MathWorks
AUTOMOTIVE CONFERENCE 2023
Europe

VDA SIL Standard – How it changes the SW and system development in the automotive industry

Amir Sardari, Robert Bosch GmbH

Gernot Schraberger, MathWorks

BOSCH

MathWorks

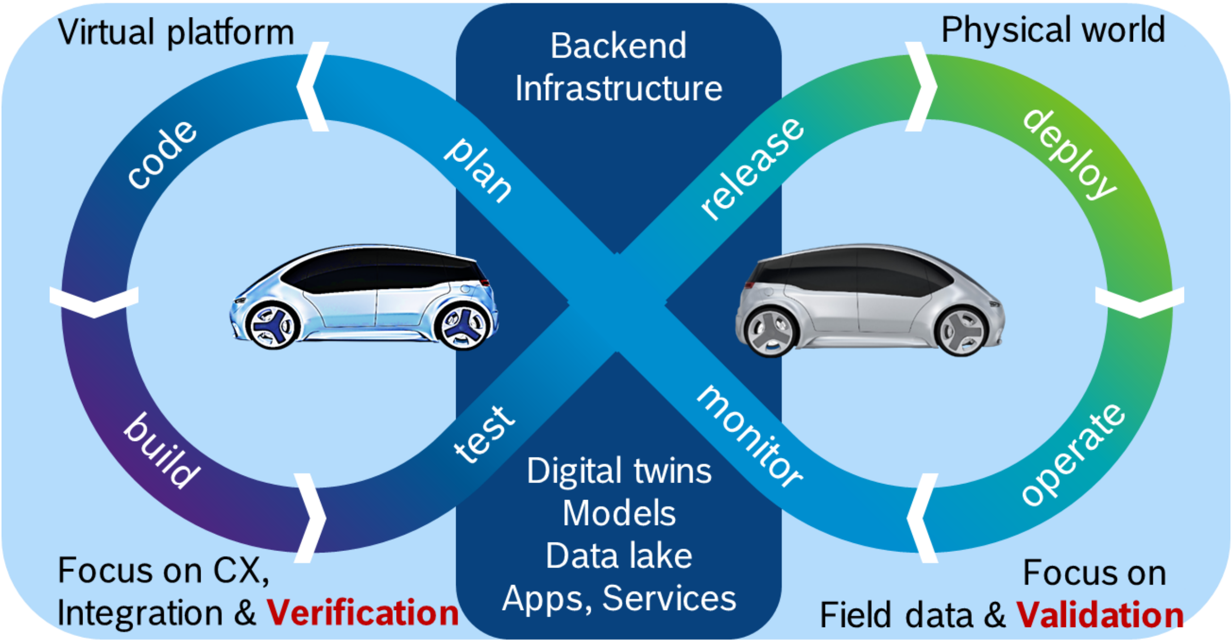
VDA Verband der Automobilindustrie

Agenda

- Why Software-in-the-Loop?
- Why Standards for **SiL**?
- VDA Automotive **SiL** Architecture
- VDA Automotive **SiL** Management Process
- Implementation and Proof-of-Concepts
- Proof-of-Concept based on Simulink with FMI3.0
- Key Takeaways

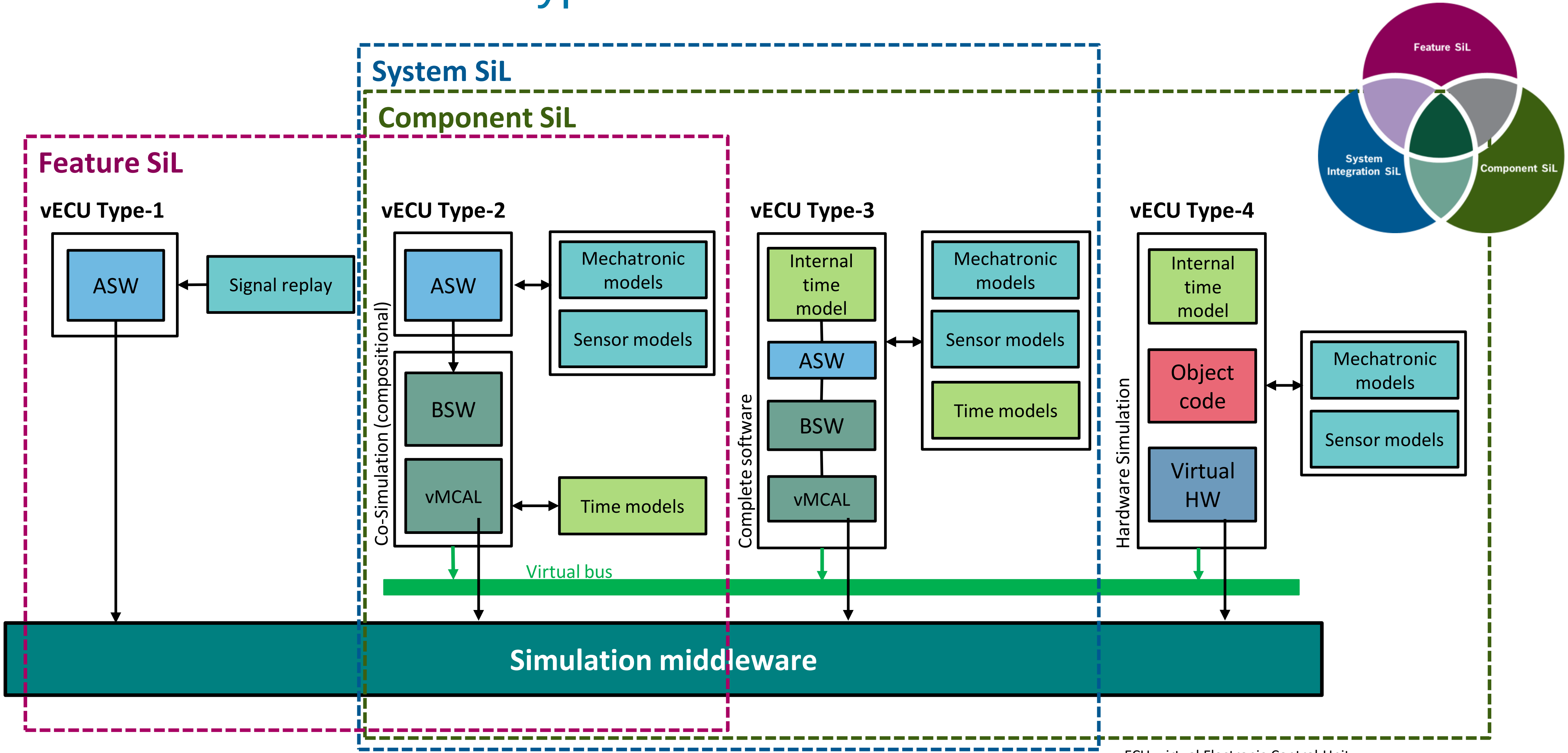
Why Software in the Loop?

- Functional **complexity** of the automotive industry is constantly increasing
- Majority of the **market growth is driven by SW-intensive systems**
- Development **cycle times reduction**
- **Frequent SW updates necessity** (e.g. security, feature upgrades, fast changing environment) required



- ➢ Verification only possible via fast and high scalable virtualization / **SiL** and digital twins, models and data
- ➢ Intensive field-based validation cycles (operation) required to validate the assumptions, the system robustness and to collect field data
- ➢ Verification & validation highly synchronized and merged to one cycle via data & digital twins in the backend

SiL Classification and Types of vECU



vECU: virtual Electronic Control Unit
ASW: Application Software
BSW: Base Software
vMCAL: virtual Microcontroller Abstraction Layer

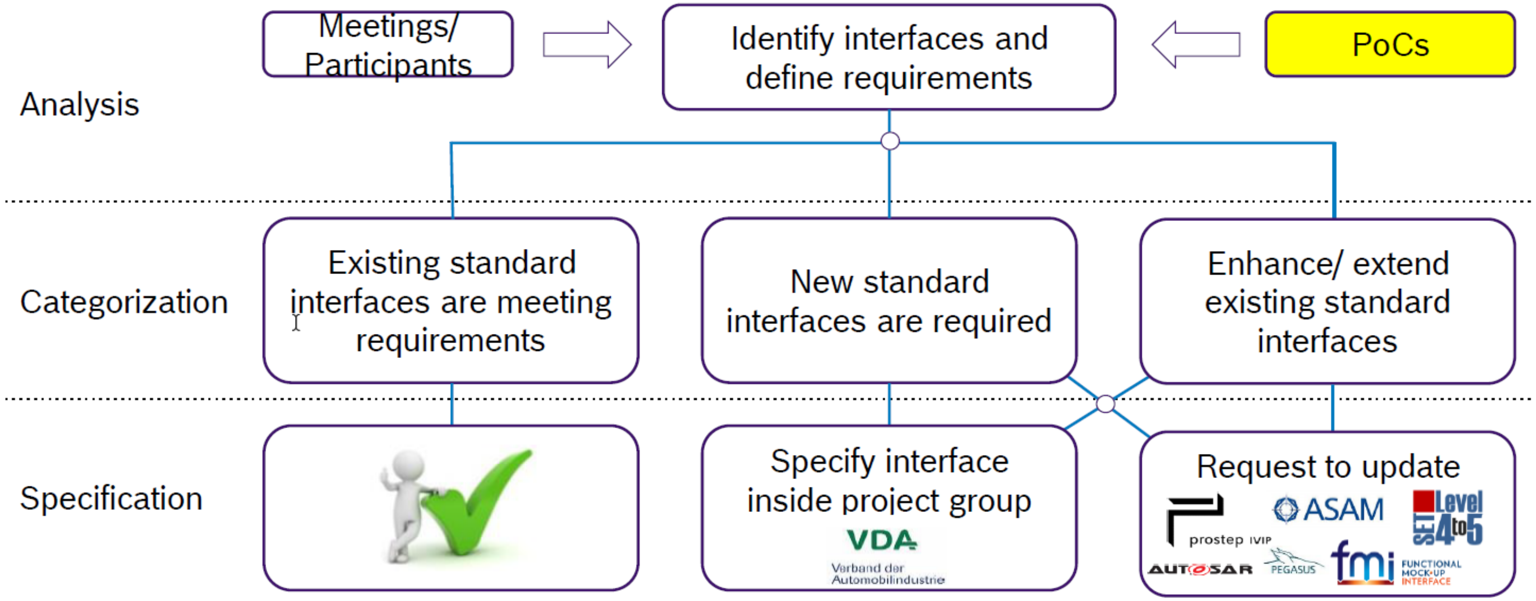
Why Standards for SiL?

- SIL components need to be compatible and therefore standardized, because
  - Functions being distributed across several nodes and domains need to be verified early in SIL environments
    (-> several vECUs to be combined in one SIL setup)
  - **X-domain compatibility**
  - SIL components in projects are coming from different companies (e.g. OEMs / TIER1s / tool provider)
  - **X-company compatibility**
  - Components need to be runnable in different execution platforms (e.g. PC, server, cloud)
  - **X-platform compatibility**

VDA Project Group SiL Standardization

Goals:

- Recommendation for the topic SiL-System Interface Standards
- **No reinvention of existing standards, enhance/ extend Requirements if required**
- Proprietary solutions in the industry shall be replaced by new standards



Analysis

Meetings/ Participants

Identify interfaces and define requirements

PoCs

Categorization

Existing standard interfaces are meeting requirements

New standard interfaces are required

Enhance/ extend existing standard interfaces

Specification

Specify interface inside project group

Request to update

Current VDA Project Group Participants

AVL

BOSCH

MathWorks®

dSPACE

etas

CARIAD

A VOLKSWAGEN GROUP COMPANY

PORSCHE®

STELLANTIS

ZF

AKKA

PASSION FOR TECHNOLOGIES

tracetronic

Valeo

VECTOR

SYNOPSYS®

Automotive SiL Architecture by VDA

SiL-System Level Interfaces

1 V-ECU (IF-Adapter) – SiL Framework

2 V-ECU (IF-Adapter) – Test Automation

3 V-ECU (IF-Adapter) – MCD

4 V-ECU (IF-Adapter) - Virtual E/E

5 model (IF-Adapter) - SiL Framework

6 model (IF-Adapter) - Test Automation

7 SiL Framework - Test Automation

8 SiL Framework – MCD

9 SiL Framework - Virtual E/E

10 Virtual E/E - Test Automation

11 Virtual E/E - MCD

12 GUI Interface

13 Cloud Interface

14 V-ECU (IF-Adapter) - Debugger

15 Test Automation - MCD

16H model (IF-Adapter) – V-ECU (IF-Adapter), High Cut

16L model (IF-Adapter) – V-ECU (IF-Adapter), Low Cut

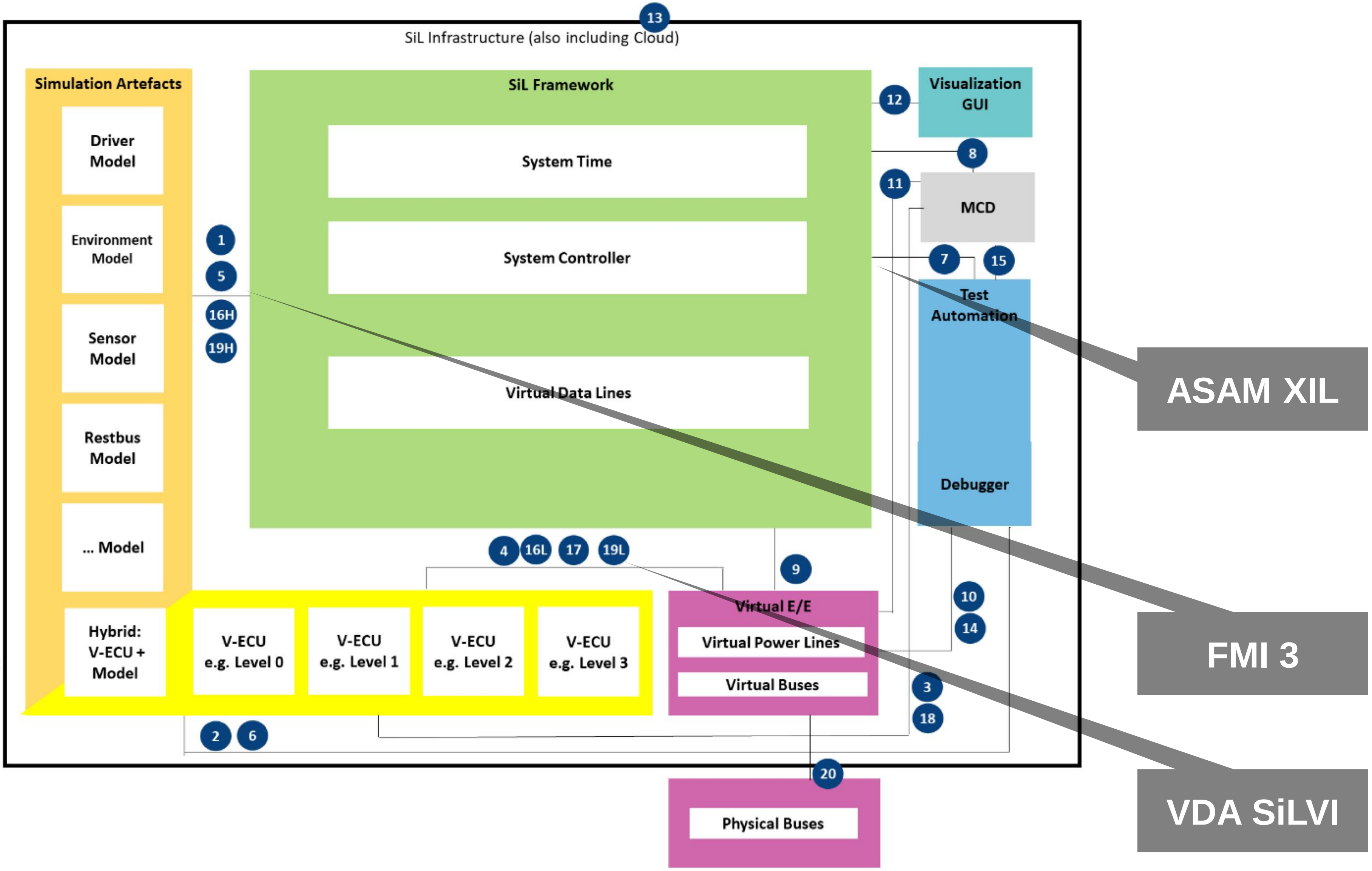
17 model (IF-Adapter) - Virtual E/E

18 model (IF-Adapter) - MCD

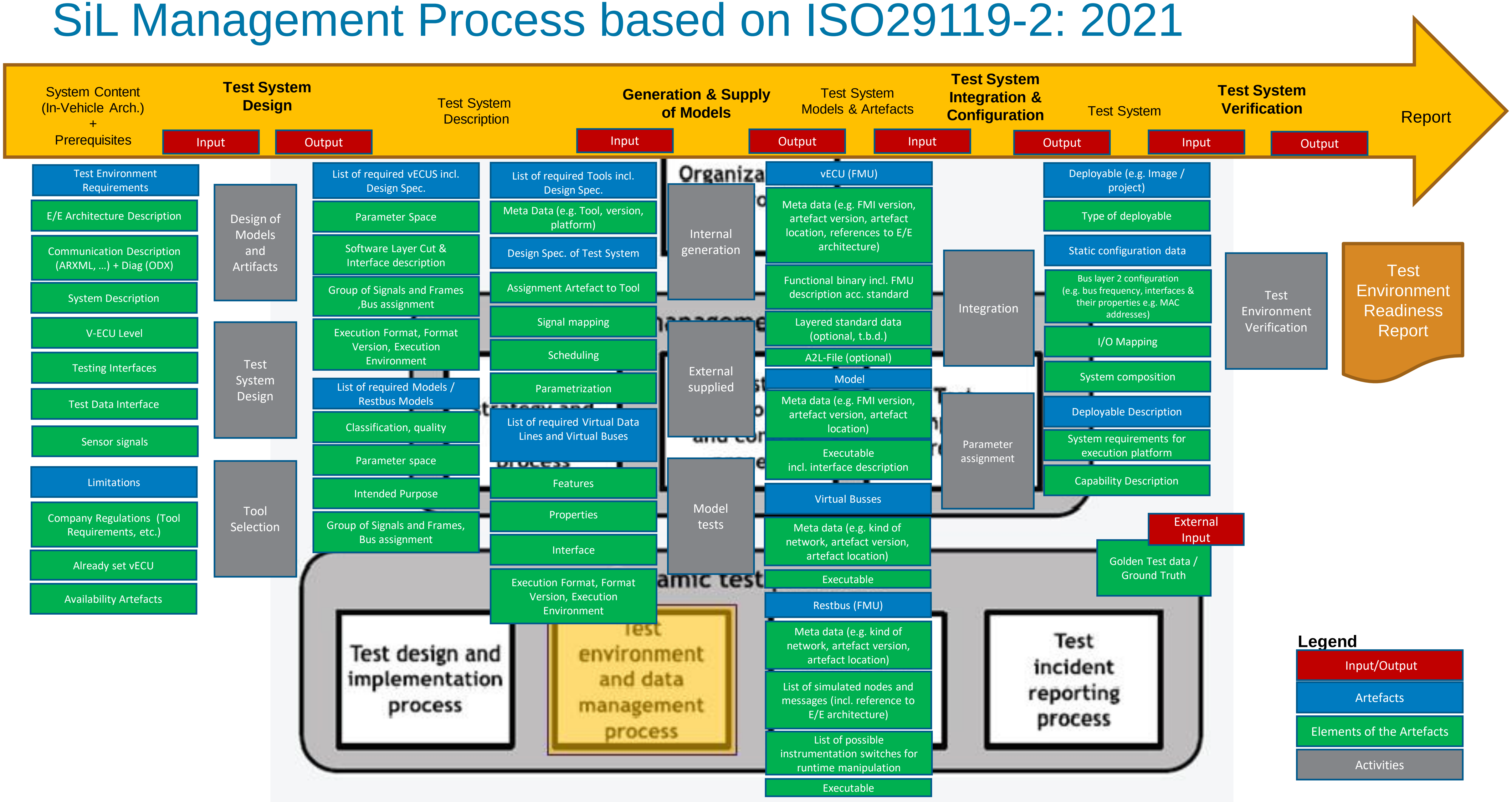
19H V-ECU (IF-Adapter) – V-ECU (IF-Adapter), High Cut

19L V-ECU (IF-Adapter) – V-ECU (IF-Adapter), Low Cut

20 V-ECU (IF-Adapter) – ECU



SiL Management Process based on ISO29119-2: 2021


Organiza
management
strategy and
process
and co
amic test
Test design and implementation process
Test environment and data management process
Test incident reporting process

System Content (In-Vehicle Arch.) + Prerequisites

Test System Design

Input | Output

Test System Description

Generation & Supply of Models

Input | Output

Test System Models & Artefacts

Test System Integration & Configuration

Input | Output

Test System

Test System Verification

Input | Output

Report

- Test Environment Requirements
- E/E Architecture Description
- Communication Description (ARXML, …) + Diag (ODX)
- System Description
- V-ECU Level
- Testing Interfaces
- Test Data Interface
- Sensor signals
- Limitations
- Company Regulations (Tool Requirements, etc.)
- Already set vECU
- Availability Artefacts

- Design of Models and Artifacts
- Test System Design
- Tool Selection

- List of required vECUS incl. Design Spec.
- Parameter Space
- Software Layer Cut & Interface description
- Group of Signals and Frames ,Bus assignment
- Execution Format, Format Version, Execution Environment
- List of required Models / Restbus Models
- Classification, quality
- Parameter space
- Intended Purpose
- Group of Signals and Frames, Bus assignment

- List of required Tools incl. Design Spec.
- Meta Data (e.g. Tool, version, platform)
- Design Spec. of Test System
- Assignment Artefact to Tool
- Signal mapping
- Scheduling
- Parametrization
- List of required Virtual Data Lines and Virtual Buses
- Features
- Properties
- Interface
- Execution Format, Format Version, Execution Environment

- Internal generation
- External supplied
- Model tests

- vECU (FMU)
- Meta data (e.g. FMI version, artefact version, artefact location, references to E/E architecture)
- Functional binary incl. FMU description acc. standard
- Layered standard data (optional, t.b.d.)
- A2L-File (optional)
- Model
- Meta data (e.g. FMI version, artefact version, artefact location)
- Executable incl. interface description
- Virtual Busses
- Meta data (e.g. kind of network, artefact version, artefact location)
- Executable
- Restbus (FMU)
- Meta data (e.g. kind of network, artefact version, artefact location)
- List of simulated nodes and messages (incl. reference to E/E architecture)
- List of possible instrumentation switches for runtime manipulation
- Executable

- Integration
- Parameter assignment

- Deployable (e.g. Image / project)
- Type of deployable
- Static configuration data
- Bus layer 2 configuration (e.g. bus frequency, interfaces & their properties e.g. MAC addresses)
- I/O Mapping
- System composition
- Deployable Description
- System requirements for execution platform
- Capability Description

External Input

- Golden Test data / Ground Truth

- Test Environment Verification

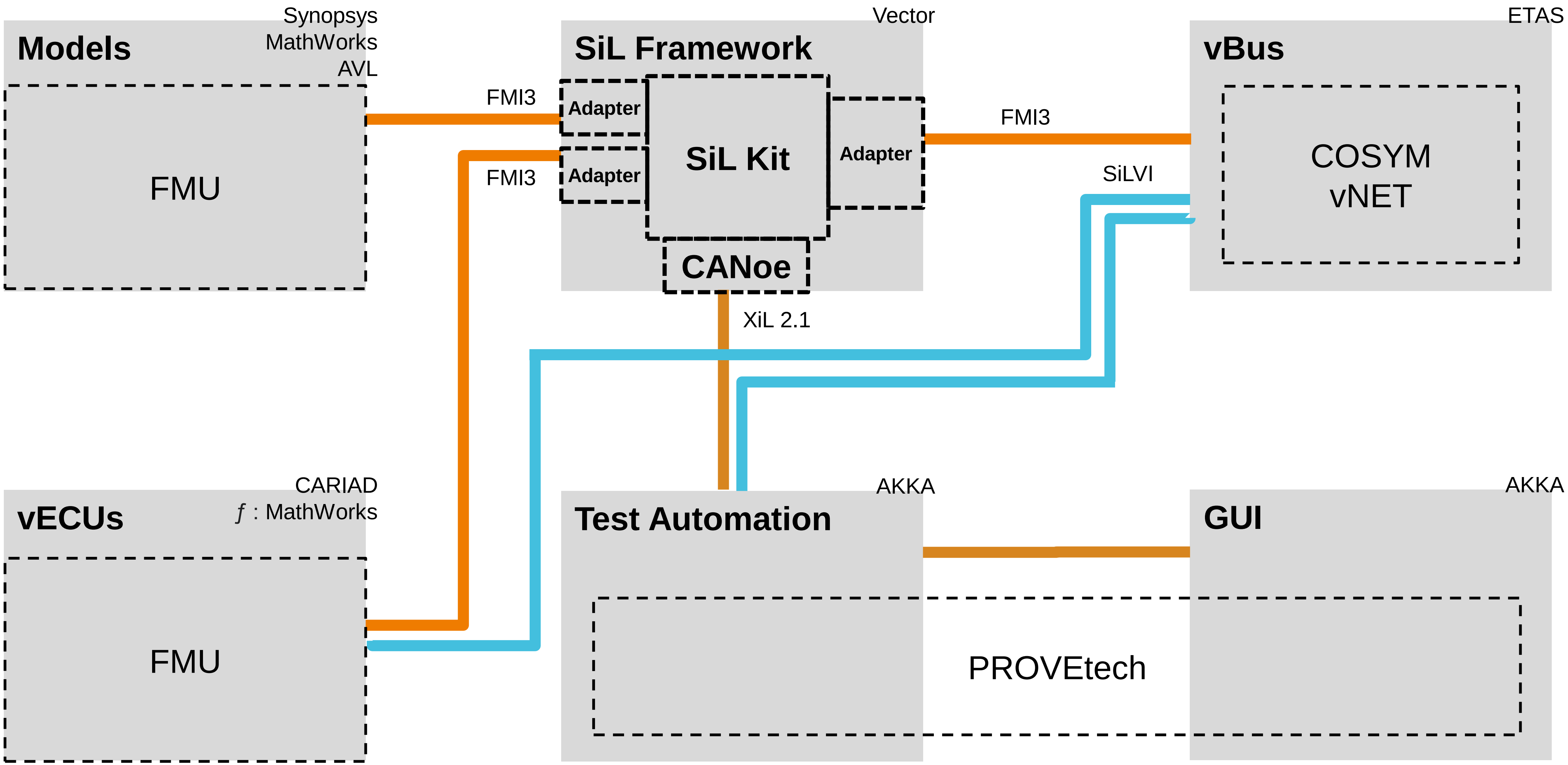
Test Environment Readiness Report

Legend

- 🟥 Input/Output
- 🟦 Artefacts
- 🟩 Elements of the Artefacts
- ⬜ Activities

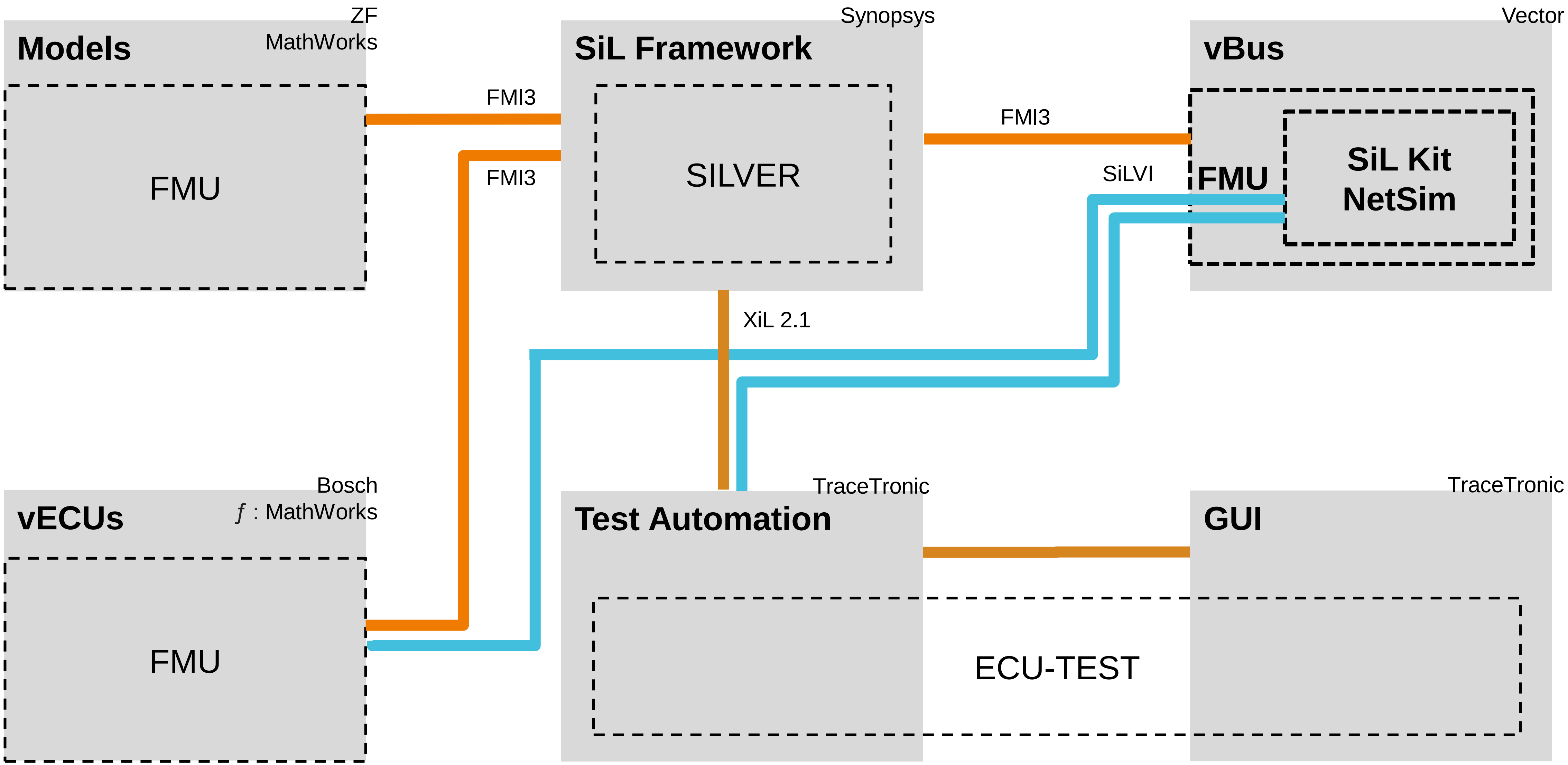
Implementation, Proof of Concept I, Use Case ACC

Integration : Bosch



Implementation, Proof of Concept II, Use Case Window Regulator

Integration : Synopsys



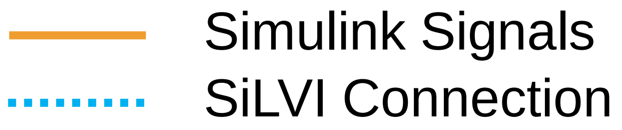
MathWorks participation in VDA SiL standardization project

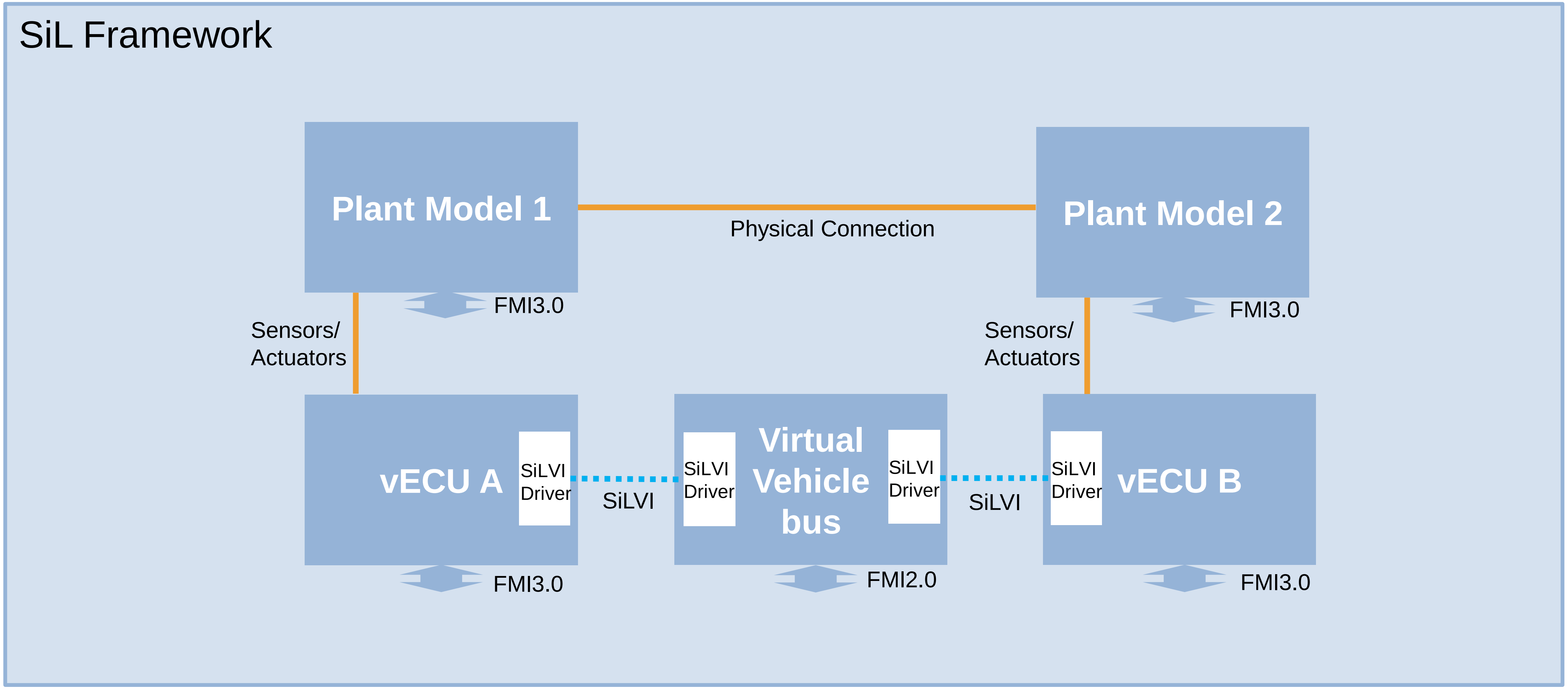
- SiLVI interface defined for standardized coupling of vECUs with virtual vehicle bus system

- FMUs used to integrate SiL artifacts (plant models, vECUs, virtual vehicle bus) into a simulation framework

- MathWorks working on establishing Simulink as a SiL framework
  - FMI 3.0 import and export support starting with R2023b

Simulink as a SiL framework



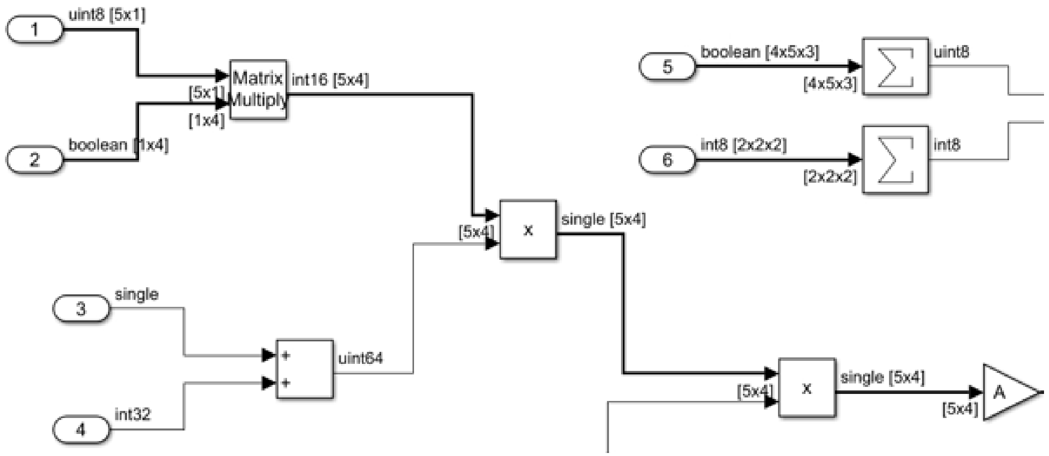


Simulink supports the FMI standard

Starting with R2023b FMI 3.0 is also supported

New FMI 3.0 features in R2023b

- Support for binary, all integer and single-precision data types for I/Os and parameters
  - No casting to double and int32 data types needed like in FMI2.0

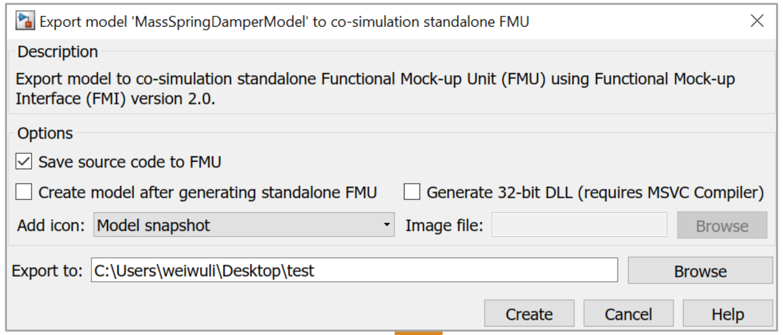


- Support of array data (vector, matrix) for I/Os and parameters

- Direct Feedthrough
  - For Co-Simulation Mode if Event Mode is supported by importing tool
  - If Event Mode is not supported, one step delay like in FMI2.0

Other supported features for FMI 3.0 and FMI 2.0

- Source code FMU export
  - Generated source code in C can be used for cross-platform workflows



Export model 'MassSpringDamperModel' to co-simulation standalone FMU

Description

Export model to co-simulation standalone Functional Mock-up Unit (FMU) using Functional Mock-up Interface (FMI) version 2.0.

Options

☑ Save source code to FMU

□ Create model after generating standalone FMU   □ Generate 32-bit DLL (requires MSVC Compiler)

Add icon: Model snapshot   Image file: Browse

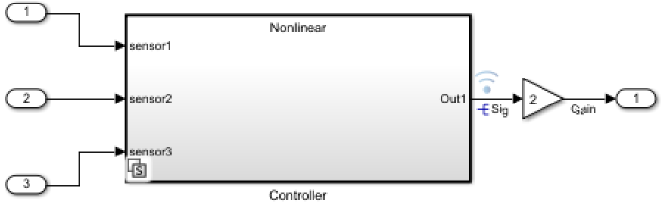
Export to: C:\Users\weiwuli\Desktop\test   Browse

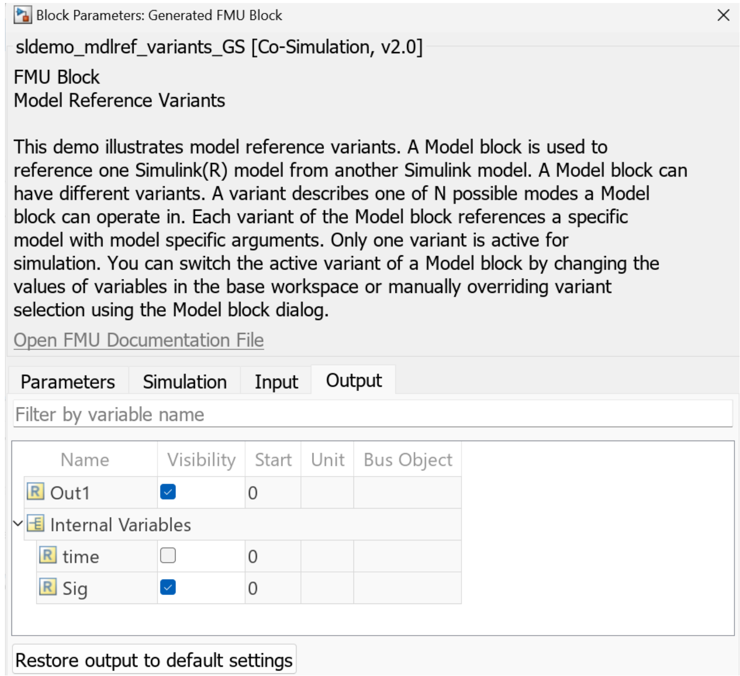
Create   Cancel   Help

Deploy desktop simulation on different OS

Real-time Simulation

- Log internal variables and expose them as FMI outputs





Block Parameters: Generated FMU Block

sldemo_mdlref_variants_GS [Co-Simulation, v2.0]

FMU Block
Model Reference Variants

This demo illustrates model reference variants. A Model block is used to reference one Simulink(R) model from another Simulink model. A Model block can have different variants. A variant describes one of N possible modes a Model block can operate in. Each variant of the Model block references a specific model with model specific arguments. Only one variant is active for simulation. You can switch the active variant of a Model block by changing the values of variables in the base workspace or manually overriding variant selection using the Model block dialog.

Open FMU Documentation File

Parameters | Simulation | Input | Output

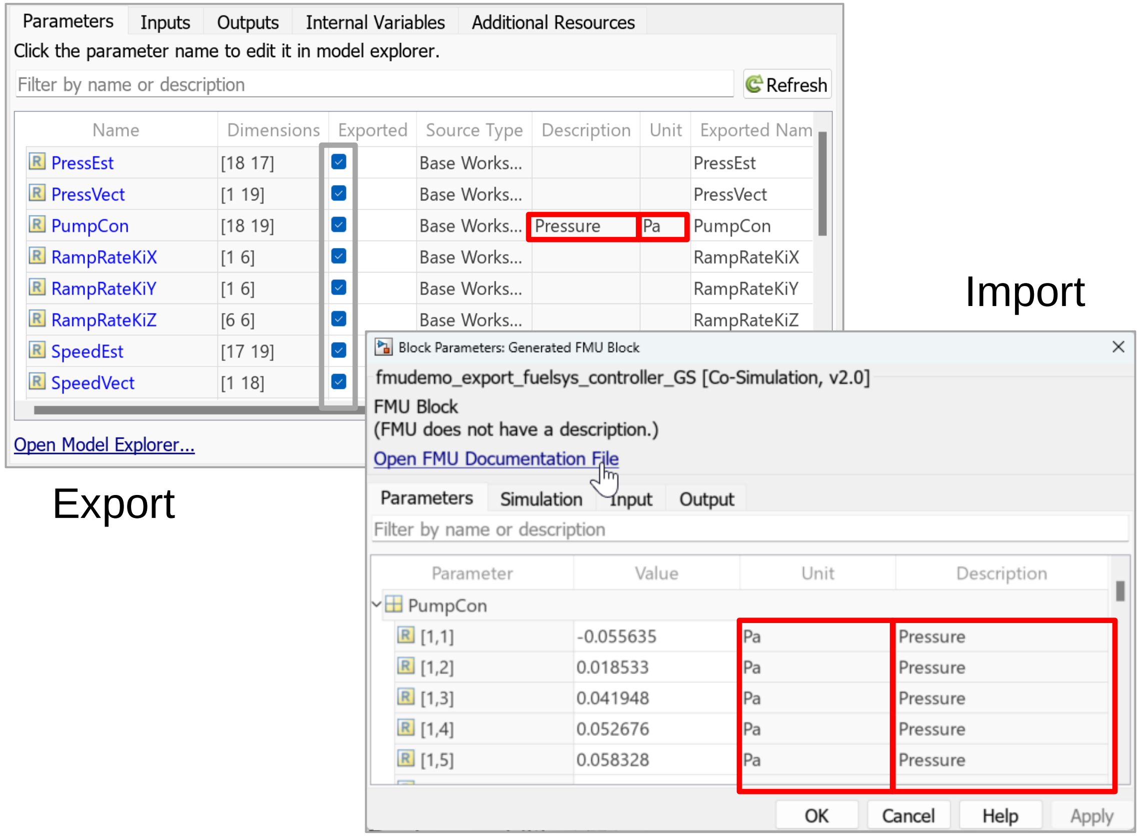
Filter by variable name

Name	Visibility	Start	Unit	Bus Object
Out1	☑	0		
Internal Variables				
time	☐	0		
Sig	☑	0		

Restore output to default settings

Other supported features for FMI 3.0 and FMI 2.0

- Support of units, description and individual selection for parameters during export



Parameters | Inputs | Outputs | Internal Variables | Additional Resources

Click the parameter name to edit it in model explorer.

Filter by name or description

Refresh

Name	Dimensions	Exported	Source Type	Description	Unit	Exported Nam
PressEst	[18 17]	☑	Base Works...			PressEst
PressVect	[1 19]	☑	Base Works...			PressVect
PumpCon	[18 19]	☑	Base Works...	Pressure	Pa	PumpCon
RampRateKiX	[1 6]	☑	Base Works...			RampRateKiX
RampRateKiY	[1 6]	☑	Base Works...			RampRateKiY
RampRateKiZ	[6 6]	☑	Base Works...			RampRateKiZ
SpeedEst	[17 19]	☑				
SpeedVect	[1 18]	☑				

Open Model Explorer...

Export

Import

Block Parameters: Generated FMU Block

fmudemo_export_fuelsys_controller_GS [Co-Simulation, v2.0]

FMU Block
(FMU does not have a description.)

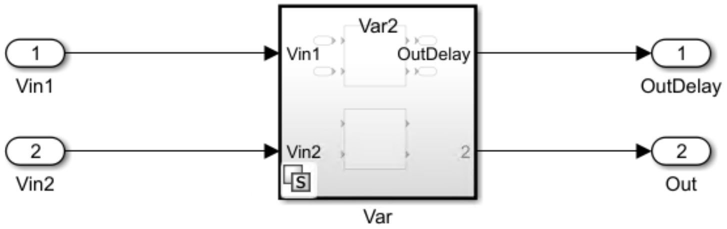
Open FMU Documentation File

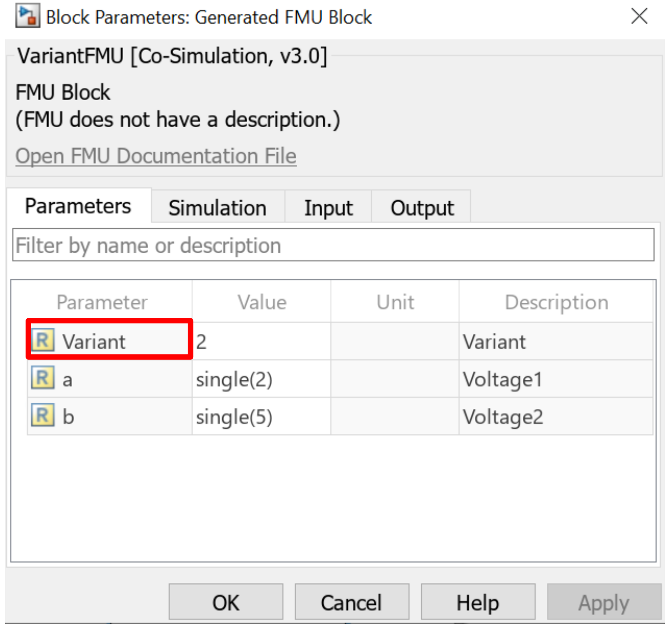
Parameters | Simulation | Input | Output

Filter by name or description

Parameter	Value	Unit	Description
PumpCon			
[1,1]	-0.055635	Pa	Pressure
[1,2]	0.018533	Pa	Pressure
[1,3]	0.041948	Pa	Pressure
[1,4]	0.052676	Pa	Pressure
[1,5]	0.058328	Pa	Pressure

- Export and import a model with variants
  - Use Startup Variant





Block Parameters: Generated FMU Block

VariantFMU [Co-Simulation, v3.0]

FMU Block
(FMU does not have a description.)

Open FMU Documentation File

Parameters | Simulation | Input | Output

Filter by name or description

Parameter	Value	Unit	Description
Variant	2		Variant
a	single(2)		Voltage1
b	single(5)		Voltage2

Key Takeaways

- SiL is one of the main continuous testing environments for the automotive industry
- SiL components need to be compatible and standardized for flexible tool selection and by reduction of proprietary Interfaces. SiL standards enable highly automated SiL environment generation
- Standardization is the basis for scalable Cloud based CX Integration platforms
- MathWorks is collaborating with the industry through **VDA SiL Standardization project group** to demonstrate proof of concept for SiL workflow
- MathWorks has support for FMI 2.0 and 3.0

MathWorks

AUTOMOTIVE CONFERENCE 2023

Europe

Thank you

MathWorks®