DENSO
Crafting the Core

PMSF
IT Consulting

:em engineering methods AG

MBD of a TCU* with SSP and FMI

Hasan Esen, Nicolas Ochoa — DENSO Automotive Deutschland GmbH
Pierre R. Mai — PMSF IT Consulting, Germany
Klaus Mai — :em AG, Germany
Hiroshi Tashiro — DENSO CORPORATION, Japan

23.09.2021, Modelica Conference

*Transmission Control Unit

Contents

- Background
- Proof of Concept Development for TCU Design
- MBD Toolchain and FMI/SSP Architecture
- Conclusion & Outlook

Background

Collaboration between engineering teams, whether between two companies or within one, often requires exchanging simulation models of multi-component systems.

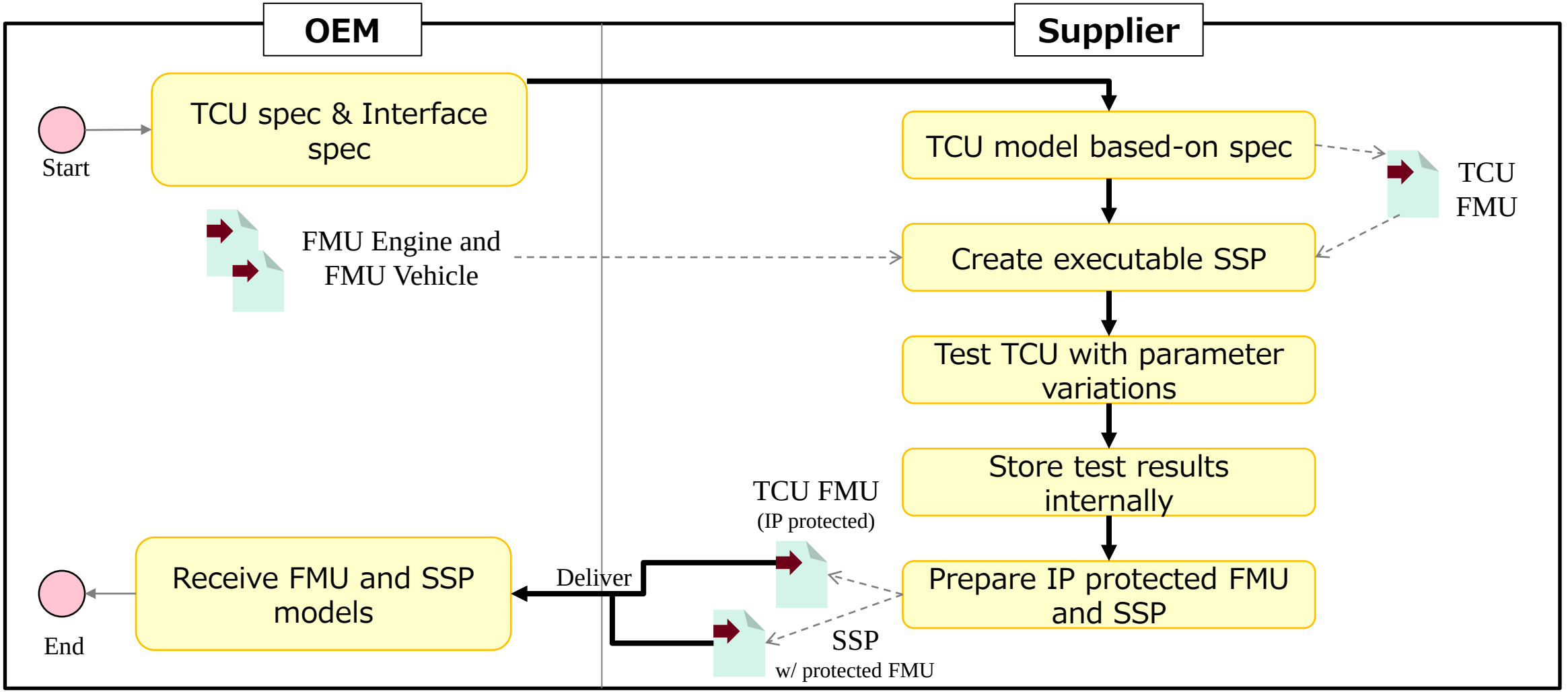
➔ Technical Challenges:

- Parameter handling in integrated simulation
- IP protection of exchanged models
- Assessing simulation performance

The authors developed a co-simulation demonstrator to evaluate how the SSP and FMI standards can facilitate such collaborations.

Proof of Concept (PoC) Development for TCU Design

Use Case*: Supplier to design, test and calibrate Transmission Control Unit (TCU) based on OEM specifications and requests.



*This is an example use case. It does not represent any real business case.

PoC Development – Generating FMUs

TCU model based-on spec

↓

Create executable SSP

↓

Test TCU with parameter variations

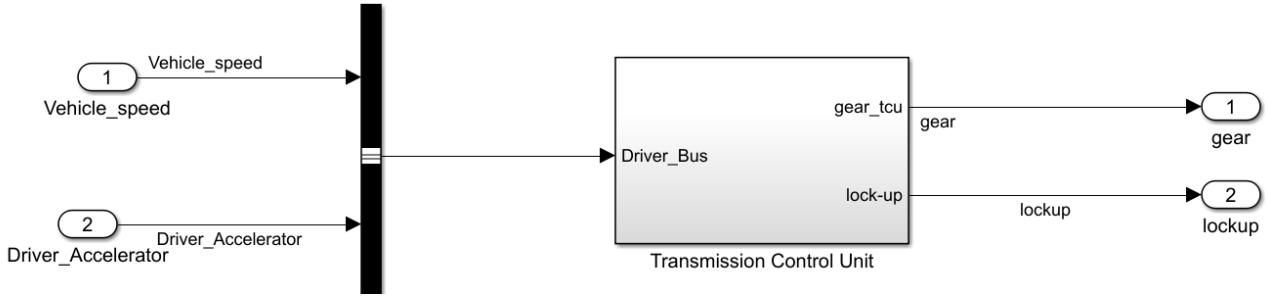
↓

Store test results internally

↓

Prepare IP protected FMU and SSP

- The TCU is developed in MATLAB / Simulink R2020a



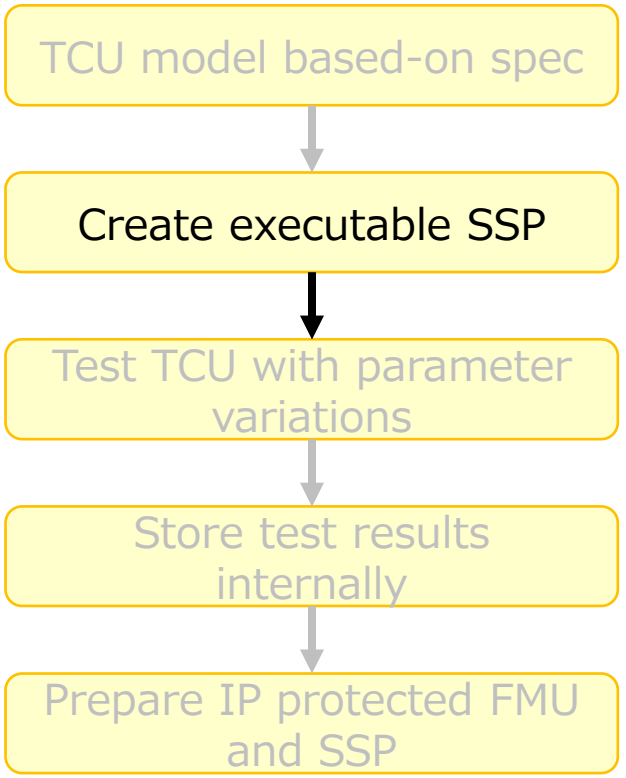
- Using Modelon FMI Toolbox an FMU 2.0 Co-Simulation FMU is generated

TCU FMU

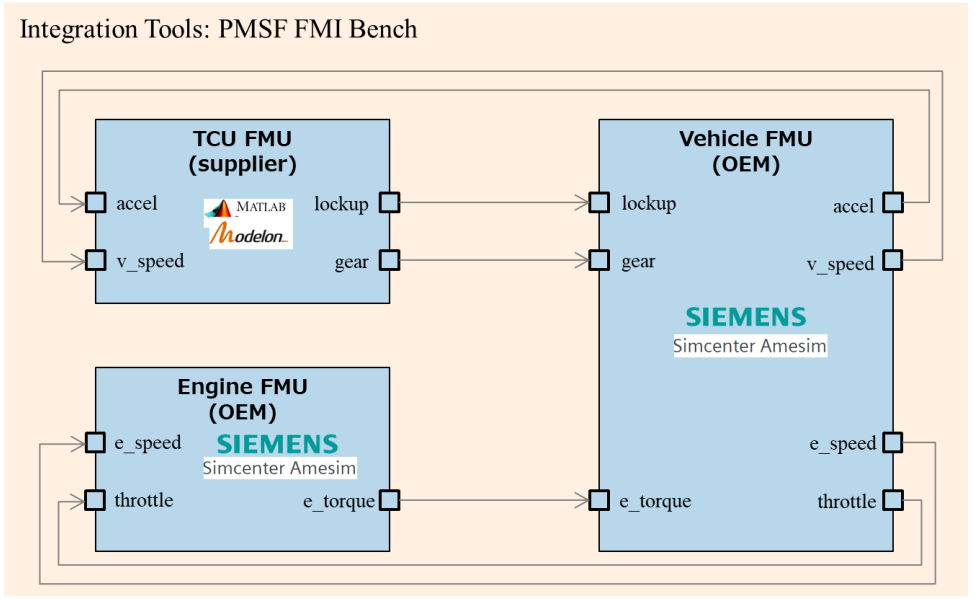
- Engine and vehicle FMUs were generated by partners acting as OEM using SIEMENS Simcenter Amesim

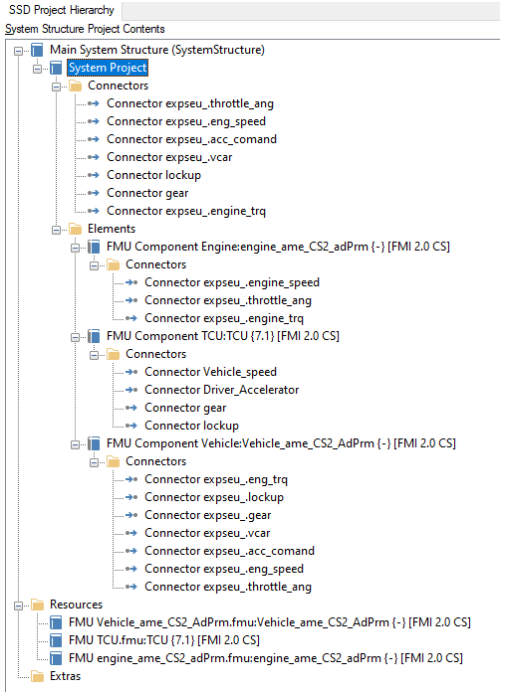
FMU Engine and FMU Vehicle

PoC Development – Executable SSP



- Using PMSF FMI Bench an SSD Project is created
- 3 FMUs (vehicle, engine, TCU) are added to the SSD Project
- Connectors are added for all inputs, outputs and parameters to make them accessible from the outside of the simulation
- FMU components are connected to close the loop.





PoC Development – Stand-alone Simulator

TCU model based-on spec

↓

Create executable SSP

↓

Test TCU with parameter variations (1/2)

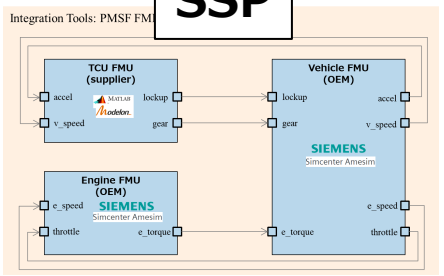
↓

Store test results internally

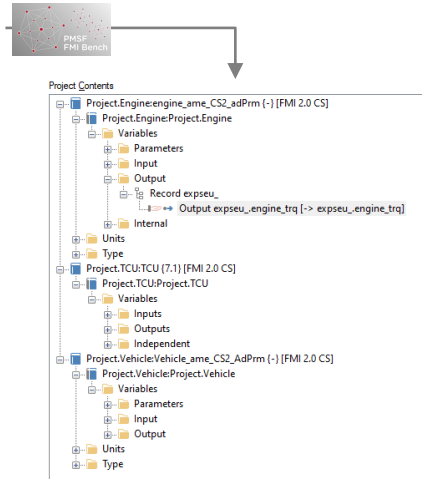
↓

Prepare IP protected FMU and SSP

SSP



Generate a FMI Bench native project



Generate a stand-alone simulator

Simulator.exe

```
-h             Print help and exit.
-v             Print version information and be verbose.
-l             Print logging messages from FMUs.
-c             Print CPU time information at end.
-lt            Print current simulation time every 5 seconds.
-T start       Set simulation start time in seconds (Default: 0.000000).
-t stop        Set simulation stop time in seconds (Default: 1180.000000).
-s step        Set simulation step size in seconds (Default: 0.001000).
-pr name val   Set real parameter named name to value val.
-pi name val   Set integer parameter named name to value val.
-pb name val   Set boolean parameter named name to value val (0 or 1).
-ps name val   Set string parameter named name to value val.
-p parfile     Read parameters from parfile.
-i infile      Read simulation inputs from infile.
-o outfile     Write all exported output variables to outfile.
-ov name       Add variable name to variables to be recorded.
-d delimiter   Use delimiter instead of comma for CSV I/O.
-r rate        Only write output for every rate-th step.
```

PoC Development – TCU Parameter Variations

TCU model based-on spec

Create executable SSP

Test TCU with parameter variations (2/2)

Store test results internally

Prepare IP protected FMU and SSP

Variation test scripting with Jupyter Notebooks

jupyter TCU Parameterization Testruns

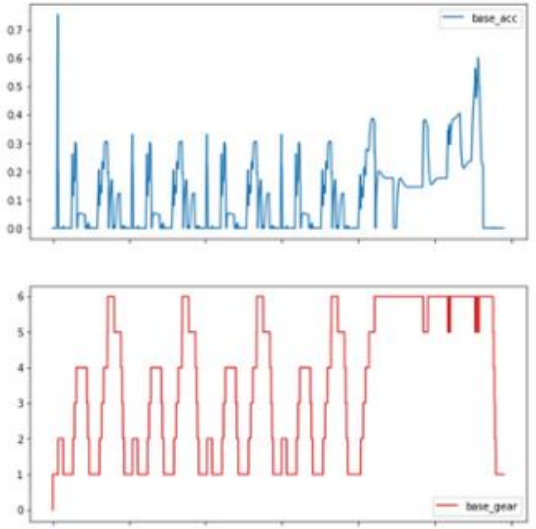
Initialization:
- Import libraries & Setup
- Parallel simulations
- TCU Baseline parametera

Simulation Loop:
- Parameter variations
- Run simulations
- Save output data

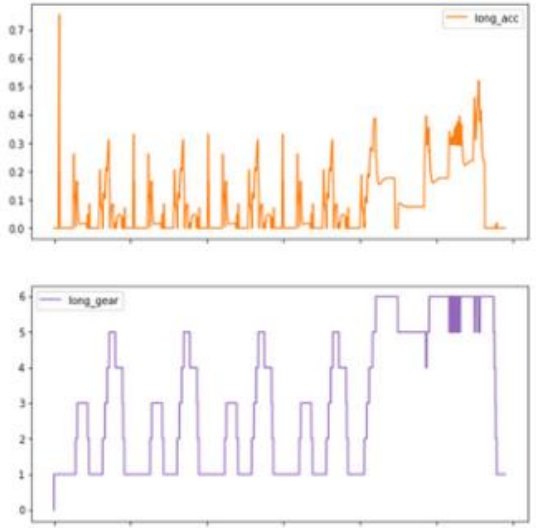
Simulator.exe

Results

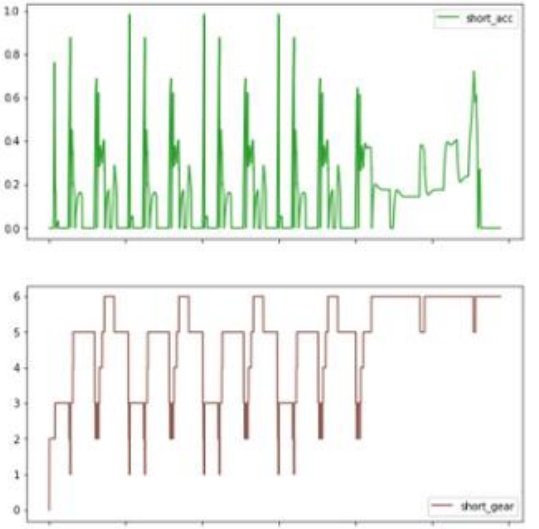
Baseline



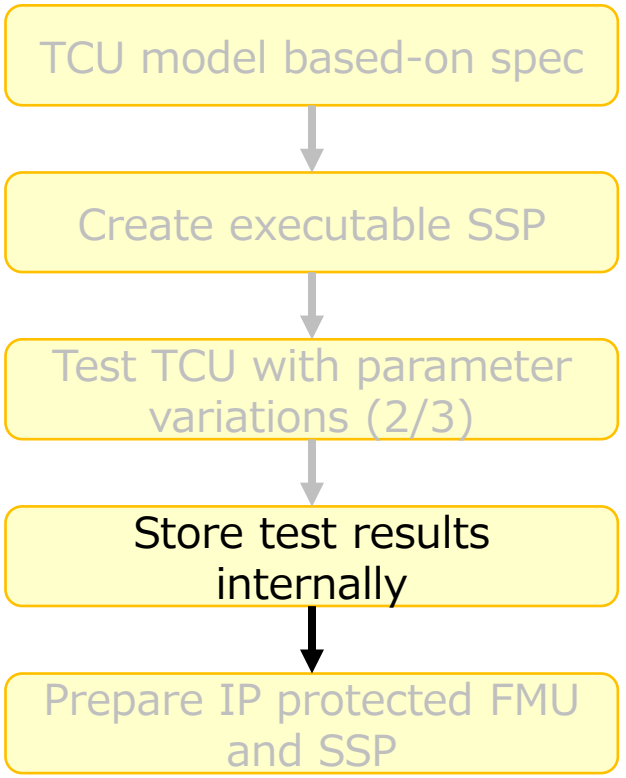
Var-I



Var-II



PoC Development – Simulation Data Management



- This is an on-going work with :em AG.
- User stories:
  1. Basis data and meta data management capabilities
  2. Traceability of simulation results
- Example Workflow for User Story 1:
  - Model import and identification
  - Lifecycle management inside SDM environment (release / change models)
  - Edit metadata of a model inside SDM environment
  - Manage FMUs and SSP files
  - Create group element for clustering models
  - …

PoC Development – IP protection

TCU model based-on spec

Create executable SSP

Test TCU with parameter variations (2/3)

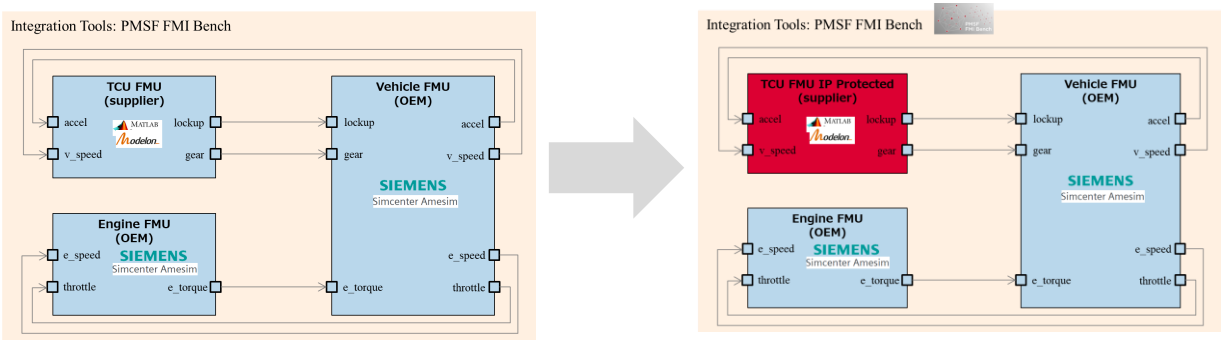
Store test results internally

Prepare IP protected FMU and SSP

Generate a wrapped FMU to provide IP protection to the supplier, before TCU FMU is delivered to the OEM.

- The exposed parameters are stripped out/hidden
- The FMU is encrypted and modified to make it less attackable
- A licensing module is inserted into the wrapper to control the use of the FMU

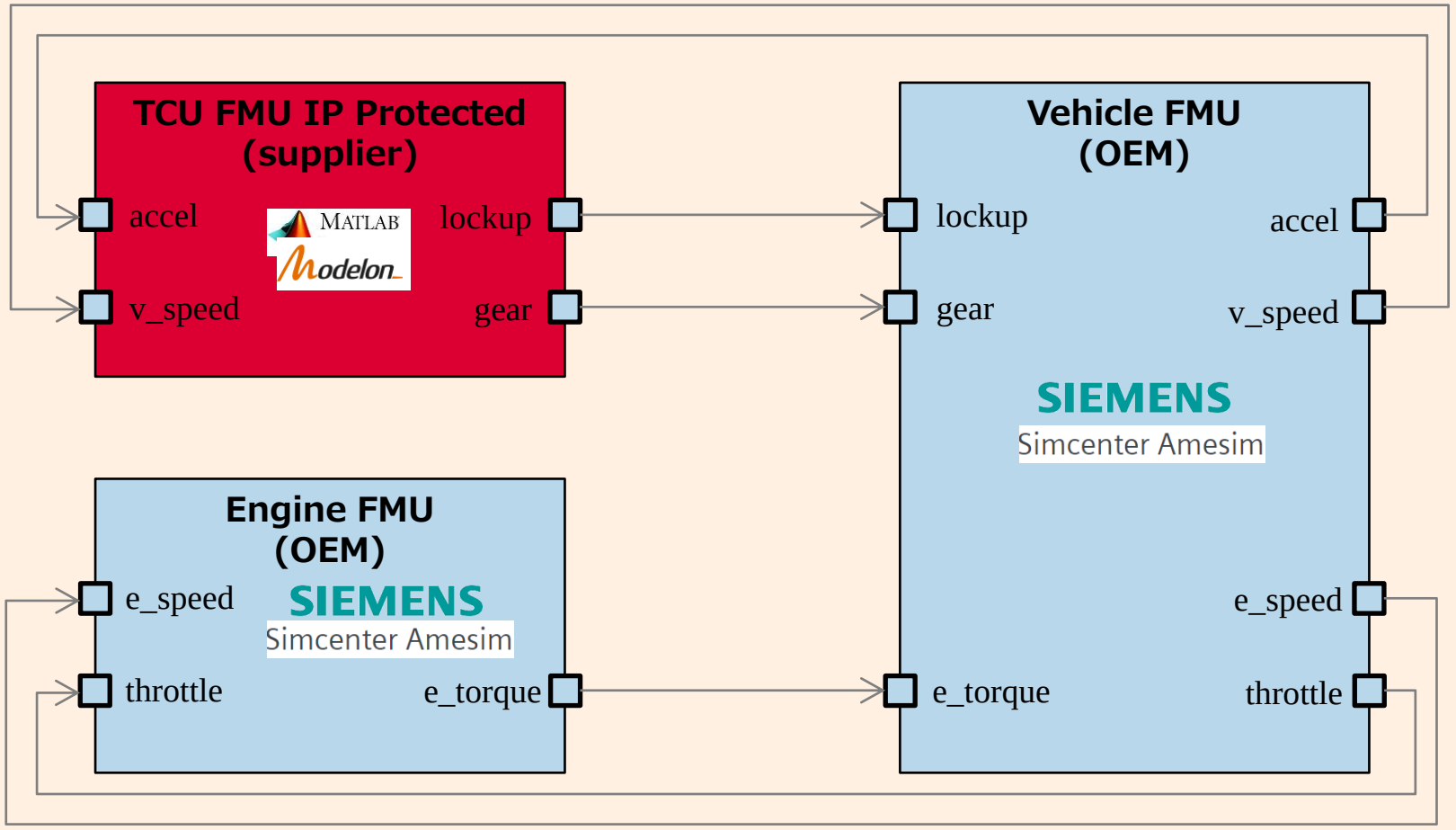
➔ FMU parameters are not accesible anymore, but the FMU interface itself is not changed. Therefore, initial TCU FMU can be replaced with IP-protected TCU FMU easily in the SSP



Proof of Concept (PoC) Development for TCU Design

PoC Architecture and Toolchain:

Integration Tools: PMSF FMI Bench



Evaluation of results: Jupyter Notebook

Engine Model →

Internal combustion engine

Vehicle model →

rest of the vehicle. Maintains the engine state, vehicle state; provides accelerator and throttle positions

TCU model →

provides transmission lockup and gear ratio information, based on the vehicle speed / accelaration information.

Conclusions

- Generated an SSP package with multi FMUs as an example collaborative development framework
- The FMI / SSP standard offers new ways to exchange models between OEM and supplier or also internally between departments, considering compatibility and IP protection at the same time.
- The parameter handling features of FMI and SSP allow the flexibility to expose parameters during internal testing, but also hide them when they contain IP-sensitive information.
- It is difficult to build a working SSP with integrated FMUs. Different tools have different constraints regarding the usage of the FMI standard (naming, solver, IP protection mechanisms, …)
- Integration tools like FMI Bench, allow to perform modifications on the FMUs to partially increase the compatibility and performance of FMU and SSP models.

The benefits that can be achieved by using FMI and SSP prevail the possible limitations. These standards offer the opportunity to deal with system-driven design and development, which is essential for today's complex and SW-intensive products.

Outlook

- Simulation Data Management
- Bridging System Requirements and Simulation Requirements
- Interaction with other data-exchange formats such as OSI in AD/ADAS applications

DENSO

Crafting the Core