Sviluppo di un sistema di sospensioni semiattive mediante Model-Based Design con architettura AUTOSAR e conforme allo standard A-SPICE

Presented by:
Andrea Palazzetti

Milano
25/06/2019

Roma
26/06/2019
Marelli - Ride Dynamics

• Marelli – Ride Dynamics – Mechatronic team
  ➢ Design and development of semi-active suspensions system
  ➢ Responsible for the whole system

• Mechatronic’s team is based in Turin

• ECU Application Software development
  ➢ Shock Absorber damping force control strategies and diagnosis
Smart Damping Control System

• SDC system consists of
  ➢ 4 shock absorbers with one proportional EV each
  ➢ 5 accelerometers
  ➢ ECU for closed loop damping control
Key Takeaways

➢ “State of the art”: AUTOSAR and A-SPICE development process

➢ Short time to market

➢ Focus on bidirectional traceability

➢ One single development environment for all SW related processes
Software development: goals and challenges

• State of-the-art for embedded automotive application software
  • Model-Based Design and automatic code generation
  • AUTOSAR Software architecture
  • Development process compliant to A-SPICE reference model

• Such a development process and SW architecture are required by main OEMs

• Constraint: Short time to market
What is AUTOSAR?

AUTOSAR — AUTomotive Open Systems ARchitecture

Middleware and system-level standard, jointly developed by automobile manufacturers, electronics and software suppliers and tool vendors.
More than 100 members

Motto: “cooperate on standards, compete on implementations”
Reality: current struggle between OEM and Tier1 suppliers

Target: facilitate portability, composability, integration of SW components over the lifetime of the vehicle
AUTOSAR ECU SW architecture
Automotive SPICE process reference model

Focus on Software development
Subset of recommended A-SPICE base practices

• Specify software requirements
• Structure software requirements

- Establish **bidirectional traceability** between
  - software and system requirements
  - software requirements and software architectural element
  - software requirements and software units
  - software detailed design and the unit test specification
  - elements of the software architectural design and test cases
  - software qualification test specification and software qualification test results

• Develop a detailed design for each software component
• Define interfaces of software elements
• Define interfaces of software units.
Whole SW development tool set based on MATLAB & Simulink R2018a

Software Engineering Process Group (SWE)

- SWE.1 Software Requirements Analysis
- SWE.2 Software Architectural Design
- SWE.3 Software Detailed Design and Unit Construction
- SWE.4 Software Unit Verification
- SWE.5 Software Integration and Integration Test
- SWE.6 Software Qualification Test

Simulink Requirements: requirements specification
Whole SW development tool set based on MATLAB & Simulink R2018a

- **Simulink – Stateflow**: SW units design and simulation
- **Simulink Check and Design Verifier**: coding guidelines check - Simulink model analysis
- **Embedded Coder - Support package for Autosar**: SW Components’ AUTOSAR interfaces design and C-code autogeneration
Whole SW development tool set based on MATLAB & Simulink R2018a

- **Simulink Test**: Unit testing – MIL testing
- **Simulink Coverage**: for testing coverage metrics
Subset of recommended A-SPICE base practices

- Specify software requirements
- Structure software requirements

- Establish bidirectional traceability between
  - software and system requirements
  - software requirements and software architectural element
  - software requirements and software units
  - software detailed design and the unit test specification
  - elements of the software architectural design and test cases
  - software qualification test specification and software qualification test results

- Develop a detailed design for each software component
- Define interfaces of software elements
- Define interfaces of software units
Requirements’ structure: three levels

Level 0: System

Level 1: ECU

Level 2: SWC

System requirements

ECU SW requirements

Application software requirements

Basic/Boot software requirements

Application SW component’s requirements

SWC1

SWC2

SWC3

SWC#n

SWC1 Req

SWC2 Req

SWC3 Req

SWC#n Req
SW requirements specification

➢ Simulink Requirements is used for requirements specification and linking
➢ Several “Requirement sets” used for grouping requirements
➢ One requirement set for every SW Component
Requirements set: example
Bidirectional traceability: Simulink Requirements view

- Requirement specification
- Additional information

**BIDIRECTIONAL LINKS**
- link to implementation Simulink model
- link to verification harness model

MATLAB EXPO 2019
Subset of recommended A-SPICE base practices

- Specify software requirements
- Structure software requirements
- Establish bidirectional traceability between
  - software and system requirements
  - software requirements and software architectural element
  - software requirements and software units
  - software detailed design and the unit test specification
  - elements of the software architectural design and test cases
  - software qualification test specification and software qualification test results
- Develop a detailed design for each software component
  - Define interfaces of software elements
  - Define interfaces of software units.
AUTOSAR INTERFACES design: BOTTOM-UP APPROACH

Simulink environment

AUTOSAR design (Meta-model)

Import SWC Description
Merge SWC Description

Export SWC Description/Generate SWC C code
Adding and mapping an AUTOSAR PORT
Detailed design

Bidirectional linking to implemented requirement

#1: CAN Inputs - Longitudinal acceleration calculation

MATLAB EXPO 2019
Data Dictionary specifies: tuneable parameters, measurable variables, constants, bus object ...

Example: measurable variable
Model Advisor: Model check before code generation

Modeling standards: MAAB

Automatic report

MAAB rules automatically checked

MATLAB EXPO 2019
Unit testing: Harness Model

Simulink Test automatically generates the HARNESS MODEL

Input test patterns

SWC’s Runnable #n

Expected outputs

Actual outputs

SW UNI under test

Automatic Check
actual results — expected result

result

MATLAB EXPO 2019
Harness model is linked to the SW unit.
Harness Model: example

Input test patterns

Assessment block: expected result evaluation
Simulink Test - Test Manager

Link to requirement under test

- CAN Inputs - Longitudinal acceleration calculation (caninputs_req_fewsdoc#1)

- SYSTEM UNDER TEST
  - Model: caninputs
  - TEST HARNESS
Unit Testing status: example

Simulink Requirements: overall view
Embedded Coder: code generation

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Accumulated Stack Size (bytes)</th>
<th>Self Stack Size (bytes)</th>
<th>Lines of Code</th>
<th>Lines</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanInputsRun10CanInputProc</td>
<td>70</td>
<td>70</td>
<td>370</td>
<td>1009</td>
<td>38</td>
</tr>
<tr>
<td>CanInputsRun10CanDiagnosis</td>
<td>14</td>
<td>14</td>
<td>55</td>
<td>145</td>
<td>6</td>
</tr>
<tr>
<td>CanInputs_Runnable_Init</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

BEGIN MEASUREMENT
/* Name */ InCan_O_AxV
/* Long Identifier */ "Vehicle longitudinal acceleration"
/* Data type */ FLOAT32_IEEE
/* Conversion method */ caninputs_CM_single_m_s2
/* Resolution (Not used) */ 0
/* Accuracy (Not used) */ 0
/* Lower Limit */ -3.4E+38
/* Upper Limit */ 3.4E+38
ECU_ADDRESS 0x8000 /* @ECU_Address@InCan_O_AxV */ */
Embedded Coder: code generation

Bidirectional link between SW element and C-code

```c
if (Can_rtB_Compare dw > 0) {
    InCan_0_AxV = 0.0F;
} else {
    InCan_0_AxV = 0.0299599504F * (real32_T)InCan_1_Acceleration_X + -21.593F;
}
```
MIL testing

Testing of the whole application layer: level 1 requirements

Plant model

SW-Cs composition
MIL testing

➢ Function callers blocks are used for simulating AUTOSAR S/R and C/S ports
➢ It is not needed to configure models for MIL
➢ Same models used for code generation are able to run even in MIL environment

AUTOSAR’s interface simulation
MIL testing example: fault injection
Achievements and Outlook

• ECU SW put in production in April 2019

• 18 months of development

• Technical, organizational and business results.
  • The standardization of development environment and the “bottom up” approach has increased the cross-competence inside the SW team
  • No need of other tools for AUTOSAR architecture design as regards to application SWCs
  • One single data base for requirements, software models, code and testing results
  • Cutting of time needed for documentation since it is automatically generated
Achievements and Outlook

• Integrated toolchain based on Simulink environment for SW development made traceability easier to achieve

• Use of the tool’s standard features only, avoiding customization (scripts) made the toolchain lean and easier to update

• Bottom – up approach made AUTOSAR SW components design quicker
Forward-looking plans

• Use of new and upcoming MathWorks tools as System Composer for

➢ System design in accordance with A-SPICE requirements
THANK YOU FOR YOUR ATTENTION