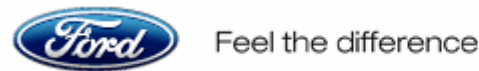


AVS: A Test Suite for Automatically Generated Code

Ekkehard Pofahl
Ford Motor Company



Torsten Sauer
Continental Automotive Systems



Oliver Busa
TUV Rheinland Industrie Service GmbH



Feel the difference

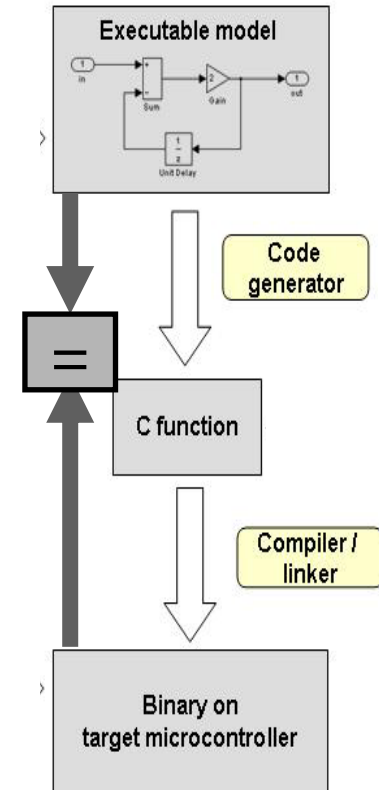


AVS: Automotive Code Validation Suite

Safeguarding the Code Generation Tool Chain

Objectives:

- **Demonstrate:** That C code corresponds to graphical model as generated from design in Simulink[®] and Stateflow[®]
- **Validate / Verify (V&V):** The quality and robustness through dynamic testing of the code generation down the tool chain (code generator / compiler / linker / target)
- **Generate:** Meaningful test cases for automatic execution in a constantly growing test suite
- **Support:** Maintaining quality levels through Regression testing



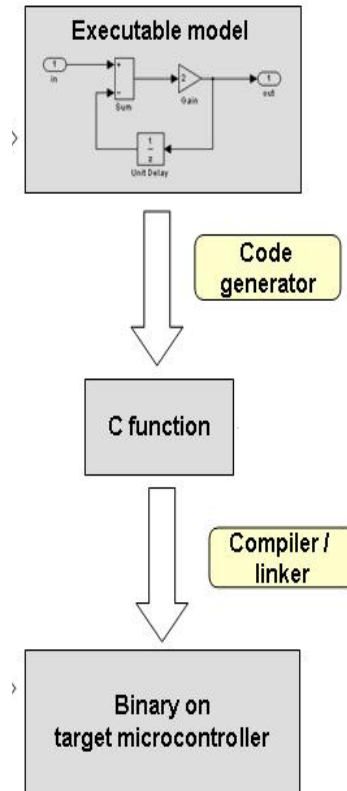
AVS: Automotive Code Validation Suite

Project History

- Joint project of Ford / ContiAutomotive / TUV Rheinland
- Originated from a Compiler Validation Project
- Work started in early 2002
- There have been three major versions since 2002
- Supports different code generator / compiler / linker tool chains, e.g., dSPACE TargetLink and The MathWorks Real-Time Workshop[®] Embedded Coder
- Meaning of the acronym AVS changed during the project



Automatic Code Generation Tool Chain

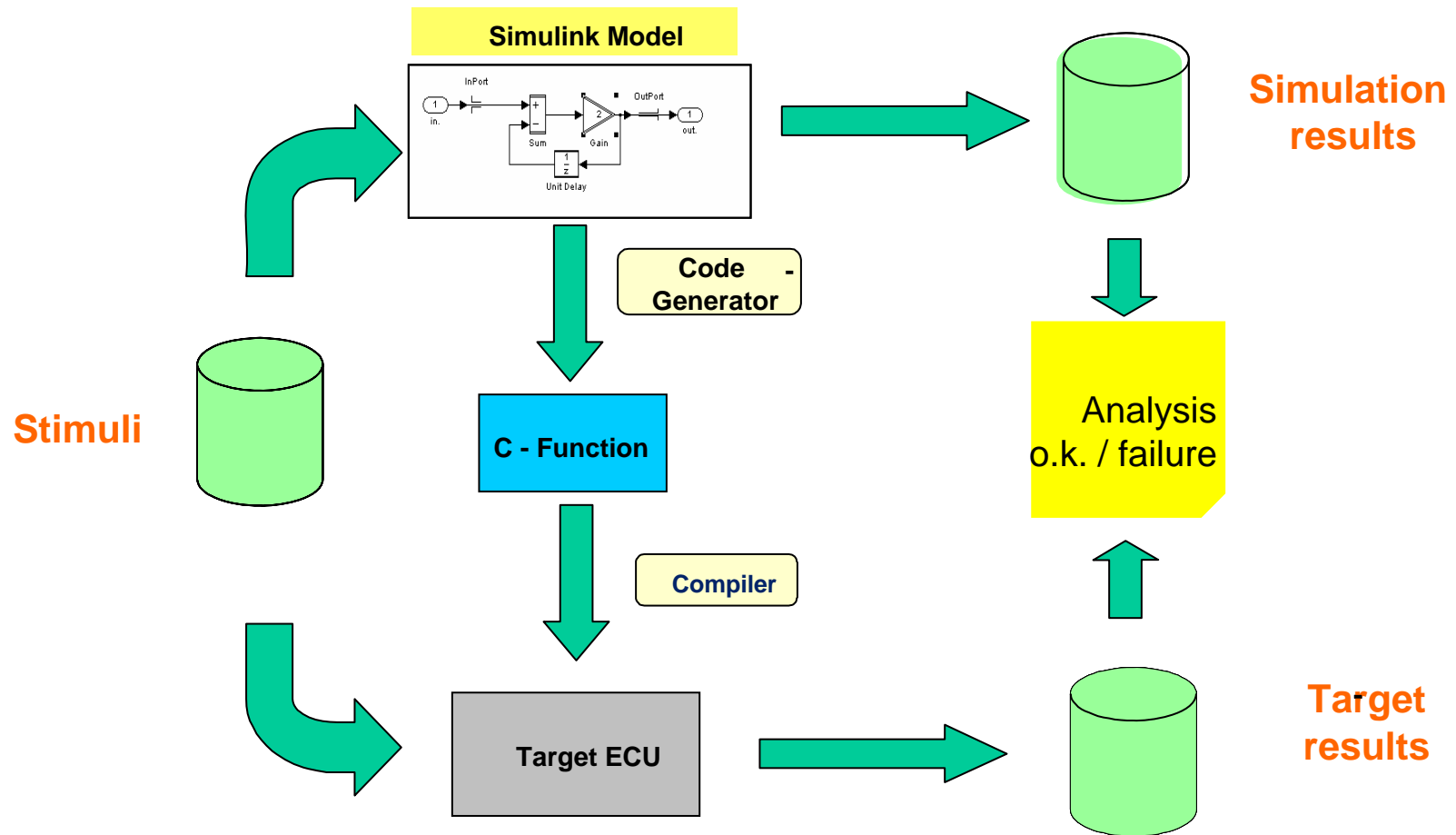


Motivation for the AVS Project

- The advance of high level software design tools into the world of safety critical applications
- The prevailing use of graphical modeling and simulation, as in MATLAB[®] and Simulink[®], among control engineers
- The facilitating of shorter production cycles through reduced complexity and approved reuse on validated design tools
- The maintenance and continuous improvement of product quality and robustness
- The tremendous risks involved in cases of faulty software in a mass produced vehicle
- The existing quality means (like e.g., FMEA) are not always applicable for software and its certification



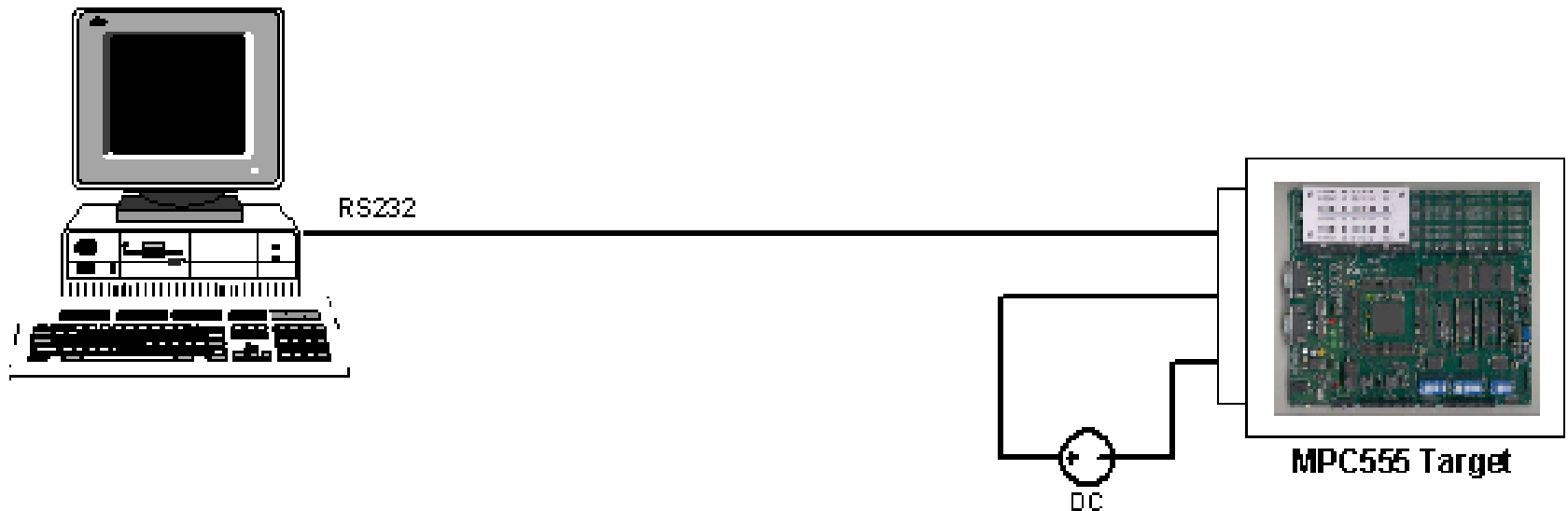
AVS (Automotive Code Validation Suite) Concept



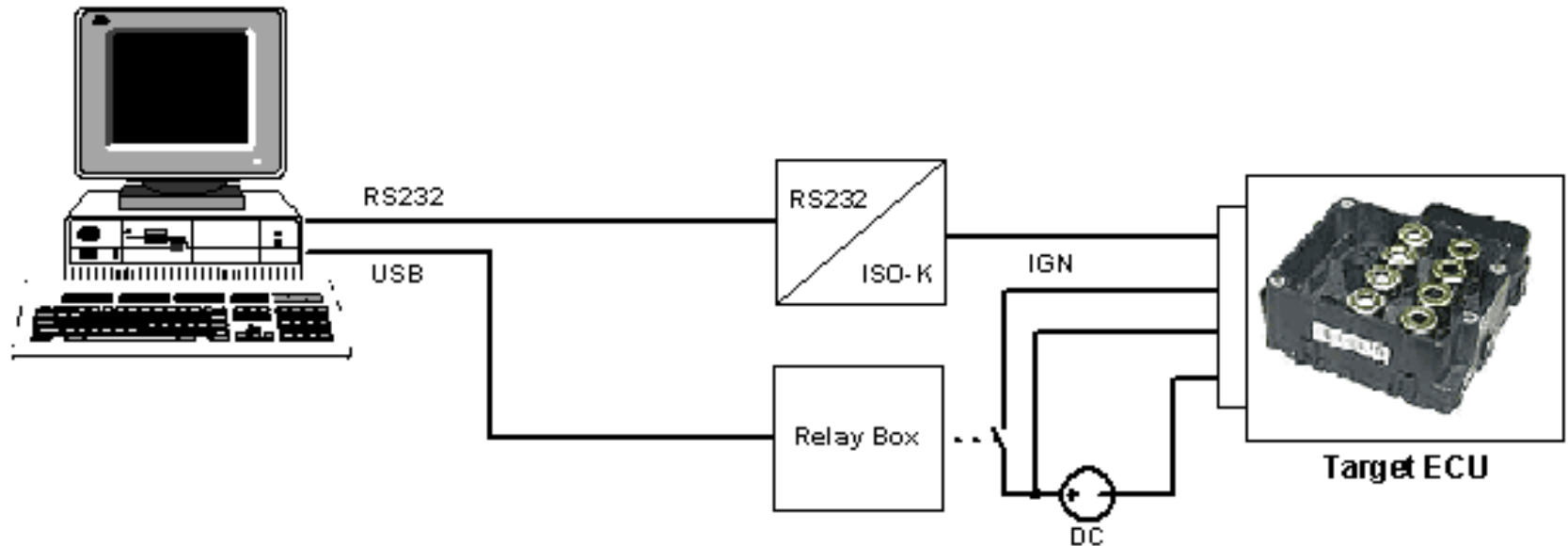
Key Properties of AVS

- Validation by means of an **automated** test-suite
- Test cases are **generic** and **derived** from the actual control algorithms
- Suite contains different test cases
 - **positive** : result o.k. expected
 - **negative** : expected to provoke error message
 - **“fault”** : documents a faulty behaviour
- Easy changeability to fit different scenarios (different code generators, compilers and optimization settings).
- Provides easy way to use private test cases to protect intellectual property (IP)
- Communication to tool vendors and automotive industry to avoid double work

Processor in the Loop (**PIL**) with Evaluation Board



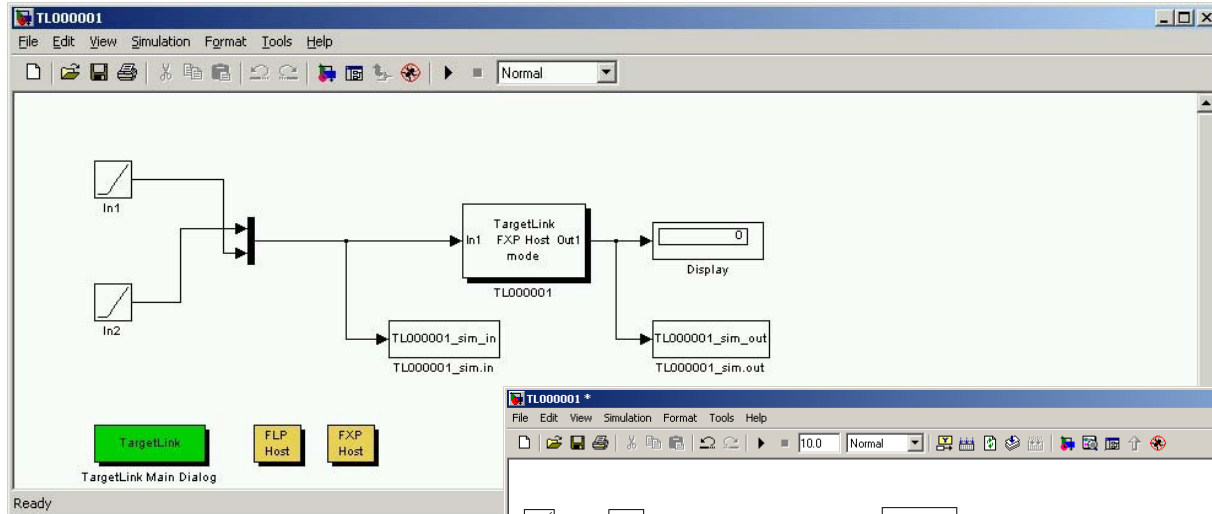
Target in the Loop (TIL) Setup with Production ECU



AVS Target in the Loop (TIL) Setup

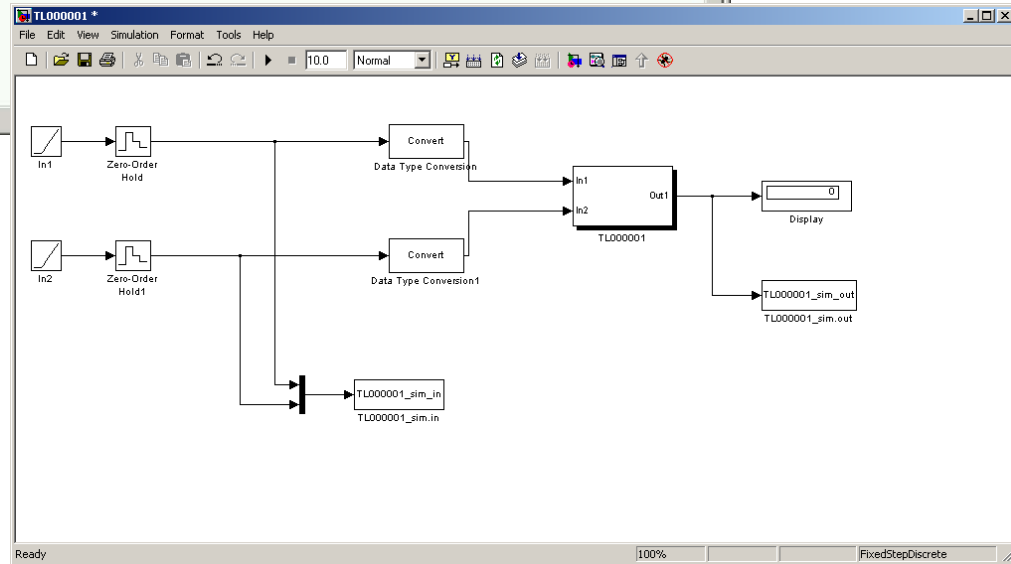


Architecture of AVS compatible Test Cases



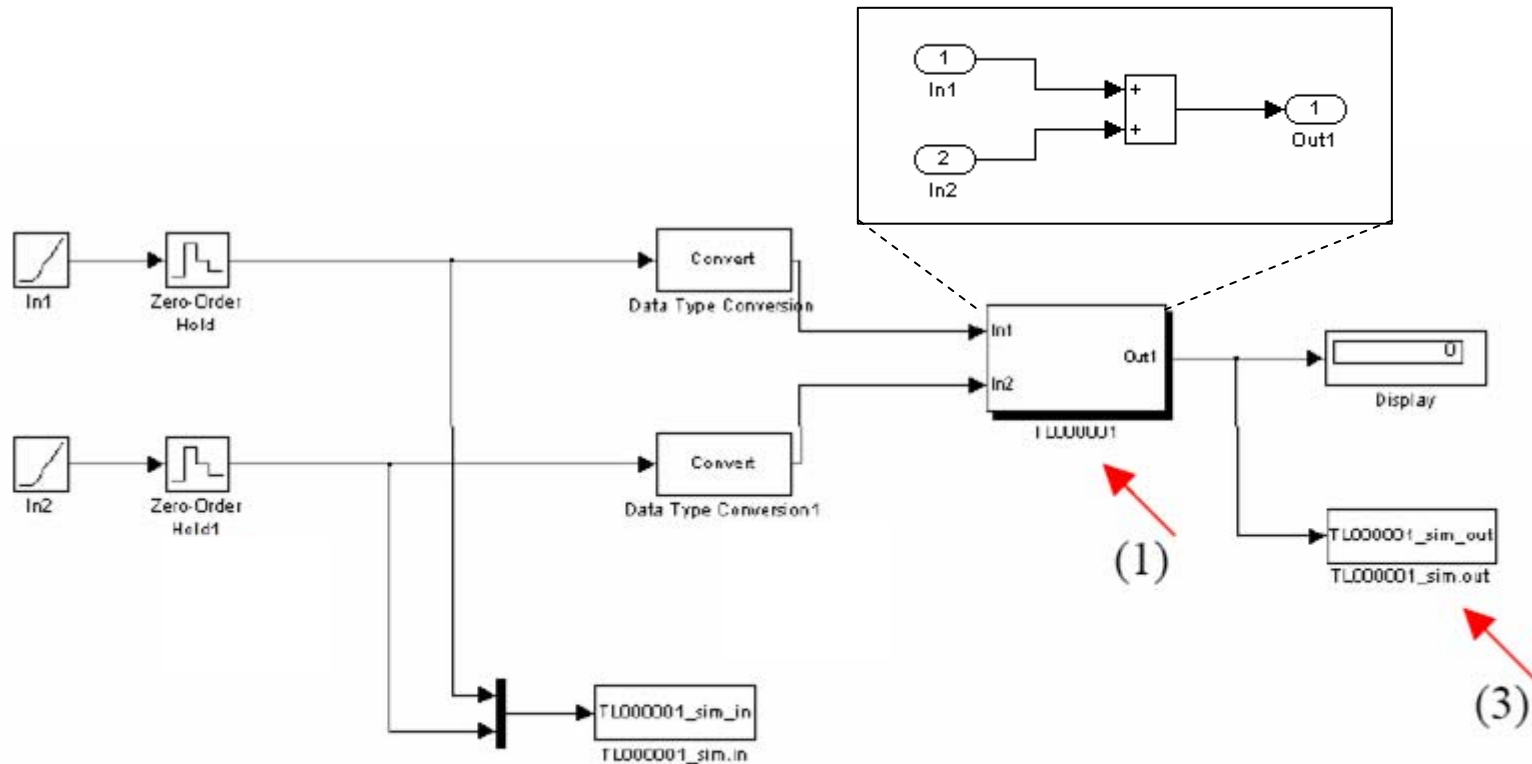
- TargetLink Test Logik

- Real-Time Workshop® Embedded Coder Test Logik



Embedded Coder Test Case

Description of Test Logic



- (1) Subsystem under test
- (2) Stimuli
- (3) Results

AVS User Interface

TÜV Rheinland Group - Autocode Validation Suite (AVS) v3.0 (2005-10-21)

Suite Help

Run Load Settings Save Settings Refresh Analyse

Testcases	Description	Information	Model Doc.
AVS			
Basic Calculate Functions			
Basic Logic Functions			
Discrete Calculate Functio...			
Expanded Calculate Funct...			
External Stimuli			
Floating Point Scaling			
TL000012	Discrete Transfer Function with a Step as input		
TL000013	Discrete Transfer Function with Sine Wave input		
TL000014	PI Controller with a Step as input		
TL000028	Testcase with several basic as well as discrete calcula...		
TL000028a	Testcase with several basic as well as discrete calcula...		
TL100003	Continental Teves Testcase "Filters" with one Band li...		
TL100014	Continental Teves Testcase "Speed" with one sine w...		
TL100014a	Continental Teves Testcase "Speed" with one sine w...		
Look Up Table Functions			
Negative Testcase			
TL000000	Provides a faulty test case		
Optimization Methods			
TL000004a	Simple If construction in Stateflow and calculate functions ...		
TL000011	Simple usage of Gains with observable variables		
TL000017d	Two Look Up Tables combined with multiplication block a...		
TL000027a	Testcase with several basic calculations and Stateflow in ...		
TL000037	Typical MinMax Block usage		
Stateflow			
TargetLink Bugs			

Choose Available Target: TMS 470_2003.bat Edit

Choose available Config File: Conti_Specific.m

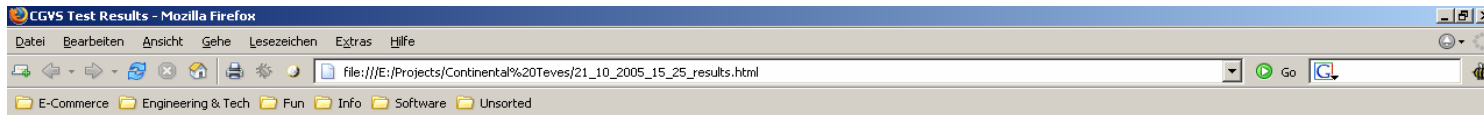
TargetLink Version
 TargetLink v1.3
 TargetLink v2.0

RTW Embedded Coder Version
 Embedded Coder v4.2 (R14 SP2)

OO :Floating Point Test Case

Multiple Categories

AVS Result Evaluation



AVS Test Record 21.10.2005

Current Status	Testcase	Current compare of Simulation results ⇄ Target results	Previous stored reference compare status	Simulation LOG	Testcase Information
	TL000000	Wrong number of simulation and target result values.	No reference file	No Error Logfile	TL000000
	TL000017d	passed	No reference file	No Error Logfile	TL000017d
	TL000018	passed	No reference file	No Error Logfile	TL000018
	TL000019	passed	No reference file	No Error Logfile	TL000019
	TL000032	failed	No reference file	TL000032	TL000032
	TL000032a	passed	No reference file	No Error Logfile	TL000032a
	TL000034	passed	No reference file	No Error Logfile	TL000034

21.10.2005 15:25

Legend:

Previous stored reference compare status:
Shows the status of a previous stored comparison results.

Simulation Log:
Contains a list of different results which were found during comparison. A link to the list is only available if different simulation and target results were determined.

- In Oct 2005, a code generation tool chain including Real-Time Workshop Embedded Coder and the TI TMS 470 C compiler successfully passed AVS v3.0

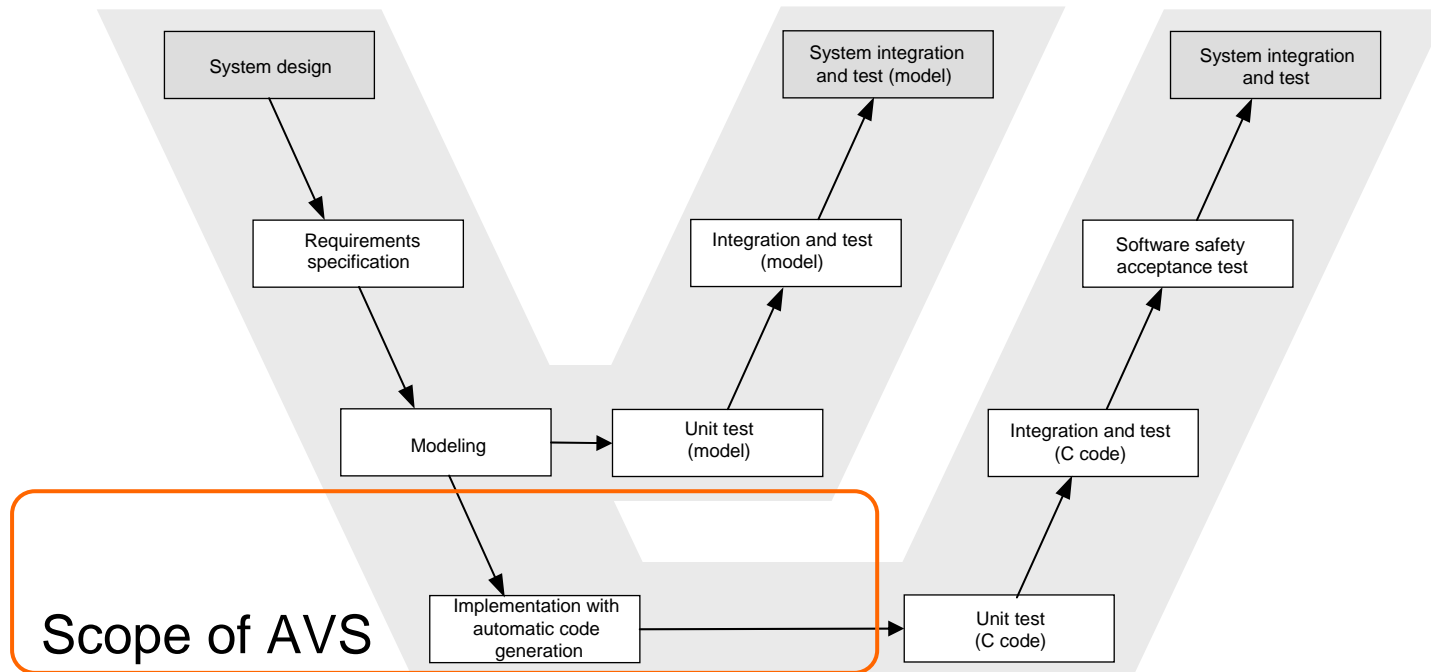
• Successful validations with varying tool chains and versions

Feel the difference



AVS for safety-related Applications, Software Safety

- Validation by means of an AVS:
additional quality assurance for organizations using production code
generation for safety-related applications



- Supports requirements according to IEC 61508 and upcoming ISO 26262

Optimization Level Verification (OLV)

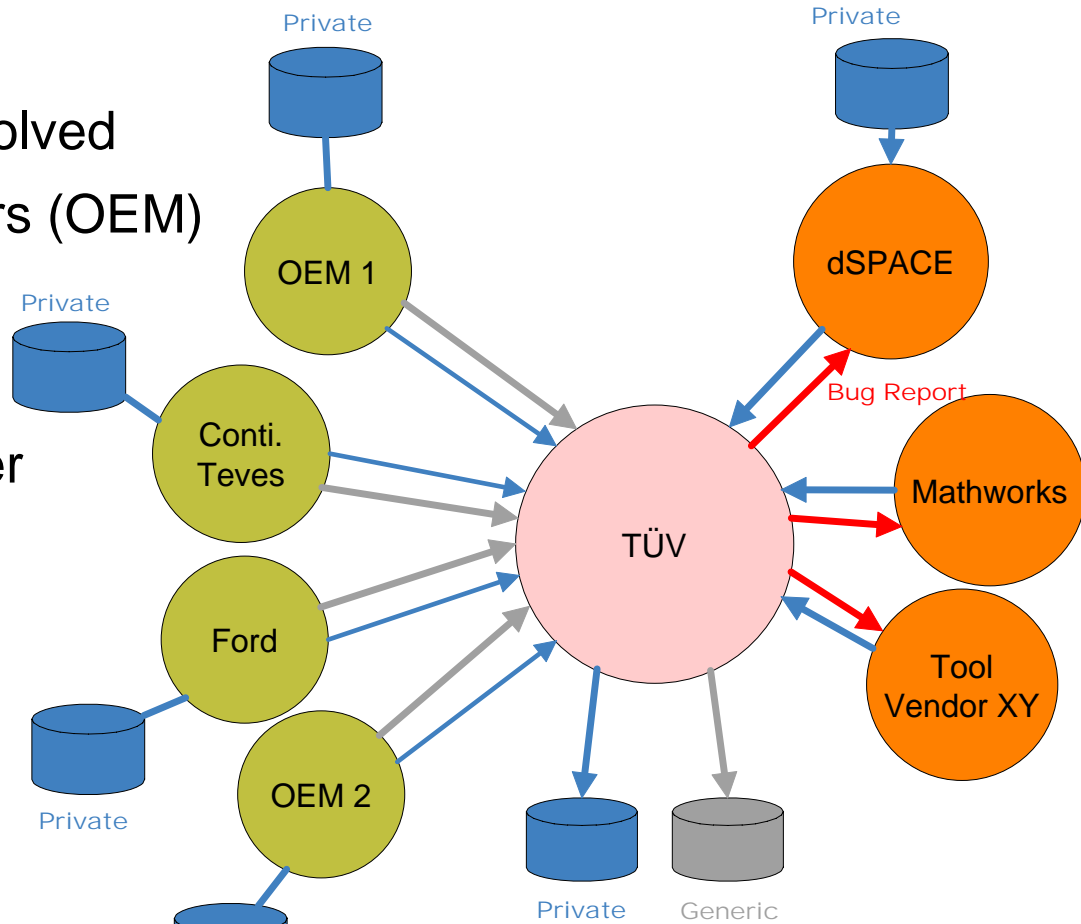
- Almost all tools (code generators, intermediate tools, compiler, assemblers, linkers) allow optimization settings
- Typical range for optimization levels is between 0 (no optimization) up to some n (e.g., 6, full optimization)
- Every optimization step means additional operations between source model and final binary code
- Every optimization can possibly change the behavior of the final ECU with regard to overall performance and timing constraints
- **OLV** property of AVS either proves “optimization works”, or gives the maximum level, where optimization has no negative impact on the final ECU



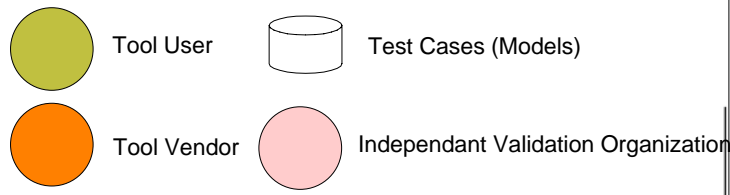
Possible Interaction of AVS Partners

Different parties involved

- Car manufacturers (OEM)
- Tier 1 supplier
- TÜV
- Tool manufacturer
- Service provider



Legend



Feel the difference



Affected Areas : Where AVS can be used

- CASE tools & methods
- Safety critical devices
- Controls
- Functional architecture
- Electric/Electronic (EE) architecture
- Vehicle integration
- Testing and regression testing



Automotive Code Validation Suite

Summary

- AVS validates the entire tool chain, including the microcontroller and target ECU
- Processor in the Loop (**PIL**) Setup, Target in the Loop (**TIL**) Setup
- Joint venture Ford / ContiAutomotive / TUV Rheinland
- Active participation in German **AVS working group** (Audi, Ford, Daimler AG, BMW, Volkswagen, ContiAutomotive, Bosch, Wabco, Siemens-VDO, Getrag-Ford, ZF Friedrichshafen), comparable to AUTOSAR aim :

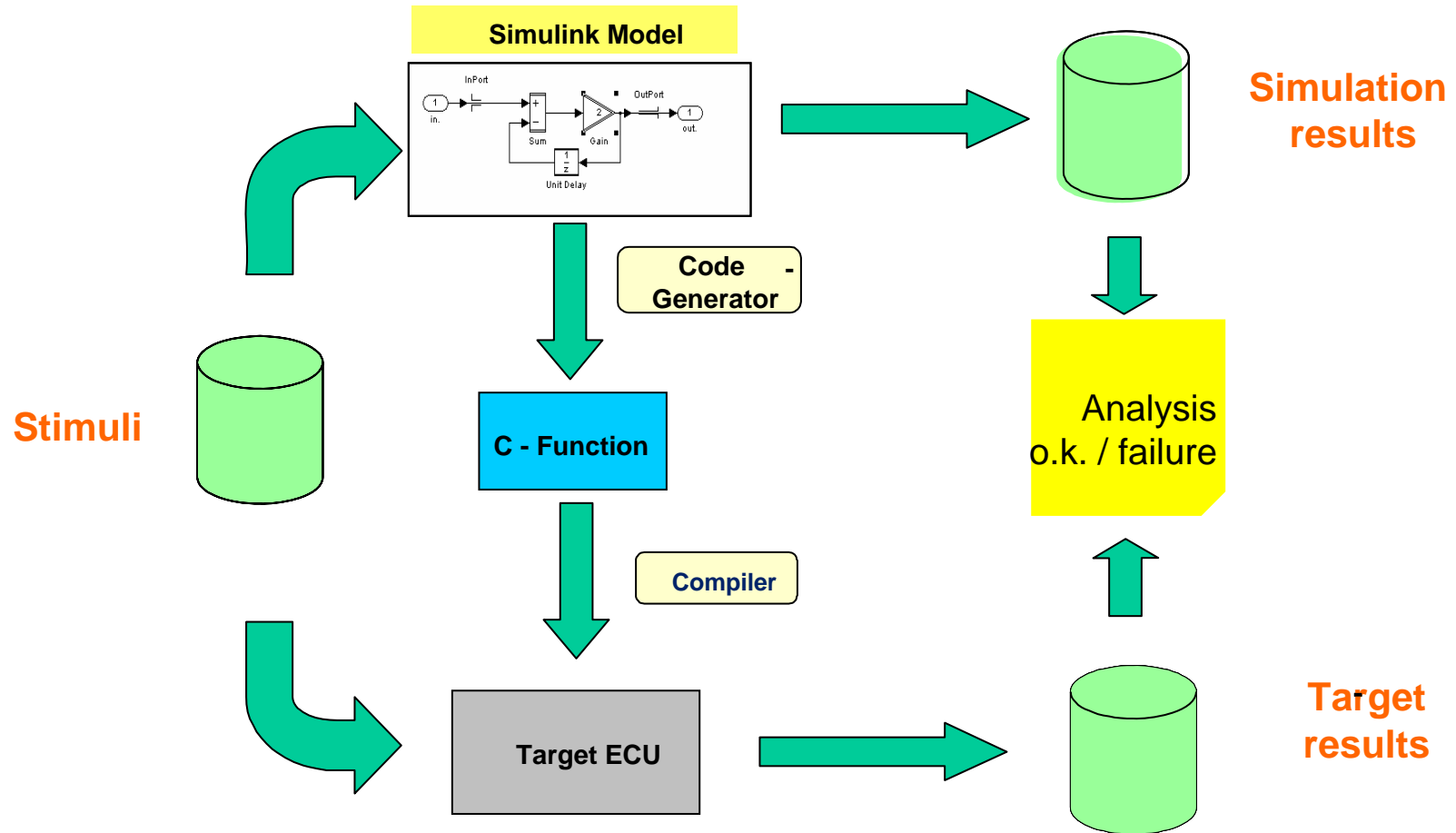
"Cooperate on standards, compete on implementation"

- Open for **new partners**
- 3 major revisions showed **high quality** of investigated tool chains
- Optimization Level Validation (**OLV**) saves target memory and execution time
- New features and areas of improvement for AVS are defined

Feel the difference



AVS (Automotive Code Validation Suite) Concept



REFERENCES

- "Development Guidelines for Vehicle-Based Software." MISRA, 1994 (<http://www.misra.org.uk/>).
- MISRA-C:2004: "Guidelines for use of the C language in critical systems." MISRA, 2004 (<http://www.misra.org.uk/>).
- IEC 61131-3:2003-01: "Programmable controllers Part 3: Programming languages." Genf/Schweiz: Bureau Central de la Commission Electrotechnique Internationale (<http://www.iec.ch/>), 2003
- IEC 61508: "Functional safety of electrical/electronic/programmable electronic safety-related systems." (<http://www.iec.ch/>), 1998
- ISO 26262:2006: "Road vehicles - Functional safety." Working draft, 2006
- ECE Regulation No. 13: "Uniform Provisions concerning the approval of vehicles of categories M, N and O with regard to braking." United Nations Economic Commission for Europe, 2003 (<http://www.unece.org/>)
- "Validation of the MathWorks code generator Real-Time Workshop® Embedded Coder with the Autocode Validation Suite (AVS) v3.0." Report No. 968/EL 211.02/05, TÜV Rheinland Group, 2005
- Information concerning the V-model : <http://www.v-modell.iabg.de/>
- AUTOSAR (AUTomotive Open System ARchitecture) <http://www.autosar.org/>

Feel the difference



CONTACTS

Ekkehard Pofahl, Ford Motor Company

epofahl@ford.com

Torsten Sauer, Continental Automotive Systems

Torsten.Sauer@contiautomotive.com

Oliver Busa, TÜV Rheinland Group

Oliver.Busa@de.tuv.com

Feel the difference

