

30 June 2020

AUTOSAR Architecture

Modeling of Multi-core

Electric Powertrain Controller

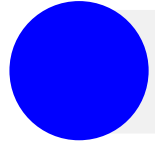
Dr Sakthivel Manikandan Sundharam /
Software Architect

Delphi
Technologies

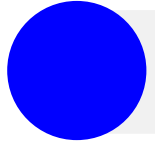


Bio : Sakthivel Manikandan Sundharam

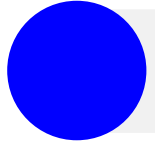
True! Bit longer name - Shortly "Sakthi"



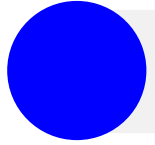
Software Architect – Powertrain Electrification & Electronics



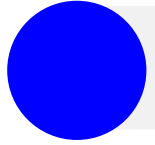
15+ Years of Automotive Embedded System Experience



Ph.D. in Timing Aware Model-Based Design to Automotive Embedded Systems, University of Luxembourg, Luxembourg




Masters in Embedded Systems, College of Engineering Chennai, India



Work revolves around software architectural topics incorporating timing, memory, and safety constraints of automotive software.

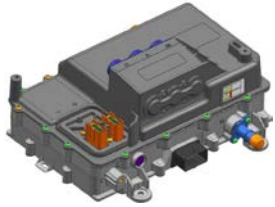


Outline / Agenda

- 
- 1 Delphi Technologies - Powertrain Electrification Product Portfolio
 - 2 HV Inverter System Context
 - 3 Pitfalls in Legacy Approach of SW Architecture Modeling
 - 4 Evaluation of Journey
 - Requirements to Architecture
 - Architecture authoring
 - Interfaces / Data dictionary
 - 5 Lessons learnt and Best practices

Delphi Technologies - Powertrain Electrification Product Portfolio

Low cost, high density, rugged with various levels of integration available



Single Inverter



Inverter w/ DC/DC (CIDD)



48V Inverter for BSG



48V DC/DC Converter



High Voltage Box



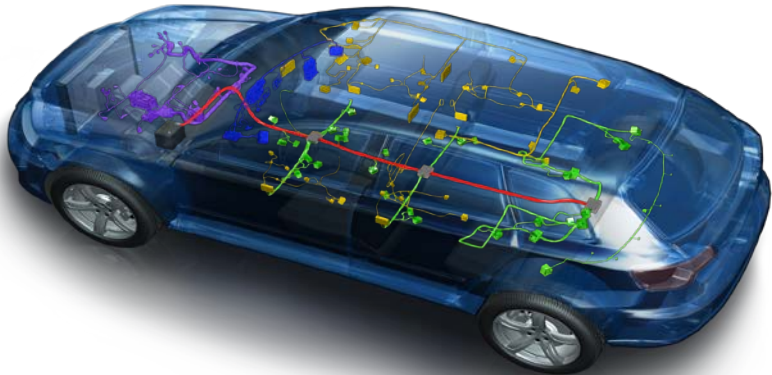
3-in-1 Inverter



DC/DC Converters



Dual Inverter



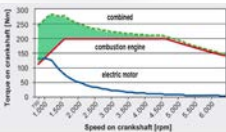
Battery Pack Controller



Dual Inverter/ Converter/ Hybrid Controller



Supervisory Controller (Hybrid Control Unit)



Hybrid Control Software



On-board Battery Chargers

Delphi Technologies Inverter – The Next Generation

Inverter with conventional Power Module



Conventional
Many, many wire-bonds

Gravimetric power density
(kVA/Kg)

16.1 20.6
25% higher

Volume (L)

10.7 7.6
30% smaller

Mass (Kg)

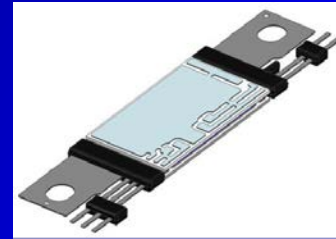
14.5 8.4
40% lighter

Efficiency Improvement (MPG)

0 2



Delphi Technologies Inverter with viper

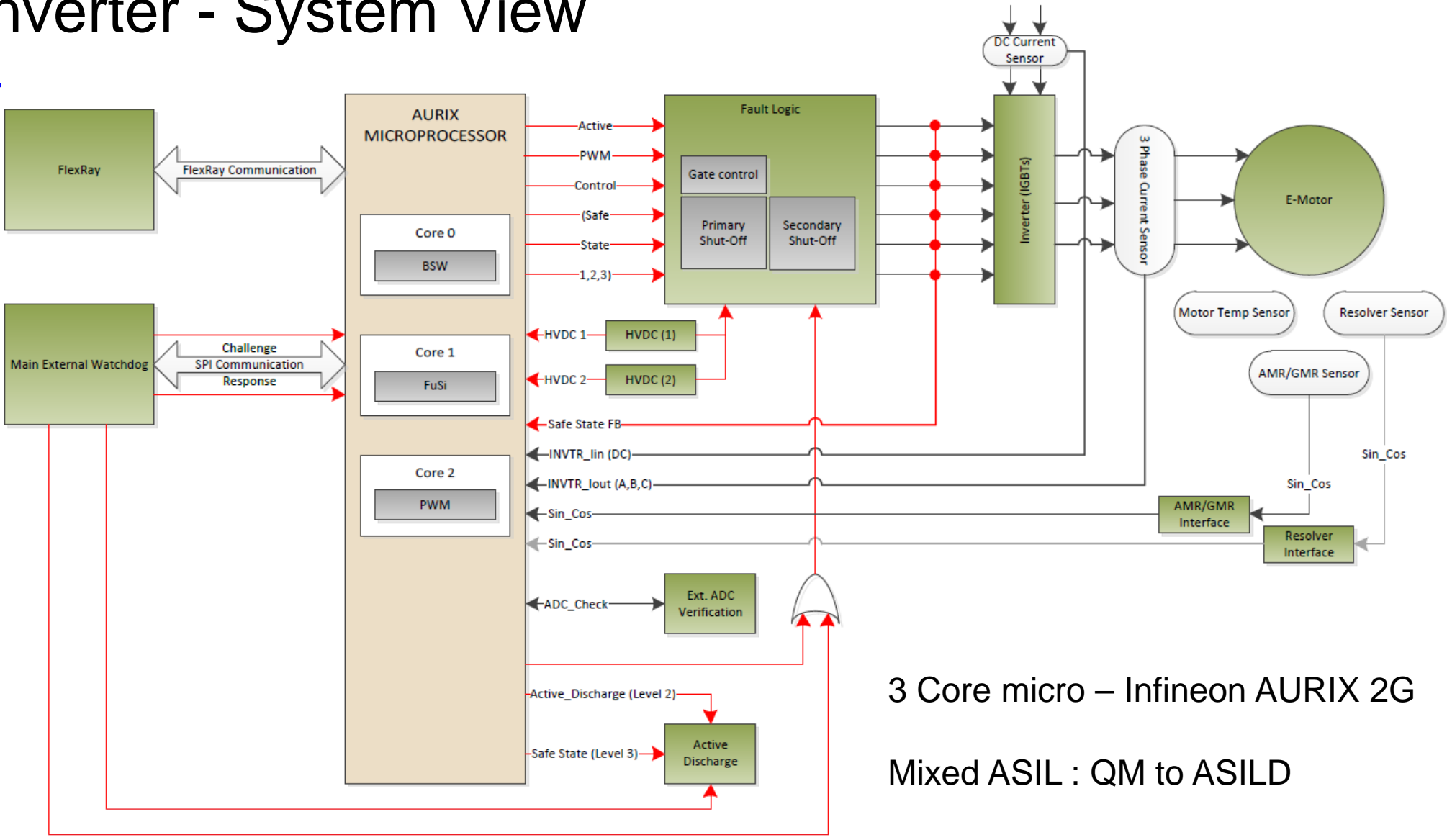


Delphi Inverter next generation

- Next gen Viper enables extra high voltage **800V** bus inverters
- Flexibility to move from **Si** to **SiC** power switch to enable higher efficiency & lower cost
- **Advanced capacitor** enables up to 70% reduction in component volume & weight



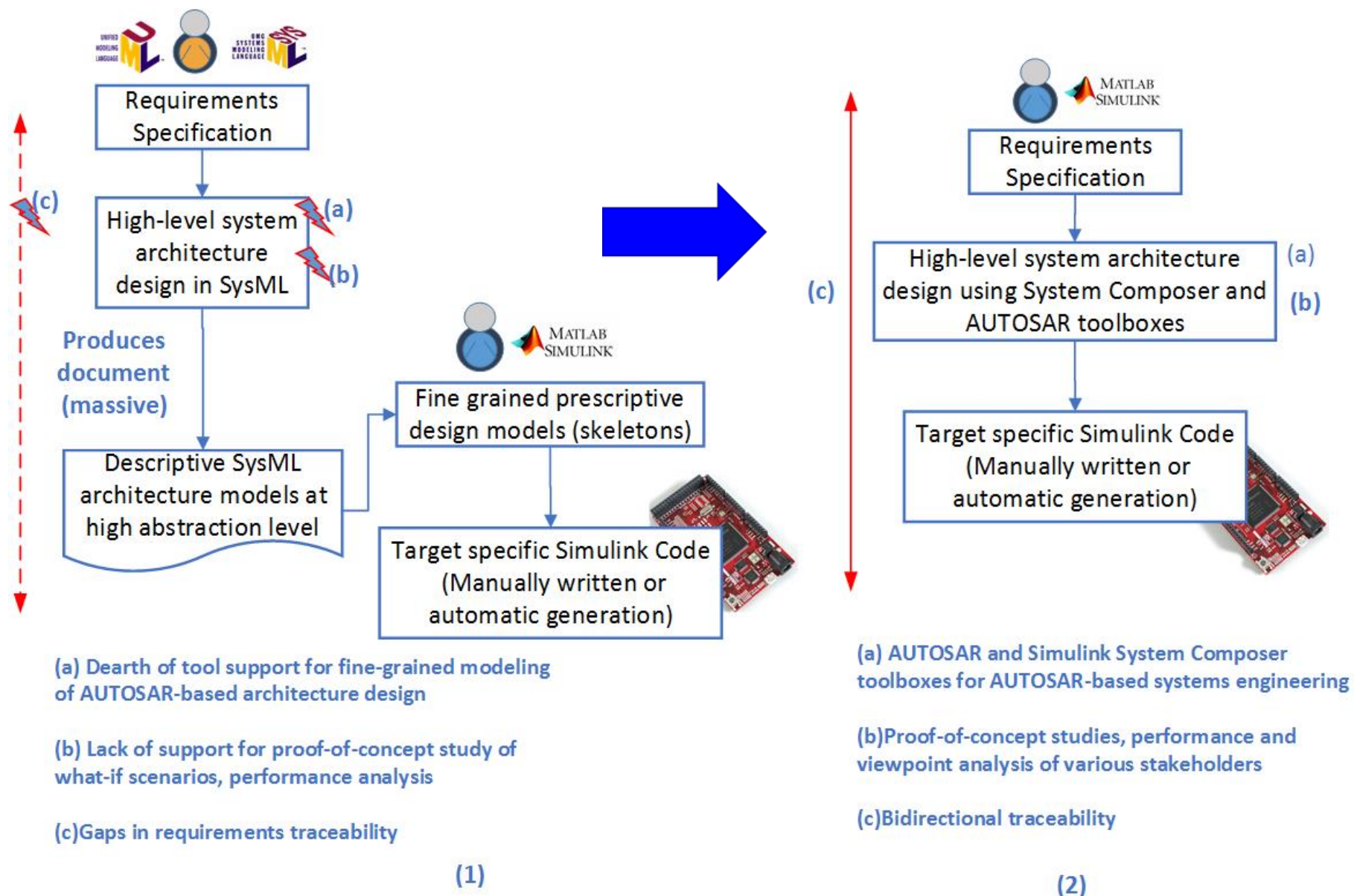
HV Inverter - System View



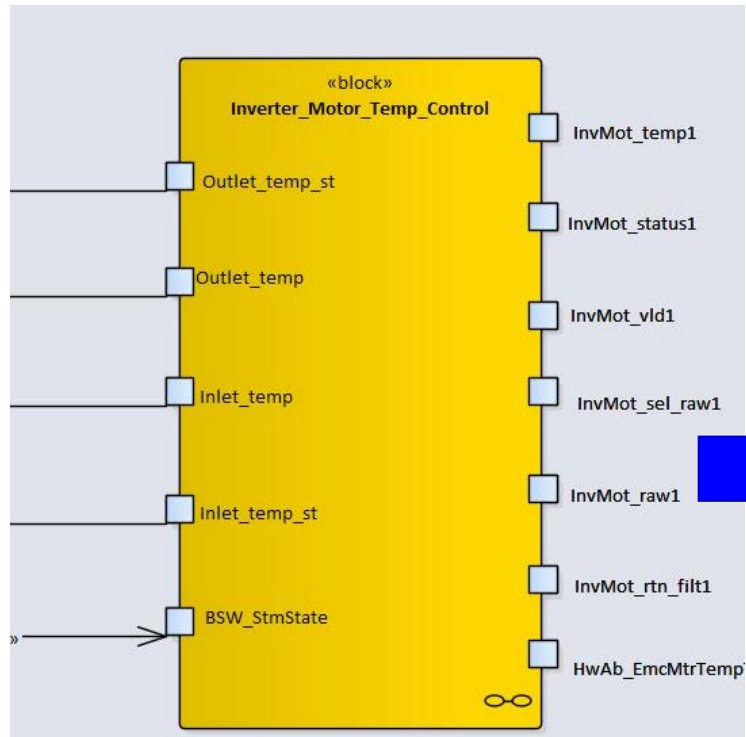
3 Core micro – Infineon AURIX 2G
 Mixed ASIL : QM to ASILD

Multicore Electric Powertrain Controller

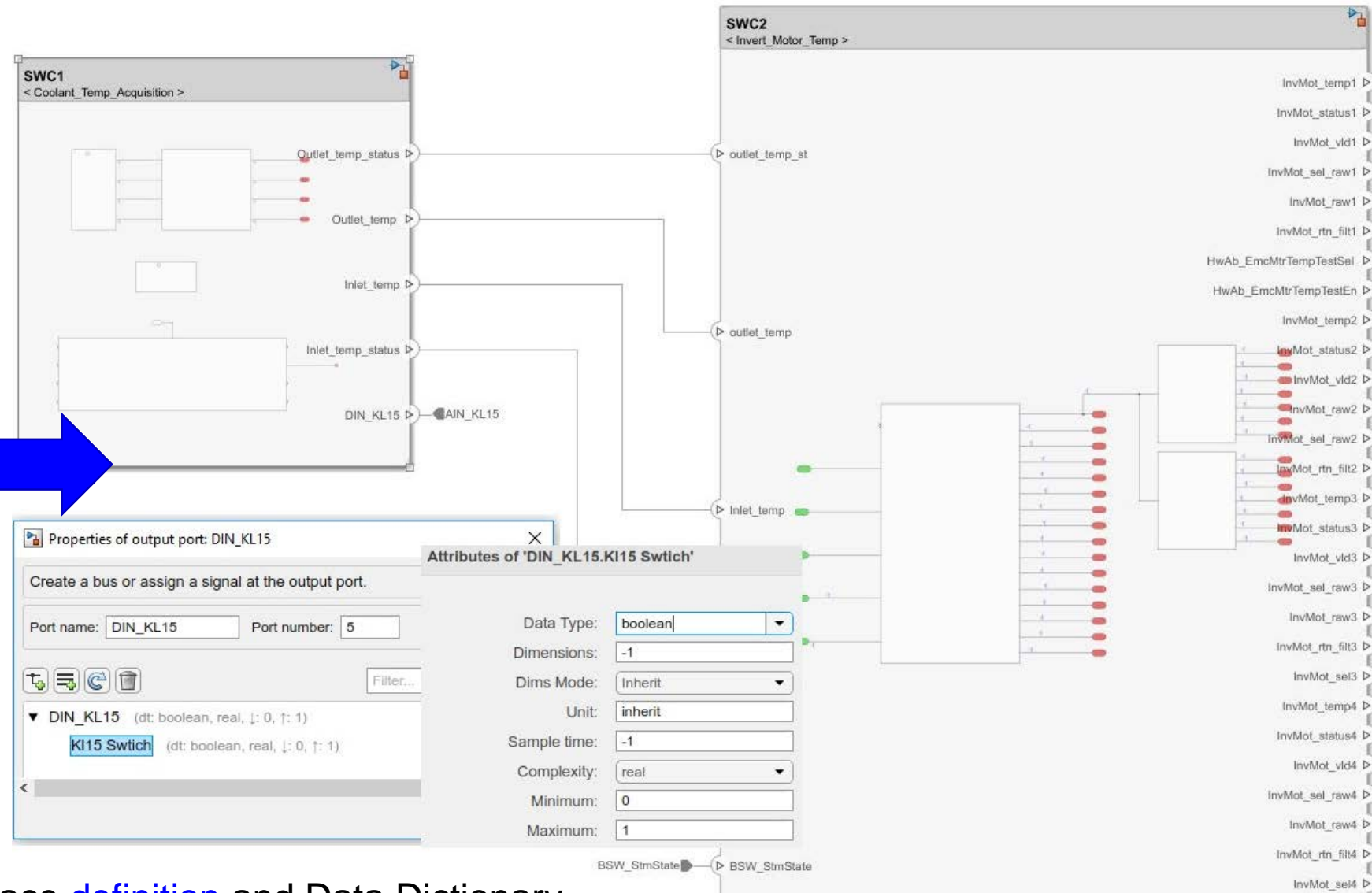
Pitfalls in Legacy Approach and Best Practices Evaluated



Static Software Architecture Tooling Twins MLSL's AUTOSAR Blockset + System Composer



Legacy SysML



Interface definition and Data Dictionary

Publishing Architecture Modeling onto Requirements Database

The screenshot shows the software interface with a context menu open over a diagram element. The menu options are:

- Publish Block/(Sub)System
 - Default
 - Requirements in E-Library
 - Testcases in Drive Pilot
 - UserStory in <create yourself project>
- Link Block/(Sub)System with Existing Item
- Push/refresh Polarion attributes to Simulink
- Open Linked WorkItem (2)...
- Refresh Published Diagram(s)

Below the menu, a 'Hyperlinks' table is visible:

Role	URL
external reference	http://localhost:31415/matlab/feval/r...

The background shows a diagram with components SWC1 (< Coolant_Temp_Acquisition >) and SWC2 (< Invert_Motor_Temp >). A 'Description' box is also present in the interface.

approach provides a **lean** way to **publish** the design to requirements database.

Also for existing requirements, it provides an option to **link** them


Requirements to Architecture Linking

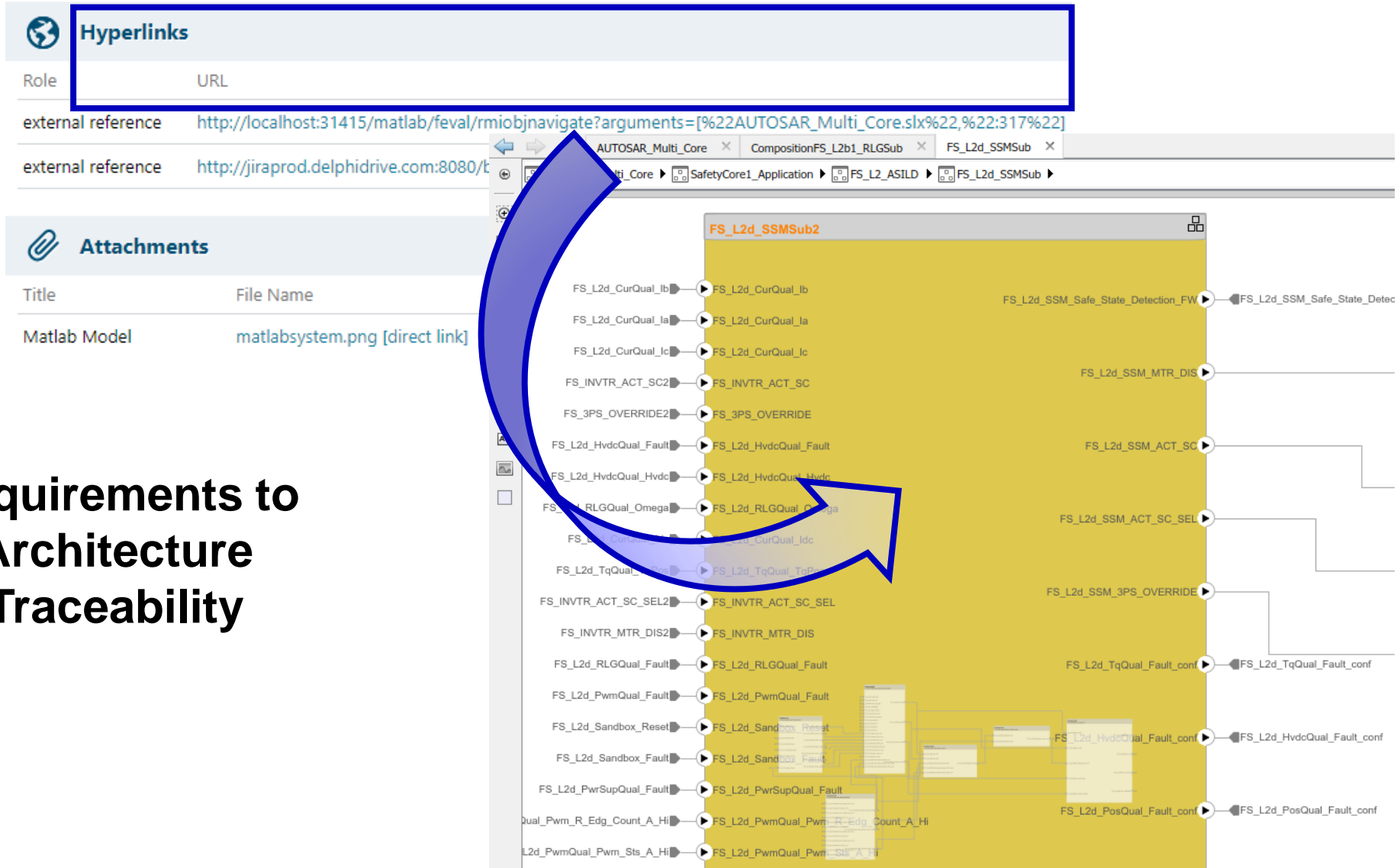
The screenshot displays a software interface for system architecture. The main area shows a diagram of a system with components like SWC1 and SWC2, and various ports and signals. A red box highlights a list of URLs in the top right corner, which are: <https://reqdemo.polarion.com/polarion/>, <https://reqdemo.polarion.com/polarion/>, and <https://reqdemo.polarion.com/polarion/>. Below the diagram, a table titled 'Requirement links - Temperature_Control' is visible, showing a list of requirements and their links to the architecture.

Label	Source	Type	Destination
Temperature_Control.slmx	Changed source: 0/9		Changed destination: 0/9
Polarion: IC-509	Temperature_Control	Implements	https://reqdemo.polarion.com/polarion/
Polarion: IC-510	SWC1	Implements	https://reqdemo.polarion.com/polarion/
Polarion: IC-512	Temperature_Control	Implements	https://reqdemo.polarion.com/polarion/
Polarion: IC-513	Out Bus Element1	Implements	https://reqdemo.polarion.com/polarion/
Polarion: IC-514	SWC3	Implements	https://reqdemo.polarion.com/polarion/
Polarion: IC-515	SWC1	Implements	https://reqdemo.polarion.com/polarion/
Polarion: IC-516	SWC2	Implements	https://reqdemo.polarion.com/polarion/
Polarion: IC-517	SWC1	Implements	https://reqdemo.polarion.com/polarion/

Tracking of requirements back and forth between modeling and requirements database to verify fulfillment of requirements

Bi-directional Traceability - Forward

is allocated to  AINV-47779 - Reqs - SW Architecture



The screenshot displays a software architecture tool interface. On the left, there is a 'Hyperlinks' table with two entries:

Role	URL
external reference	http://localhost:31415/matlab/feval/rmiobjnavigate?arguments=[%22AUTOSAR_Multi_Core.slx%22,%22:317%22]
external reference	http://jiraprod.delphidrive.com:8080/t

Below the table is an 'Attachments' section with one entry:

Title	File Name
Matlab Model	matlabsystem.png [direct link]

The main part of the screenshot shows a detailed diagram of a component named 'FS_L2d_SSMSub2'. The diagram consists of numerous nodes and connections. A large blue arrow points from the 'Hyperlinks' table towards the diagram, indicating a forward traceability path. The diagram nodes include various fault and state detection identifiers such as 'FS_L2d_CurQual_Ib', 'FS_L2d_SSM_Safe_State_Detection_FW', 'FS_L2d_SSM_MTR_DIS', and 'FS_L2d_SSM_ACT_SC'. The diagram is overlaid on a yellow background.

**Requirements to
Architecture
Traceability**

Bi-directional Traceability - Backward

The screenshot illustrates the process of backward traceability in a software development environment. On the left, a hierarchical tree view shows the project structure, with the 'FS_L2d_SSMSub2' component selected. A context menu is open over this component, with the 'Polarion' option selected, leading to a sub-menu where 'Open Linked WorkItem (1)...' is chosen. This action opens a 'Linked Work Items' window in the top right, which displays a table with the following data:

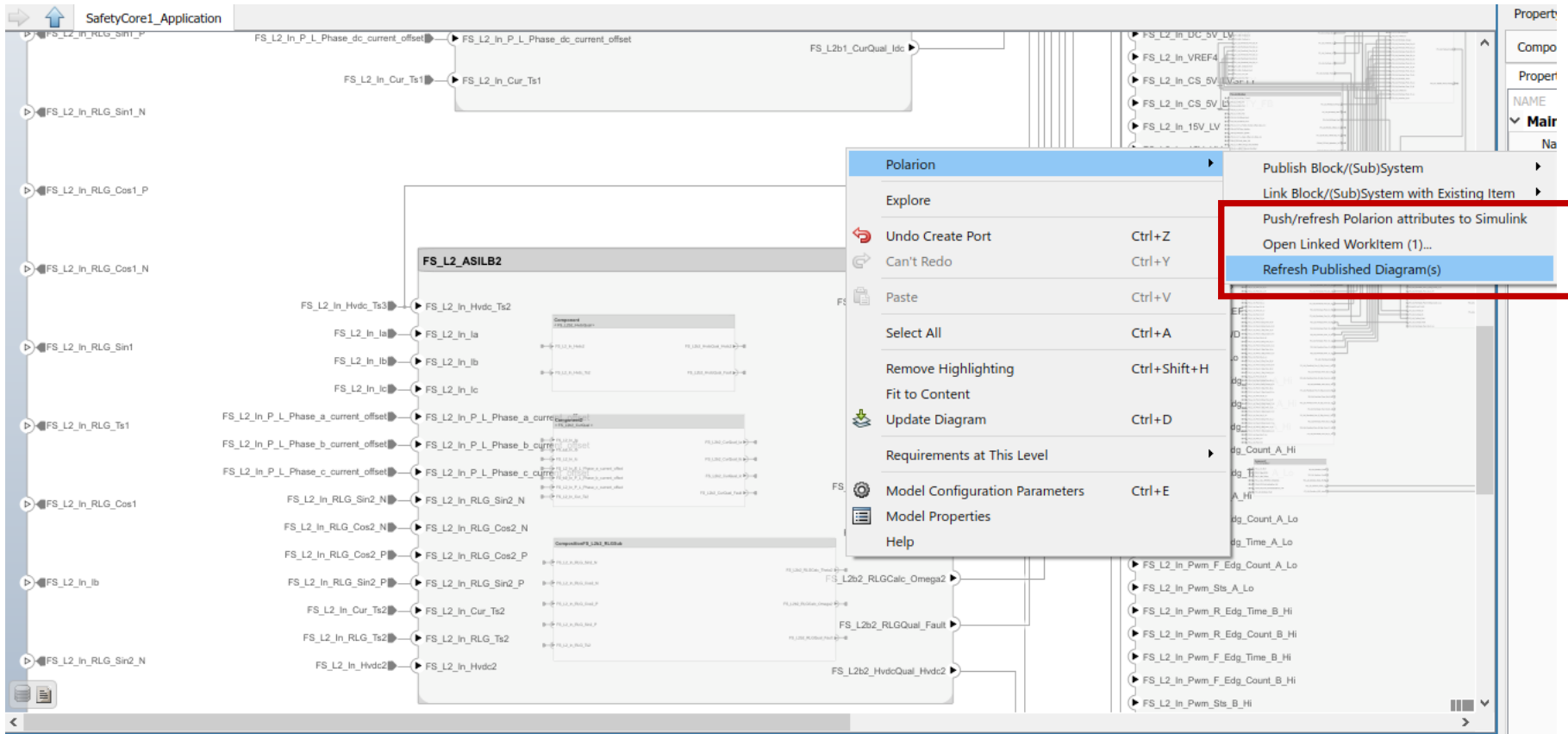
ID	Title
AINV-55960	Planning - SW Architecture - Safe State Manager

Below the table, a detailed view of the selected work item is shown. It includes the title 'AINV-55960 - Planning - SW Architecture - Safe State Manager', a status of 'In Progress', and a description area. A large blue curved arrow points from the 'Open Linked WorkItem (1)...' menu option to the work item details, indicating the flow of information from the architecture back to the requirements.

At the bottom of the screenshot, a component diagram for 'FS_L2d_SSMSub2' is visible, showing various sub-components and their interconnections, such as 'FS_L2d_CurQual_Ib', 'FS_L2d_CurQual_Ia', 'FS_L2d_CurQual_Ic', 'FS_INVTR_ACT_SC', 'FS_L2d_TqQual_TqPos', 'FS_L2d_HvdcQual_Fault', and 'FS_L2d_CurQual_Fault'.

Architecture to Requirements Traceability

Architecture to Requirements – Seamless Approach



- Whenever model updated due to maturity of the project, refresh option updates the same model onto requirements database
- Reversely, requirement attributes changed on the requirements database can easily be pushed back to SW architecture

Requirement links - AUTOSAR_Multi_Core

Label	Source	Type	Destination
AUTOSAR_Multi_Core.slmx	Changed source: 0/3		Changed destination: 0/3
Polarion: AINV-58075	SafetyCore1_Application	Implements	http://polarionprod1.delphidrive.com/polarion/
Polarion: AINV-58341	BSWCore0_Application	Implements	http://polarionprod1.delphidrive.com/polarion/
Polarion: AINV-59475	AUTOSAR_Multi_Core	Implements	http://polarionprod1.delphidrive.com/polarion/

arxml Import from BSW Tools (f.e Vector BSW-stack Tools)

Vector DaVinci Developer - ALB - [Component Type CrashManager]

File Edit View Options Window Help

Workspace

ECU Projects

ALB

Software Design

Data Mapping

End-to-End Protection

Library

Bus_Diagnostic

BusDiag_Mgr

ComWrapper

Control_Boa

Coolant_Tem

CrashManager

Dem_AppMgr

Dem_Like

diagnosis_infra

diagnosis_infra

ElectricMachin

Export to XML

File:

AUTOSAR version:

AUTOSAR V4.3.0

Export user-defined attributes

Create data types/constants for signals if required

Don't export Communication data

Don't export Data Mapping

Don't export End-to-End Protection

Create UUID on export

Export AUTOSAR package (platform types,...)

SWArchitecture

SWArchitecture

AUTOSAR

Component

AUTOSAR Importer App

Select ARXML

Create Component

Select AUTOSAR software description to import

Arxml Files: "CrashManager.arxml" "DataTypes.arx" Browse

What to consider

Import an AUTOSAR software description ARXML file to create a Simulink model.

SWArchitecture

AUTOSAR

Component

< CrashManager_1 >

PpCrashMgr_Trigger_SafeState

Com_HV_Stable

PpPL_Crash_AD_Active

DELOutf_AB_Deaktivierung_HV

PpPL_Crash_React_st

IoHwAb_KL15_Switch

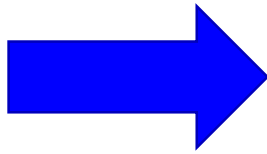
PpPL_Crash_st

MotCtrl_Control_Mode_R

P_L_Actvdcha_main_command_S

Generation of SW Architecture Documents

The image shows the MATLAB R2019b environment. The file explorer on the left displays a project structure with folders like 'arxmls', 'documentation', and 'FS_Arch'. The editor window shows MATLAB code for defining enumeration types. A small 'Autocode Helper' dialog box is open in the foreground, showing 'AUTOSAR_Multi_Core.slx' selected and a 'Perform Autocode' button.



Automated Scripts

The screenshot shows a PDF document viewer displaying the 'AUTOSAR_Multi_Core.pdf'. The table of contents includes chapters on Model Version, Root System, Subsystems, and Dictionary. A metadata table is also present.

Delphi Technologies	AINV54431
	Issue 1.0

Software Architecture Document	
Author	Sakthivel Manikandan SUNDHARAM
Date	29-Jan-2020
Model	AUTOSAR_Multi_Core

Delphi Technologies confidential
All rights reserved. No part of this publication may be reproduced in any material form (including photocopying or storing in any medium by electronic means and whether or not transiently or incidentally for some other use of this publication) without the written permission of Delphi Technologies.

Lessons Learnt and Best Practices

MLSL

AUTOSAR SW Architecture Authoring

- Modeling of AUTOSAR-based system architecture using AUTOSAR blockset together with System composer toolbox in recent releases of Matlab/Simulink.
- Creating fine-grained AUTOSAR architecture models using Simulink System Composer data dictionary support.



Requirements to SW architecture mapping

- Employing seamless approach to establish bidirectional traceability between modeling environment and the requirements database. Tracking of requirements back and forth between both the environments to verify fulfillment of requirements.
- To publish requirements and design on to requirements database. Also, the approach updates both requirements and design whenever adapted for changes due to technical discussions in a more efficient way.



Architectural simulation and SAD

- Import and export of ARXMLs between architectural modeling environment to Basic software (BSW) configuration and development tool-chain to reduce ambiguity on architectural considerations and development time.
- Early model-based performance and trade-off analysis of non-functional requirements using custom-defined profiles (e.g. employing Matlab/Simulink and System Composer toolboxes).

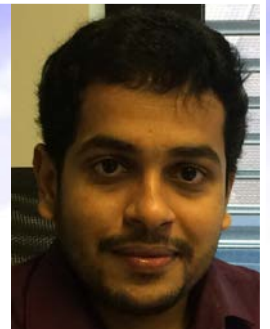


30 June 2020

AUTOSAR Architecture
Modeling of Multi-core
Electric Powertrain Controller

Dr Sakthivel Manikandan Sundharam /
Software Architect

Delphi
Technologies



Q & A

