Simulation-based development of automotive control software with Modelica

Emmanuel Chrisofakis, Dr. Anton Rink, Daimler AG
Dr. Andreas Junghanns, QTronic GmbH
Christian Kehrer, ITI GmbH

8th International Modelica Conference, March 2011, Dresden
Contents

- Software in the loop simulation at Daimler
- SIL-environment / functionality
- Plant model
- Model compatibility
- Integration of a SIL-Project
- Summary
Software in the loop simulation at Daimler

Application area

- Testing and deployment of functional code
- Version update safeguarding of functional code
- „Desktop“-application / -calibration
- Fault simulation
- Virtual endurance testing
  - safeguarding of drivetrain components
  - calculation of load collectives for gearbox and drivetrain

Requirements

- Powerful, stable and fast simulation environment
- Easy to use by any engineer

Tool chain

SIL-tool
- Backbone (in house development)
- Silver (QTronic GmbH)

Plant model
- MSL 2.2 in Dymola 6.2 (Dassault Systèmes)
- In the future MSL > 3.1 with Dymola from v. 7.4 or SimulationX from v. 3.4 (ITI GmbH)

Test generator
- TestWeaver (QTronic GmbH)

Software integration platform
- Microsoft Visual Studio 2005 or 2008
SIL-environment / functionality

- The simulation is controlled by a special program (e.g. Silver) which guards the single modules.
- Every module (called „Client“) sends its Outputs to Backbone and reads its Inputs from him, i.e. no direct communication between the modules occurs (except for the CANape-coupling with the control software).
- The communication step time is fixed and represents the lowest task time step of the functional code (5, 10 or 20 ms).
- The plant model is wrapped with a numerical solver which calculates with smaller time steps.
- Backbone waits after each communication step until all clients are finished so that the next step will be initiated (slow model slows down the simulation).

- Every „Client“ must be available in C-Code or be pre-compiled (obj-file).
- The integration in the SIL environment takes place by “wrapping” the C-code with the desired API (backbone or Silver).
Plant model

**Requirement:** accurate calculation of gear shifting
- Filling and draining of clutch pistons
- Detailed representation of piston mechanics
- Calculation of the gearbox kinematics including the impact of it to the internal inertia

**Description**
- The plant model describes the torsional (1-D) and translational (1-D) dynamics of an entire vehicle.
- The modeling focus lies on the detailed description of the gearbox (piston mechanism, coefficient of friction, filling and draining of pistons, etc.). It is about the 7-gear planetary automatic transmission of Mercedes-Benz (7G-Tronic).
- The engine model is descriptive (look-up table characteristics) and includes an idle speed controller as well as the functionality to manipulate the engine torque during the gear shifting.
- The drag forces are calculated in the vehicle model.
- The model is cut out for the SIL environment and in this form, it is designated for the SIL-export.
Plant model: Modelica Libraries

For the creation of the plant model, own devised libraries and standard Modelica components have been used.

**Car-Library** includes basic models for building hydraulic and mechanical structures (e.g. orifice, valve edge, planetary gear, parking lock, etc.)

**AtNag2-Library** includes and describes transmission specific models such as hydraulic control unit, mechanical model, clutches and brakes, etc.

The libraries **Car** and **AtNag2** were originally created in Dymola 6 with Modelica 2.2
Plant model: mechanics

Hydraulic interface (from the hydraulic control unit)

All independent inertias of the gearbox are explicitly modeled.

Parameter masks of a clutch model
Plant model: hydraulics (control unit)

The electro-hydraulic control model has been modeled phenomenologically (control logic, no dynamics) for the sake of simulation performance. However, many components such as orifices, shifting valves, fluid volumes etc. have a physical model description in order to accurately describe important effects in the simulation (filling, draining, pressure switch, sticking valves etc.)
Model compatibility

- The model is compatible to both Dymola 7.4 as well as SimulationX 3.4 as long as MSL 3.1 is used
- Existing models based on older MSL versions have to be “upgraded”
- Once this job is done, no further adjustment is necessary and the models can be easily loaded in both software tools (Dymola and SimulationX)
Integration of a SIL-Project

**SiL environment**
- Simulation: Silver (QTronic)
- Measurement: CANape (Vector)
- Debugging: Visual Studio (Microsoft)
- Automated Test: TestWeaver (QTronic)
- Code Coverage: CTC++ (Verifysoft)
Integration of a SIL-Project

**SiL environment**
- Simulation: Silver (QTronic)
- Measurement: CANape (Vector)
- Debugging: Visual Studio (Microsoft)
- Automated Test: TestWeaver (QTronic)
- Code Coverage: CTC++ (Verifysoft)

**graphical user-interface:**
- interaction of driver/user with simulated car
- accel pedal, steering, etc. can be controlled
- plotter, breakpoints, scripting, file in/out, ...

Configurable GUI

Silver
Integration of a SIL-Project

**SiL environment**
- Simulation: Silver (QTronic)
- Measurement: CANape (Vector)
- Debugging: Visual Studio (Microsoft)
- Automated Test: TestWeaver (QTronic)
- Code Coverage: CTC++ (Verifysoft)

**hardware DLL:**
- simulated vehicle, engine and transmission
- Dymola/SimulationX
Integration of a SIL-Project

**SiL environment**
- Simulation: Silver (QTronic)
- Measurement: CANape (Vector)
- Debugging: Visual Studio (Microsoft)
- Automated Test: TestWeaver (QTronic)
- Code Coverage: CTC++ (Verifysoft)

**XCP with Canape/INCA:**
- XCP measurements via TCP/IP
- no limitation of bandwidth as with CAN
- online calibration of parameters
Integration of a SIL-Project

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

ECU control software as DLL:
- entire ECU control software
- frame software emulated by wrapper

Configurable GUI

Silver

Plant model

FMU

CANape

via TCP/IP

XCP

control software wrapper DLL
Integration of a SIL-Project

**SiL environment**
- Simulation: Silver (QTronic)
- Measurement: CANape (Vector)
- Debugging: Visual Studio (Microsoft)
- Automated Test: TestWeaver (QTronic)
- Code Coverage: CTC++ (Verifysoft)

**A2L and parameter:**
- A2L with address information adapted to the DLL
- Parameter values loaded at simulation start
Integration of a SIL-Project

**SiL environment**
- Simulation: Silver (QTronic)
- Measurement: CANape (Vector)
- Debugging: Visual Studio (Microsoft)
- Automated Test: TestWeaver (QTronic)
- Code Coverage: CTC++ (Verifysoft)

**Scripting with Python:**
- automate frequently used procedures (e.g. engine start, adaptation procedure etc.)
- implement control tasks (e.g. driver behaviour)

**Configurable GUI**

**Silver**
- reader
- writer
- control software wrapper DLL

**Plant model**

**CANape**
- via TCP/IP
- XCP

**FMU**

**Tests & Adaptation**

**Python**

**Tests & Adaptation**
Integration of a SIL-Project

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

Debugging with Visual Studio:
- suspend simulation at any time
- attach Visual Studio Debugger to Silver

Silver
- Configurable GUI
- Python
- Tests & Adaptation
- PAR
- DCM
- HEX
- A2L
- Reader
- Writer
- control software
- wrapper DLL
- XCP
- CANape
- via TCP/IP
Summary

- SIL is an essential tool in the gearbox development at Daimler
- For the creation of the SIL plant model, Dymola (MSL 2.x) has been used
- Upgrade of the model to MSL 3.1 enables full compatibility to SimulationX v. 3.4
- For the plant model export to SIL the new Modelisar-FMI can be applied
- SIL integration of the functional code (TCU) is done by wrapping the original code with the Silver-API and emulating the frame software
- Silver offers the possibility to measure and calibrate TCU-internal signals either directly in the Silver GUI or by coupling to commercial calibration tools such as CANape or INCA
- The functional code can be easily debugged by using the features of MS-Visual-Studio
- The utilisation of SIL during the development process leads to accurate code coming along with essential development cost reduction