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

**7G-Tronic** since 1998
- Rapid-Prototyping via Backbone
- Fix-point code simulation
- System tests
- Continuous operation simulations with fix-point code

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 transmission (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

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

Microsoft Visual Studio Compiler

dSpace code generator

cross compiler

C-code

object code for Infineon TriCore

object code for x86

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

- software for SiL
- A2L database, application parameter
- DLL for simulation
- objects for all 150 modules
Structure of the ECU software

Structure of the SiL software

The wrapper emulates the functions of the frame software.

Many wrapper functions simply return default values.

operating system
  • device driver
  • memory
  • ...

frame software
  • EEPROM
  • CAN data
  • ...

control software
  • 150 modules

wrapper
  + complete control software
  + CAN Data
  + EEPROM

- no error code memory
- no diagnostic functions
Software developer edits his module

generates code

compiles his module using Microsoft Visual Studio

get the other 149 modules as object code

links using Microsoft Visual Studio

control software with 150 modules

Working results:

• DLL for x86 PC containing the entire control software
• A2L database with addresses 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

A2L
software for SiL

 DLL wrapper

.Model-based Development of a Dual-Clutch Transmission using Rapid Prototyping and SiL, 01.07.09
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 manipulated
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 transmission
- developed in-house using Dymola

Configuration model

Dymola DLL

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 bandwidth as with CAN
- online calibration of parameters
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

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)

A2L and parameter:

• A2L with address information adapted to the DLL
• Complete and latest parameter values loaded at simulation start

Hardware-model

Canape

via TCP/IP

Dymola DLL

Configurable GUI

Silver Core

Reader

Writer

control software

wrapper DLL

PDB

PAR

DCM

HEX

A2L
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)

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

Canape

Simulation environment

Configurable GUI

Silver Core

Python

Test & Adaptation

Reader

Writer

PDB

PAR

DCM

HEX

A2L

Control software

Wrapper DLL

Dymola DLL

via TCP/IP

XCP
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.

Simulation environment tools:
- Simulation: Silver (QTronic)
- Measurement: Canape (Vector)
- Debugging: Visual Studio (Microsoft)
- Automated Test: TestWeaver (QTronic)
- Code Coverage: Testwell CTC++ (Verifysoft)
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
• Debugging
  • 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.
Rapid Prototyping

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

CancardXL

control software

wrapper DLL
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

<table>
<thead>
<tr>
<th>Gear</th>
<th>Current Target</th>
<th>worst scenario</th>
<th>scenario &amp; time</th>
<th>scenarios not matching criteria</th>
<th>percentage</th>
<th>More Examples</th>
<th>scenario &amp; time</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>02</td>
<td>z1000</td>
<td>z90.0</td>
<td>z2,31%</td>
<td>z1000</td>
<td>z90.0</td>
<td>z2,31%</td>
</tr>
<tr>
<td>02</td>
<td>01</td>
<td>z1067</td>
<td>z77.0</td>
<td>z58%</td>
<td>z1067</td>
<td>z77.0</td>
<td>z58%</td>
</tr>
<tr>
<td>03</td>
<td>02</td>
<td>z1250</td>
<td>z52.0</td>
<td>z1,47%</td>
<td>z1250</td>
<td>z52.0</td>
<td>z1,47%</td>
</tr>
<tr>
<td>04</td>
<td>03</td>
<td>z1776</td>
<td>z4.0</td>
<td>z0,05%</td>
<td>z1776</td>
<td>z4.0</td>
<td>z0,05%</td>
</tr>
<tr>
<td>05</td>
<td>04</td>
<td>z2176</td>
<td>z1.0</td>
<td>z0,01%</td>
<td>z2176</td>
<td>z1.0</td>
<td>z0,01%</td>
</tr>
<tr>
<td>06</td>
<td>05</td>
<td>z2564</td>
<td>z1.0</td>
<td>z0,04%</td>
<td>z2564</td>
<td>z1.0</td>
<td>z0,04%</td>
</tr>
<tr>
<td>07</td>
<td>06</td>
<td>z3336</td>
<td>z2.0</td>
<td>z0,02%</td>
<td>z3336</td>
<td>z2.0</td>
<td>z0,02%</td>
</tr>
<tr>
<td>08</td>
<td>07</td>
<td>z5831</td>
<td>z1.0</td>
<td>z0,09%</td>
<td>z5831</td>
<td>z1.0</td>
<td>z0,09%</td>
</tr>
</tbody>
</table>
Automated testing using TestWeaver

2 Interface to Silver simulation environment

- 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.

<table>
<thead>
<tr>
<th>Gear</th>
<th>Case</th>
<th>Result</th>
<th>Percentage</th>
<th>More Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>G2</td>
<td>2.126</td>
<td>2.31%</td>
<td>310 2.99, 315 2.08, 3161 19.26, 3154 19.14</td>
</tr>
<tr>
<td>02</td>
<td>G1</td>
<td>3.1672</td>
<td>312.4%</td>
<td>318 37.93, 3118 10.99, 3163 37.92, 3128 37.32</td>
</tr>
<tr>
<td>03</td>
<td>G2</td>
<td>3.1236</td>
<td>2.47%</td>
<td>314 31.5, 3135 2.55, 3234 2.45, 3228 3.45</td>
</tr>
<tr>
<td>04</td>
<td>G2</td>
<td>3.0022</td>
<td>2.50%</td>
<td>3101 2.70, 3193 2.70, 3126 2.68, 3120 2.68</td>
</tr>
<tr>
<td>05</td>
<td>G4</td>
<td>3.177</td>
<td>0.05%</td>
<td>3177 6.16, 3177 6.16, 3177 6.16, 3177 6.16</td>
</tr>
<tr>
<td>06</td>
<td>G2</td>
<td>3.2112</td>
<td>1.0%</td>
<td>3211 2.95</td>
</tr>
<tr>
<td>07</td>
<td>G3</td>
<td>3.086</td>
<td>3.44%</td>
<td>3086 2.01, 3020 1.88, 3019 2.28, 314 39.52</td>
</tr>
<tr>
<td>08</td>
<td>G5</td>
<td>3.264</td>
<td>2.01%</td>
<td>3264 7.08, 3120 7.08, 3105 7.07, 3113 7.03</td>
</tr>
<tr>
<td>09</td>
<td>G4</td>
<td>3.1664</td>
<td>2.00%</td>
<td>3166 2.00, 3166 2.00, 3166 2.00, 3166 2.00</td>
</tr>
<tr>
<td>10</td>
<td>G3</td>
<td>3.3338</td>
<td>0.04%</td>
<td>3333 10.94, 3333 10.94</td>
</tr>
<tr>
<td>11</td>
<td>G4</td>
<td>3.333</td>
<td>0.01%</td>
<td>3333 10.94, 3333 10.94</td>
</tr>
<tr>
<td>12</td>
<td>G6</td>
<td>3.3557</td>
<td>2.35%</td>
<td>3357 25.56, 3357 25.56, 31571 37.75, 31571 36.64</td>
</tr>
<tr>
<td>14</td>
<td>G7</td>
<td>3.4106</td>
<td>0.00%</td>
<td>3410 16.19, 3410 16.19, 3410 16.19, 3410 16.19</td>
</tr>
</tbody>
</table>

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

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.

<table>
<thead>
<tr>
<th>Gear</th>
<th>Current</th>
<th>Target</th>
<th>worst scenario scenario &amp; time</th>
<th>scenario &amp; time</th>
<th>count</th>
<th>percentage</th>
<th>worst scenario scenario &amp; time</th>
<th>scenario &amp; time</th>
<th>More Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>G2</td>
<td></td>
<td>3100 2.00</td>
<td>3100 2.00</td>
<td>1.0</td>
<td>2.01%</td>
<td>3100 2.00</td>
<td>3100 2.00</td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>G1</td>
<td>1106</td>
<td>37.32</td>
<td>37.32</td>
<td>1.0</td>
<td>3.41%</td>
<td>3106 37.32</td>
<td>3106 37.32</td>
<td></td>
</tr>
<tr>
<td>03</td>
<td>G1</td>
<td>1106</td>
<td>45.40</td>
<td>45.40</td>
<td>5.0</td>
<td>1.47%</td>
<td>3106 45.40</td>
<td>3106 45.40</td>
<td></td>
</tr>
<tr>
<td>04</td>
<td>G2</td>
<td>3000</td>
<td>23.68</td>
<td>23.68</td>
<td>5.0</td>
<td>22.79%</td>
<td>3000 23.68</td>
<td>3000 23.68</td>
<td></td>
</tr>
<tr>
<td>05</td>
<td>G4</td>
<td>1776</td>
<td>6.16</td>
<td>6.16</td>
<td>20.0</td>
<td>0.05%</td>
<td>1776 6.16</td>
<td>1776 6.16</td>
<td></td>
</tr>
<tr>
<td>06</td>
<td>G2</td>
<td>3000</td>
<td>23.68</td>
<td>23.68</td>
<td>20.0</td>
<td>2.00%</td>
<td>3000 23.68</td>
<td>3000 23.68</td>
<td></td>
</tr>
<tr>
<td>07</td>
<td>G2</td>
<td></td>
<td>3338 10.94</td>
<td>10.94</td>
<td>5.0</td>
<td>0.04%</td>
<td>3338 10.94</td>
<td>3338 10.94</td>
<td></td>
</tr>
<tr>
<td>08</td>
<td>G3</td>
<td>3000</td>
<td>23.68</td>
<td>23.68</td>
<td>65.0</td>
<td>37.03%</td>
<td>3000 23.68</td>
<td>3000 23.68</td>
<td></td>
</tr>
</tbody>
</table>

TestWeaver → scenario → Silver → database
Automated testing using TestWeaver

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

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>History and Motivation</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>DCT Development</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Rapid Prototyping</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Automated Testing</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Code Coverage Analysis</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Outlook</td>
<td></td>
</tr>
</tbody>
</table>
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) : .\scenario\MON.dat (Fri Jun 26 13:38:21 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
cct2html v2.8 options : -o .\scenario\CTCHTML -nbo
Threshold percent : 100 %

TER % - covered/all File

Directory: D:\simulation\dct\sim\sim_v1.2\29aa05p\funct_c\ca\cabc\src
23 % - 240/1040 cabc.c
100 % 0/0 cabc_data.c
100 % 0/0 cabc_var.c

DIRECTORY OVERALL
(D:\simulation\dct\sim\sim_v1.2\29aa05p\funct_c\ca\cabc\src)

Directory: D:\simulation\dct\sim\sim_v1.2\29aa05p\funct_c\ca\cacv\src
46 % - 272/594 cacv.c
100 % 0/0 cacv_data.c
100 % 0/0 cacv_var.c

DIRECTORY OVERALL
(D:\simulation\dct\sim\sim_v1.2\29aa05p\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!