simulation of the hardware (vehicle)
- with Canape and XCP via TCP/IP
- with wrapper DLL and entire control software



Rapid Prototyping

2 Wrapper DLL connects to CancardXL



CancardXL

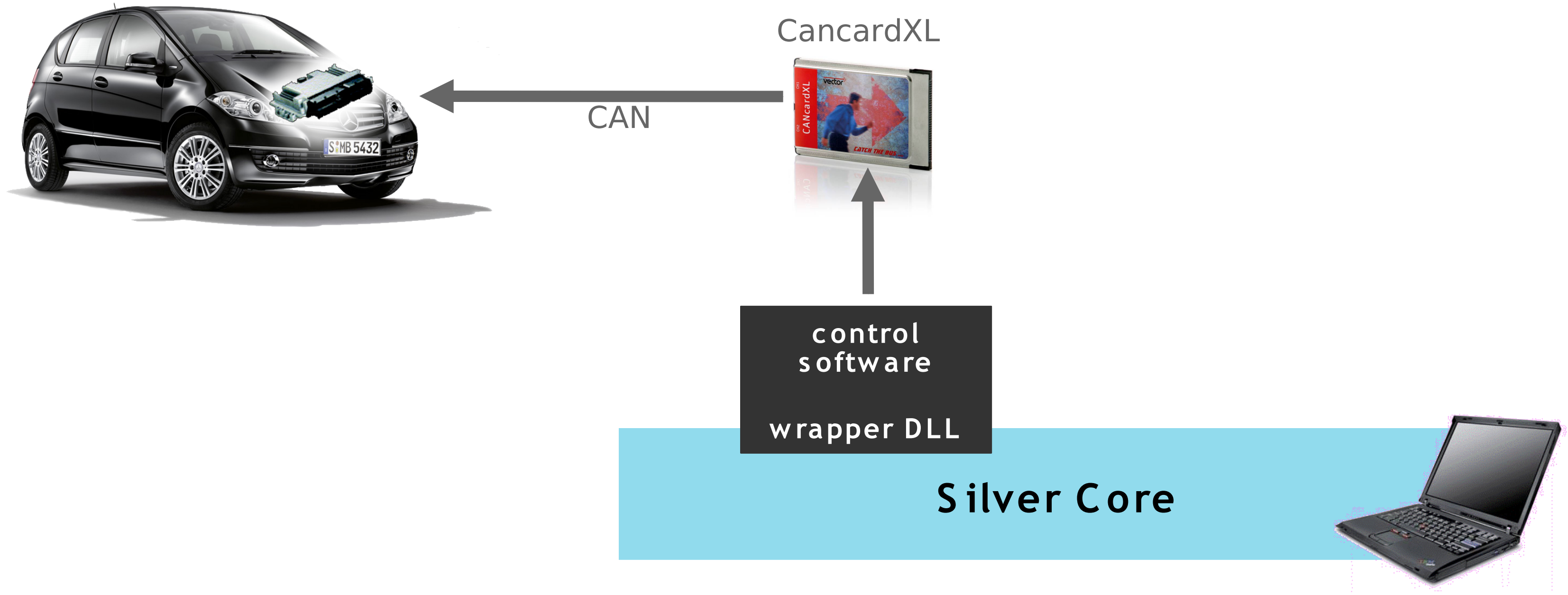
control software

wrapper DLL

Silver Core

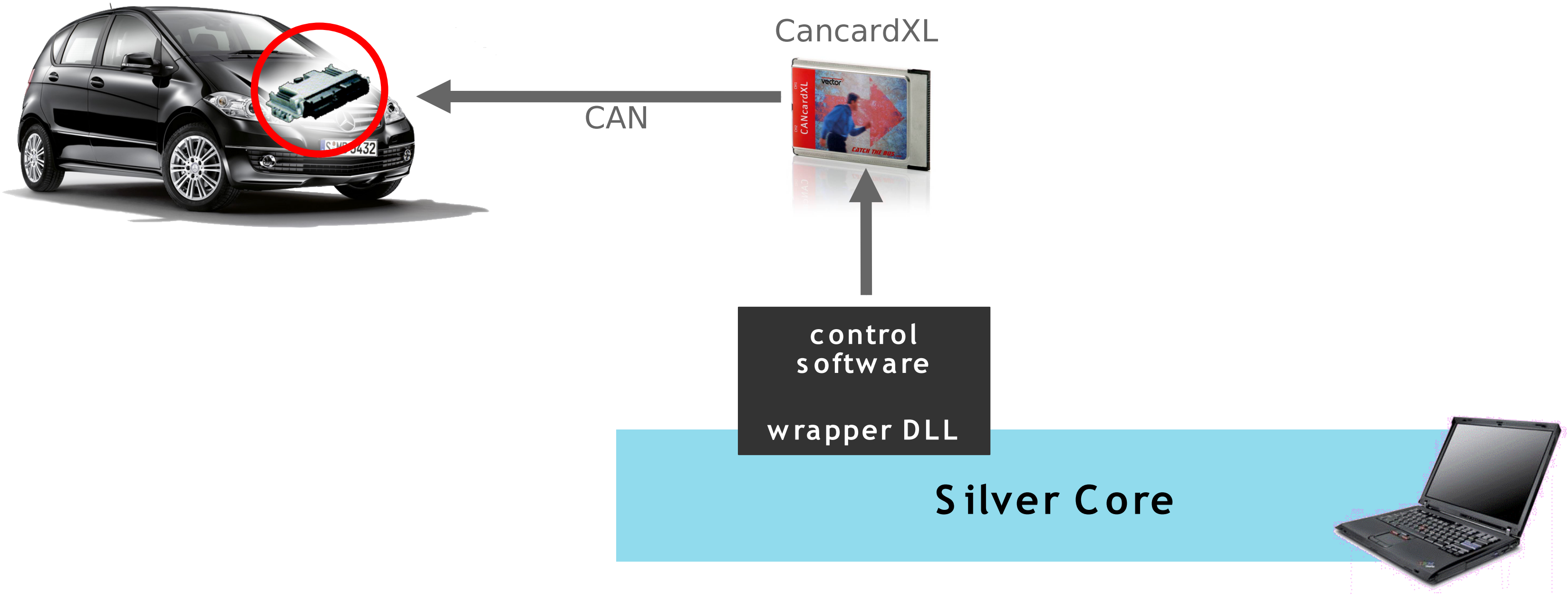
Rapid Prototyping

3 CancardXL connects to ECU in the vehicle via CAN



Rapid Prototyping

4 ECU in vehicle is set to bypass mode.
In bypass mode, the ECU overrIdes internally generated control signals by control signals received via CAN



CancardXL

CAN

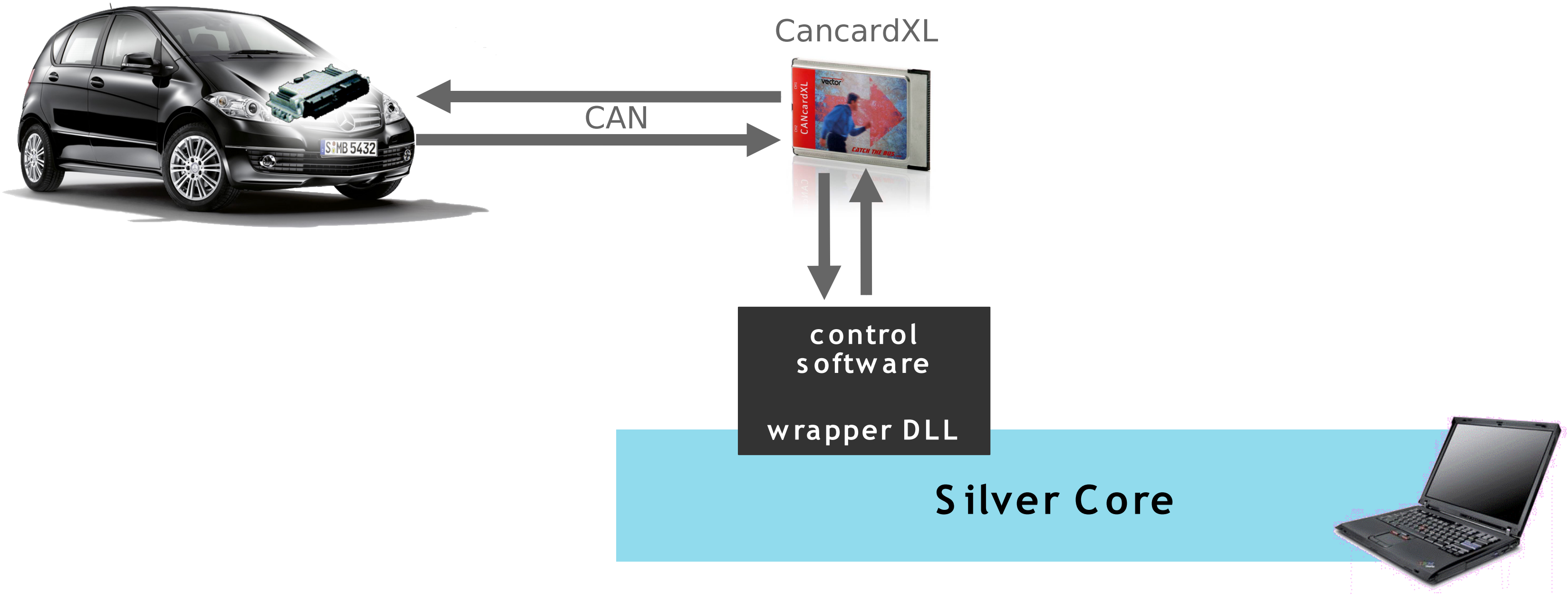
control software

wrapper DLL

Silver Core

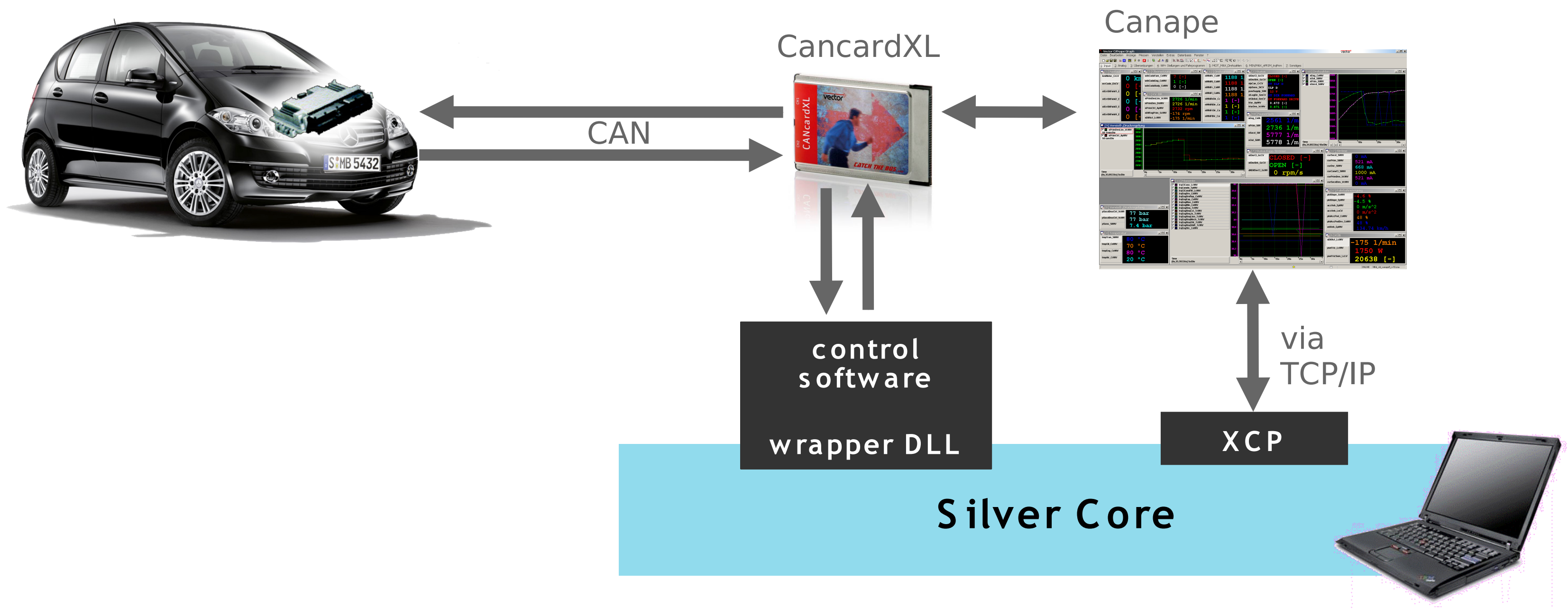
Rapid Prototyping

5 ECU in vehicle sends measured sensor values via CAN to Silver



Rapid Prototyping

6 Canape measures both, the control software internal signals via XCP, as well as ECU signals via CancardXL and CAN,



Outline of the talk

1 History and Motivation

2 DCT Development

3 Rapid Prototyping

4 Automated Testing

5 Code Coverage Analysis

6 Outlook

Objectives of automated testing

- higher quality and better validation of software before first use in a real car
- monitoring of application data, in addition to test using test rigs and continuous operation

This is achieved using

- many test scenarios, automatically generated in a controlled, intelligent way
- regression tests with simulation of continuous operation and scenario databases

Which errors are we looking for?

- runtime exceptions
- division by 0
- value out of bound w.r.t A2L
- access violation
- infinite loop

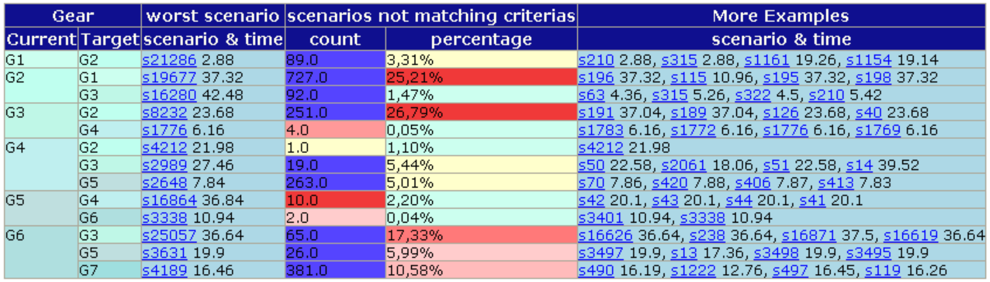
Range violations

- user-defined criteria
- overheating of components
- duration of gear shifts

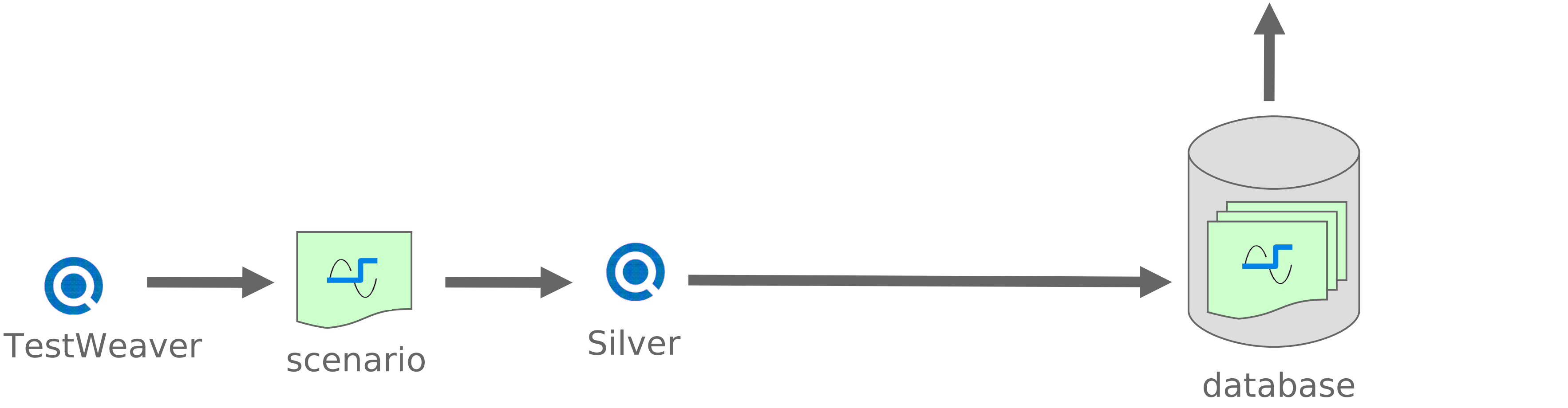
Automated testing using TestWeaver

1 Initial setup

- define inputs, outputs, report templates and good/bad criteria for assessing system behavior
- create Python-script for engine start



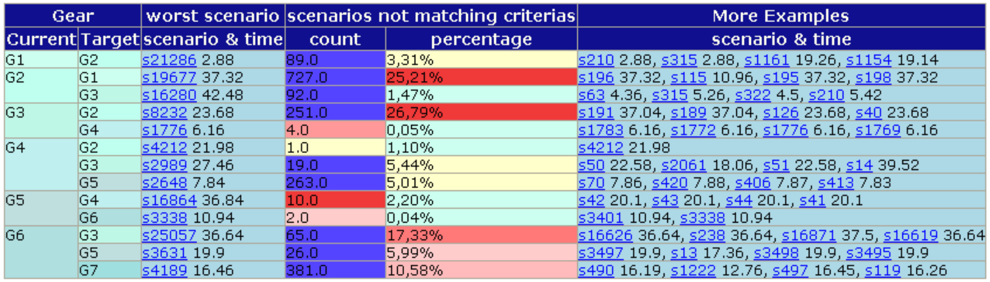
Gear		worst scenario	scenarios not matching criterias		More Examples
Current	Target	scenario & time	count	percentage	scenario & time
G1	G2	s21286 2.88	89.0	3,31%	s210 2.88, s315 2.88, s1161 19.26, s1154 19.14
G2	G1	s19677 37.32	727.0	25,21%	s196 37.32, s115 10.96, s195 37.32, s198 37.32
	G3	s16280 42.48	92.0	1,47%	s63 4.36, s315 5.26, s322 4.5, s210 5.42
G3	G2	s8232 23.68	251.0	26,79%	s191 37.04, s189 37.04, s126 23.68, s40 23.68
	G4	s1776 6.16	4.0	0,05%	s1783 6.16, s1772 6.16, s1776 6.16, s1769 6.16
G4	G2	s4212 21.98	1.0	1,10%	s4212 21.98
	G3	s2989 27.46	19.0	5,44%	s50 22.58, s2061 18.06, s51 22.58, s14 39.52
	G5	s2648 7.84	263.0	5,01%	s70 7.86, s420 7.88, s406 7.87, s413 7.83
G5	G4	s16864 36.84	10.0	2,20%	s42 20.1, s43 20.1, s44 20.1, s41 20.1
	G6	s3338 10.94	2.0	0,04%	s3401 10.94, s3338 10.94
G6	G3	s25057 36.64	65.0	17,33%	s16626 36.64, s238 36.64, s16871 37.5, s16619 36.64
	G5	s3631 19.9	26.0	5,99%	s3497 19.9, s13 17.36, s3498 19.9, s3495 19.9
	G7	s4189 16.46	381.0	10,58%	s490 16.19, s1222 12.76, s497 16.45, s119 16.26



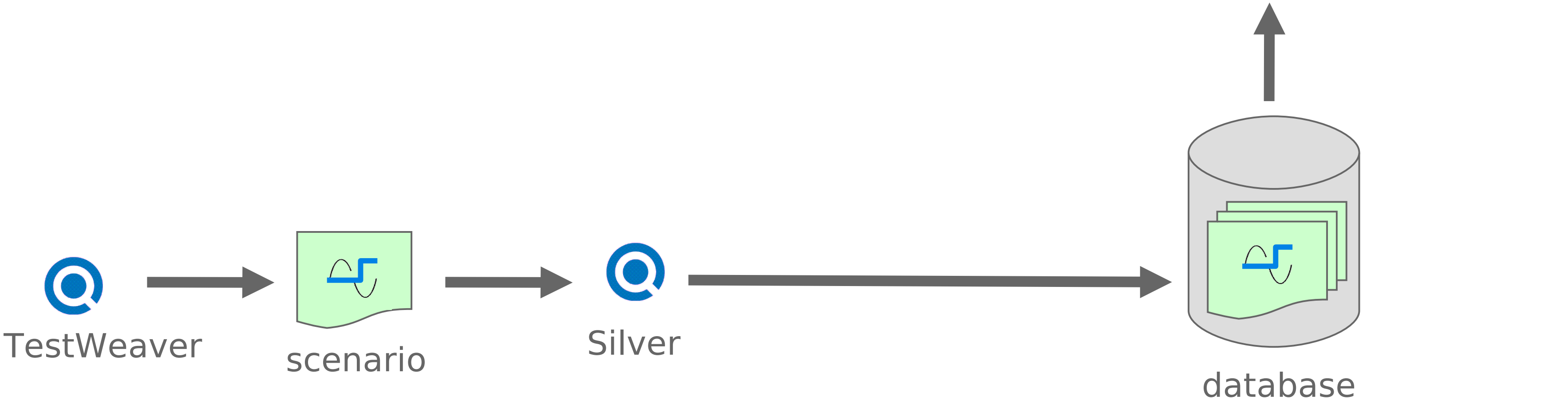
Automated testing using TestWeaver

2 Interface to Silver simulation enviornment

- For test, the same hardware and control software DLLs may be used, as for the SiL setup.
- TestWeaver starts and runs a Silver simulation
  for each generated scenario



Gear		worst scenario	scenarios not matching criterias		More Examples
Current	Target	scenario & time	count	percentage	scenario & time
G1	G2	s21286 2.88	89.0	3,31%	s210 2.88, s315 2.88, s1161 19.26, s1154 19.14
G2	G1	s19677 37.32	727.0	25,21%	s196 37.32, s115 10.96, s195 37.32, s198 37.32
	G3	s16280 42.48	92.0	1,47%	s63 4.36, s315 5.26, s322 4.5, s210 5.42
G3	G2	s8232 23.68	251.0	26,79%	s191 37.04, s189 37.04, s126 23.68, s40 23.68
	G4	s1776 6.16	4.0	0,05%	s1783 6.16, s1772 6.16, s1776 6.16, s1769 6.16
G4	G2	s4212 21.98	1.0	1,10%	s4212 21.98
	G3	s2989 27.46	19.0	5,44%	s50 22.58, s2061 18.06, s51 22.58, s14 39.52
	G5	s2648 7.84	263.0	5,01%	s70 7.86, s420 7.88, s406 7.87, s413 7.83
G5	G4	s16864 36.84	10.0	2,20%	s42 20.1, s43 20.1, s44 20.1, s41 20.1
	G6	s3338 10.94	2.0	0,04%	s3401 10.94, s3338 10.94
G6	G3	s25057 36.64	65.0	17,33%	s16626 36.64, s238 36.64, s16871 37.5, s16619 36.64
	G5	s3631 19.9	26.0	5,99%	s3497 19.9, s13 17.36, s3498 19.9, s3495 19.9
	G7	s4189 16.46	381.0	10,58%	s490 16.19, s1222 12.76, s497 16.45, s119 16.26



TestWeaver

scenario

Silver

database

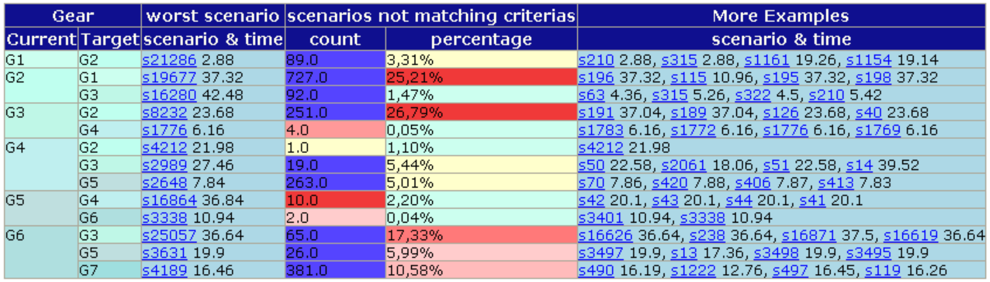
Automated testing using TestWeaver

3 Test!

a TestWeaver generates a scenario

b Silver runs the scenario,
remote controlled by TestWeaver

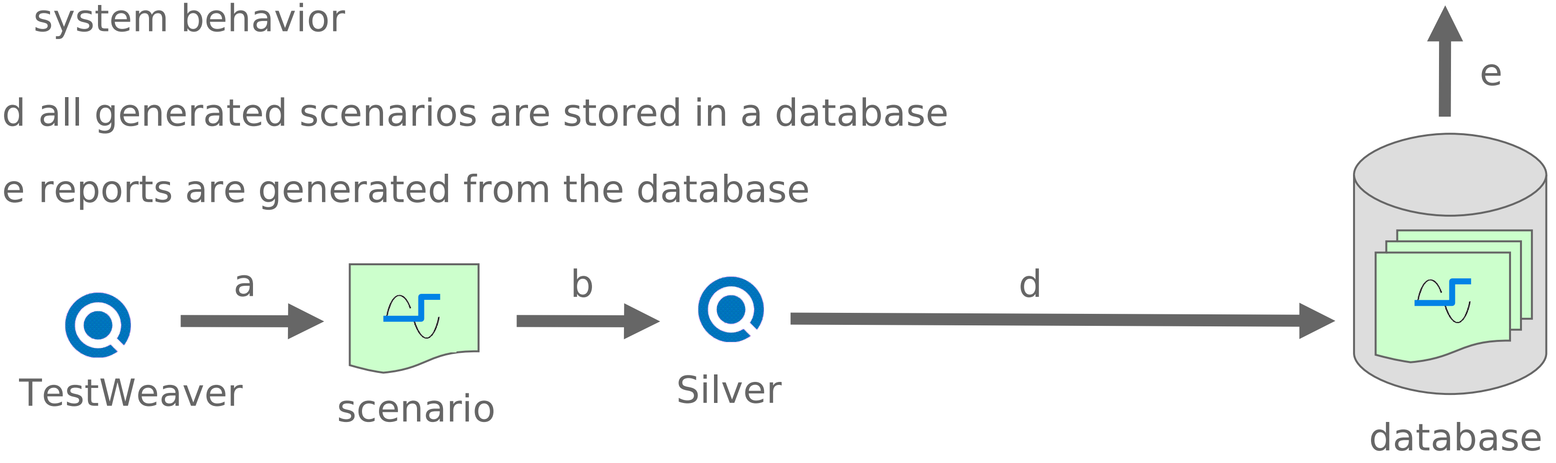
c if the scenario leads to suspicious or critical behavior, TestWeaver varies that scenario, in order to provoke hard errors and local worst case system behavior



Gear		worst scenario	scenarios not matching criterias		More Examples
Current	Target	scenario & time	count	percentage	scenario & time
G1	G2	s21286 2.88	89.0	3,31%	s210 2.88, s315 2.88, s1161 19.26, s1154 19.14
G2	G1	s19677 37.32	727.0	25,21%	s196 37.32, s115 10.96, s195 37.32, s198 37.32
	G3	s16280 42.48	92.0	1,47%	s63 4.36, s315 5.26, s322 4.5, s210 5.42
G3	G2	s8232 23.68	251.0	26,79%	s191 37.04, s189 37.04, s126 23.68, s40 23.68
	G4	s1776 6.16	4.0	0,05%	s1783 6.16, s1772 6.16, s1776 6.16, s1769 6.16
G4	G2	s4212 21.98	1.0	1,10%	s4212 21.98
	G3	s2989 27.46	19.0	5,44%	s50 22.58, s2061 18.06, s51 22.58, s14 39.52
	G5	s2648 7.84	263.0	5,01%	s70 7.86, s420 7.88, s406 7.87, s413 7.83
G5	G4	s16864 36.84	10.0	2,20%	s42 20.1, s43 20.1, s44 20.1, s41 20.1
	G6	s3338 10.94	2.0	0,04%	s3401 10.94, s3338 10.94
G6	G3	s25057 36.64	65.0	17,33%	s16626 36.64, s238 36.64, s16871 37.5, s16619 36.64
	G5	s3631 19.9	26.0	5,99%	s3497 19.9, s13 17.36, s3498 19.9, s3495 19.9
	G7	s4189 16.46	381.0	10,58%	s490 16.19, s1222 12.76, s497 16.45, s119 16.26

d all generated scenarios are stored in a database

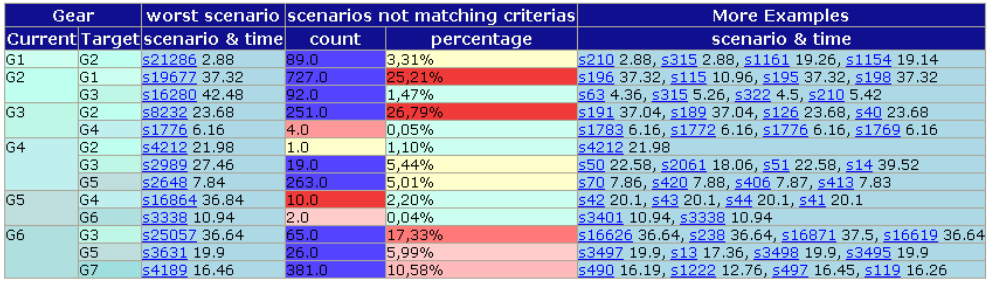
e reports are generated from the database



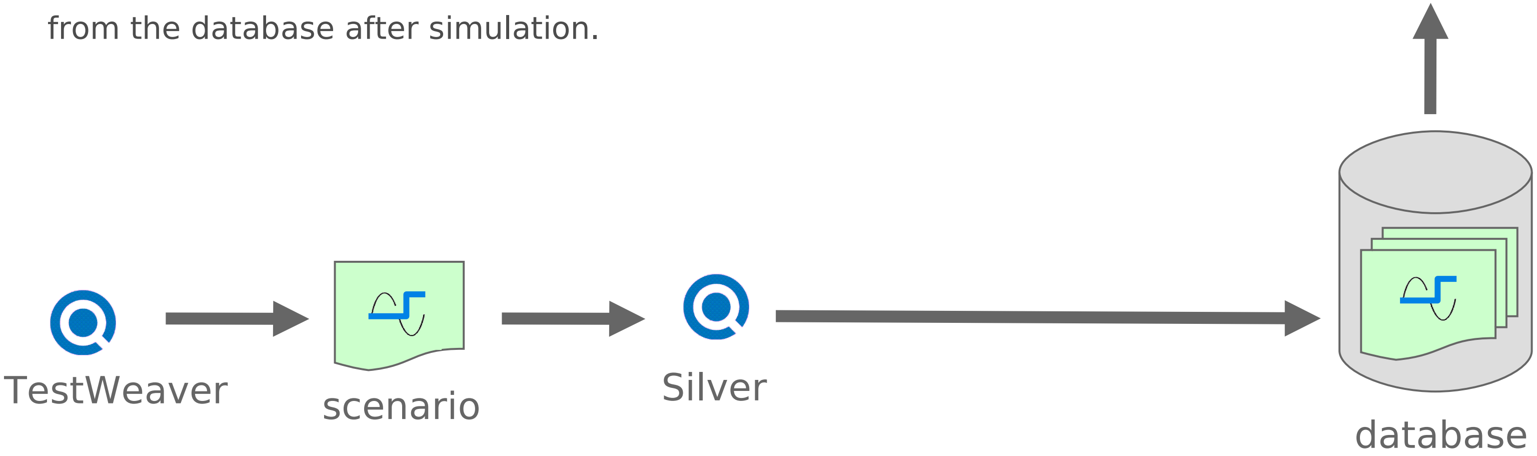
Automated testing using TestWeaver

Advantages:

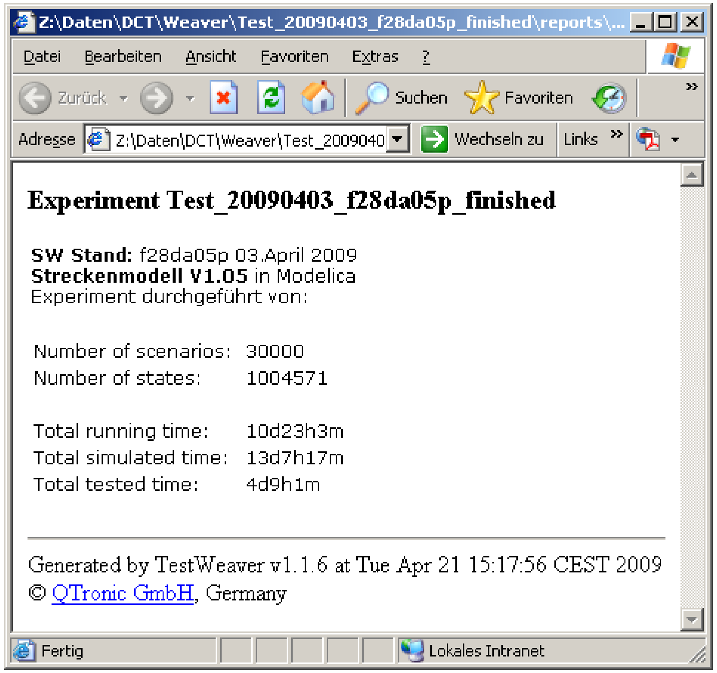
- seamless integration with the tool chain
- automated test case and scenario generation
- all scenarios can be reproduced in SiL
- support for debugging of all scenarios
- reports can be modified and updated from the database after simulation.



Gear		worst scenario	scenarios not matching criterias		More Examples
Current	Target	scenario & time	count	percentage	scenario & time
G1	G2	s21286 2.88	89.0	3,31%	s210 2.88, s315 2.88, s1161 19.26, s1154 19.14
G2	G1	s19677 37.32	727.0	25,21%	s196 37.32, s115 10.96, s195 37.32, s198 37.32
	G3	s16280 42.48	92.0	1,47%	s63 4.36, s315 5.26, s322 4.5, s210 5.42
G3	G2	s8232 23.68	251.0	26,79%	s191 37.04, s189 37.04, s126 23.68, s40 23.68
	G4	s1776 6.16	4.0	0,05%	s1783 6.16, s1772 6.16, s1776 6.16, s1769 6.16
G4	G2	s4212 21.98	1.0	1,10%	s4212 21.98
	G3	s2989 27.46	19.0	5,44%	s50 22.58, s2061 18.06, s51 22.58, s14 39.52
	G5	s2648 7.84	263.0	5,01%	s70 7.86, s420 7.88, s406 7.87, s413 7.83
G5	G4	s16864 36.84	10.0	2,20%	s42 20.1, s43 20.1, s44 20.1, s41 20.1
	G6	s3338 10.94	2.0	0,04%	s3401 10.94, s3338 10.94
G6	G3	s25057 36.64	65.0	17,33%	s16626 36.64, s238 36.64, s16871 37.5, s16619 36.64
	G5	s3631 19.9	26.0	5,99%	s3497 19.9, s13 17.36, s3498 19.9, s3495 19.9
	G7	s4189 16.46	381.0	10,58%	s490 16.19, s1222 12.76, s497 16.45, s119 16.26



Automated testing using TestWeaver


Z:\Daten\DCT\Weaver\Test_20090403_f28da05p_finished\reports\...
Datei Bearbeiten Ansicht Favoriten Extras ?
Zurück Suchen Favoriten
Adresse Z:\Daten\DCT\Weaver\Test_2009040 Wechseln zu Links
Fertig Lokales Intranet

Experiment Test_20090403_f28da05p_finished

SW Stand: f28da05p 03.April 2009
Streckenmodell V1.05 in Modelica
Experiment durchgeführt von:

```
Number of scenarios:  30000
Number of states:     1004571

Total running time:   10d23h3m
Total simulated time: 13d7h17m
Total tested time:    4d9h1m
```

Generated by TestWeaver v1.1.6 at Tue Apr 21 15:17:56 CEST 2009
© QTronic GmbH, Germany

Validation of a software release

- run at least 10.000 scenarios
- analyze reports and suspicious scenarios
- export critical scenarios to regression test database

Outline of the talk

1 History and Motivation

2 DCT Development

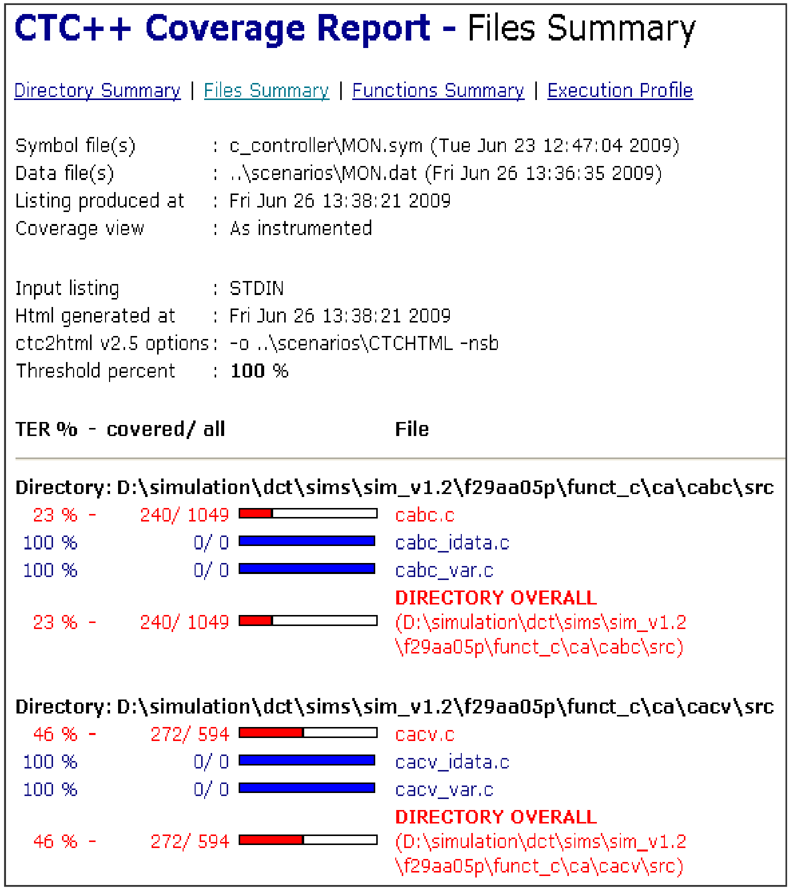
3 Rapid Prototyping

4 Automated Testing

5 Code Coverage Analysis

6 Outlook

Code Coverage Analysis with Testwell CTC++



CTC++ Coverage Report - Files Summary

Directory Summary | Files Summary | Functions Summary | Execution Profile

```
Symbol file(s)        : c_controller\MON.sym (Tue Jun 23 12:47:04 2009)
Data file(s)          : ..\scenarios\MON.dat (Fri Jun 26 13:36:35 2009)
Listing produced at   : Fri Jun 26 13:38:21 2009
Coverage view         : As instrumented

Input listing         : STDIN
Html generated at     : Fri Jun 26 13:38:21 2009
ctc2html v2.5 options : -o ..\scenarios\CTCHTML -nsb
Threshold percent     : 100 %

TER %  - covered/ all                 File

Directory: D:\simulation\dct\sims\sim_v1.2\f29aa05p\funct_c\ca\cabc\src
  23 % -    240/ 1049                 cabc.c
 100 %          0/ 0                  cabc_idata.c
 100 %          0/ 0                  cabc_var.c
                                      DIRECTORY OVERALL
  23 % -    240/ 1049                 (D:\simulation\dct\sims\sim_v1.2
                                      \f29aa05p\funct_c\ca\cabc\src)

Directory: D:\simulation\dct\sims\sim_v1.2\f29aa05p\funct_c\ca\cacv\src
  46 % -     272/ 594                 cacv.c
 100 %          0/ 0                  cacv_idata.c
 100 %          0/ 0                  cacv_var.c
                                      DIRECTORY OVERALL
  46 % -     272/ 594                 (D:\simulation\dct\sims\sim_v1.2
                                      \f29aa05p\funct_c\ca\cacv\src)
```

- integrated with TestWeaver
- separate report in TestWeaver
- coverage analysis for
  - entire project
  - C source file
  - functions
  - code path

Outline of the talk

1 History and Motivation

2 DCT Development

3 Rapid Prototyping

4 Automated Testing

5 Code Coverage Analysis

6 Outlook

Outlook: next steps

- further increase software quality
- increase code coverage
- simulation of continuous operation as regression test
- distributed simulation: software is simulated on multiple computers in parallel
- compare variants with each other
- build failure database with critical scenarios

Thank you for your attention!