MA-Project “System Structure and Parameterization” – Early Insights

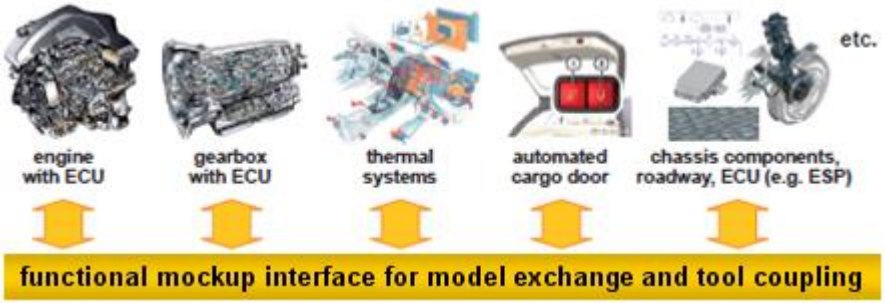
presented by Jochen Köhler (ZF)

Japanese Modelica Conference 2016-05-23/24
Tokyo / Japan

M. Deppe
J. Köhler
J. Krasser
P. R. Mai
M. Nagasawa

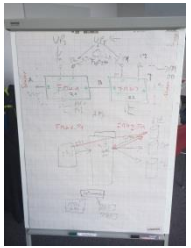
Motivation for initiating MAP “System Structure and Parameterization” (SSP) – Using FMI as Basis

- FMI is basically a great technology to make exchanging models inside and among companies much easier
- Typical use-case is a network of FMUs (System structure) …



- … Therefore, some features are missing …

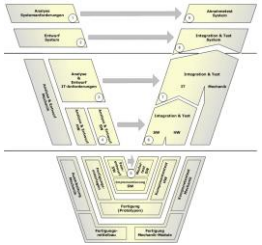
Motivation for initiating SSP – Missing features



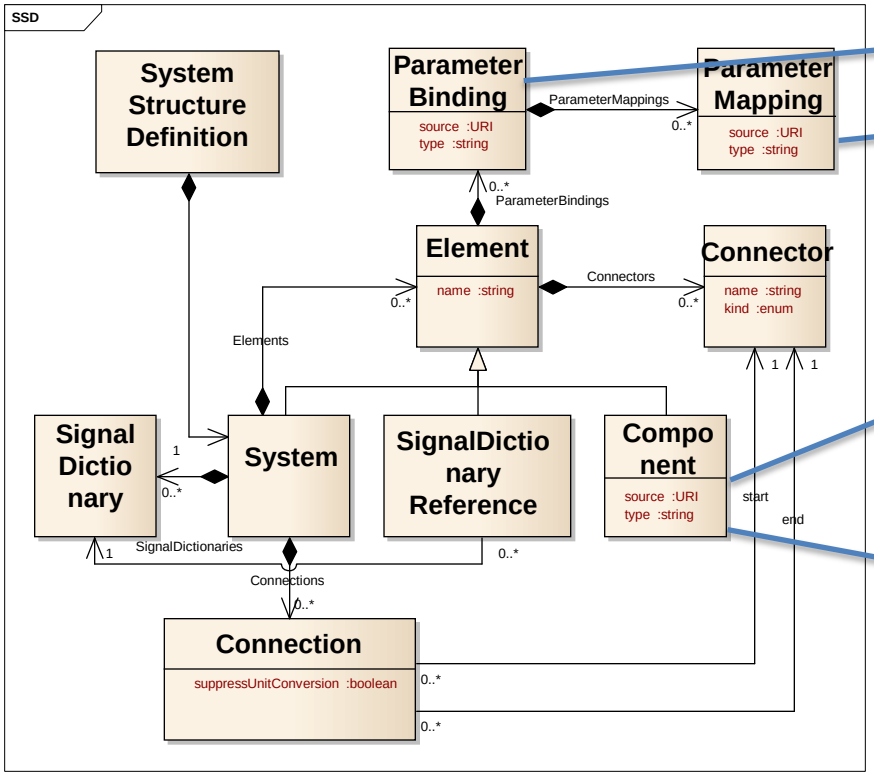
- Collected on a meeting with BMW, Bosch, ZF, PMSF (2014):
  - No possibility to separat parameter data from the FMUs
  - No possibility to change parameters in a consistent way independently from the integration environment for single FMUs.
  - No possibility to handle intellectual property of parameters
  - No possibility of mapping parameters in a network of FMUS
  - No possibility to store a network of FMUs tool independently

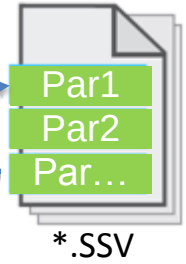
Main Purposes of SSP

- Define a standardized format for the connection structure of a network of components.
- Define a standardized way to store and apply parameters to these components.
- The developed standard / APIs should be usable in all stages of development process (architecture definition, integration, simulation, test in MiL, SiL, HiL).
- The work in this project shall be coordinated with other standards and organizations (FMI, ASAM, OMG).

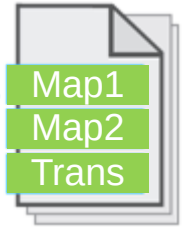


Overview of XML Schema Definitions

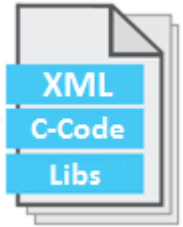




*.SSV



*.SSM

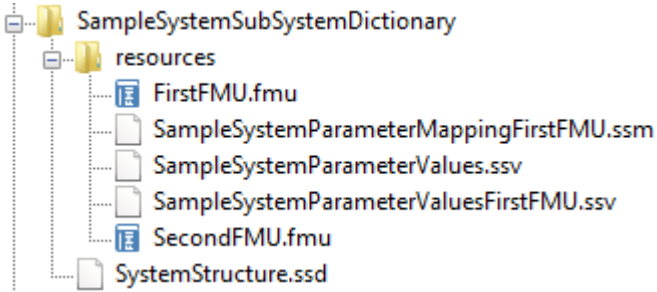


*.FMU

*.SSP

XML Schema Description - System Structure Package

*.SSP



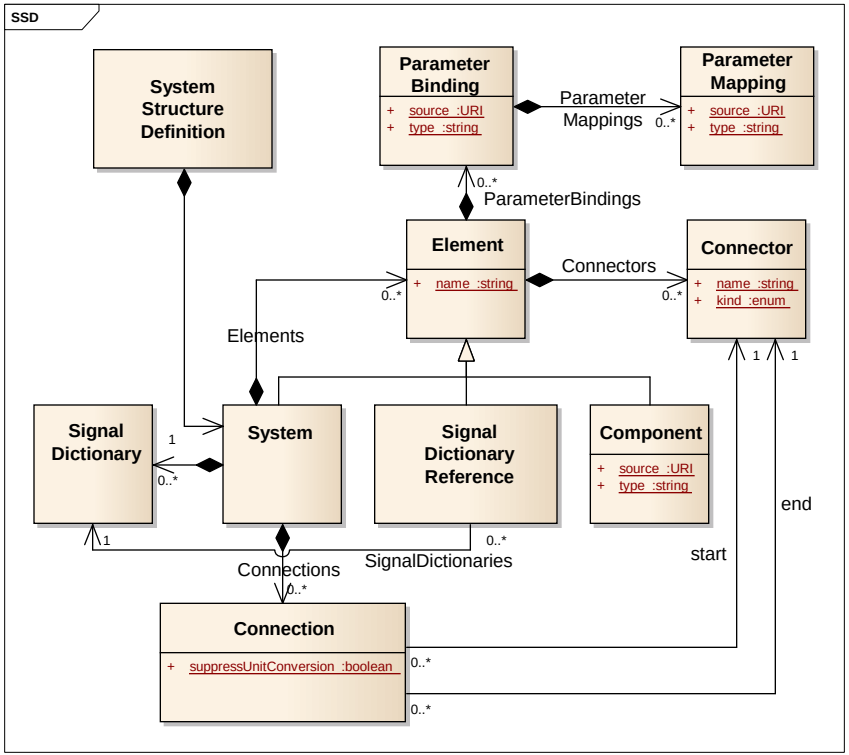
Use case

- Exchange of Complete Systems with Variants

Features

- All information (FMUs, system structure definition, parameters) can be stored in one archive (zip-file)
- Multiple SSDs in one SSP allows for variant modeling

XML Schema Description - System Structure Definition



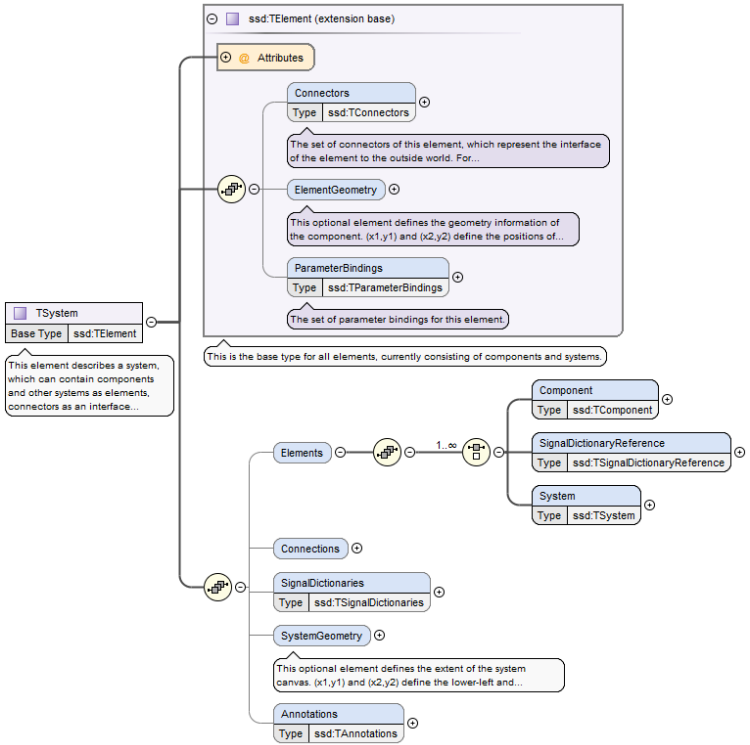
Use case

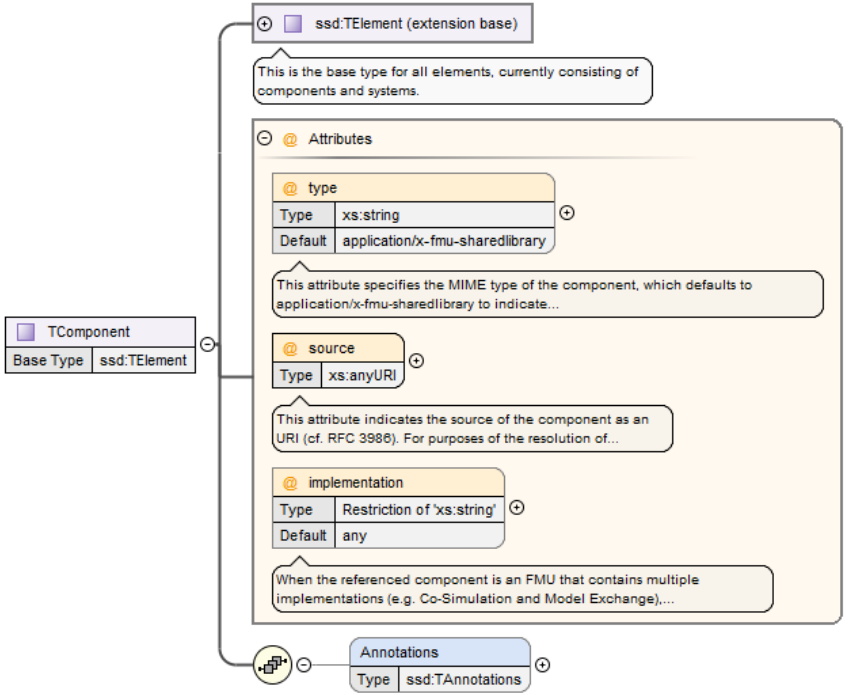
- Defining a Network of FMUs

Features

- Hierarchical sub-systems
- Empty components/FMUs as interface templates
- External resources via URIs: Both relative to SSD/SSP or absolute, e.g. via HTTP(S).
- Connections with unit conversions and optional linear/map transformations
- Optional: Diagram geometry

XML Schema Description - System Structure Definition

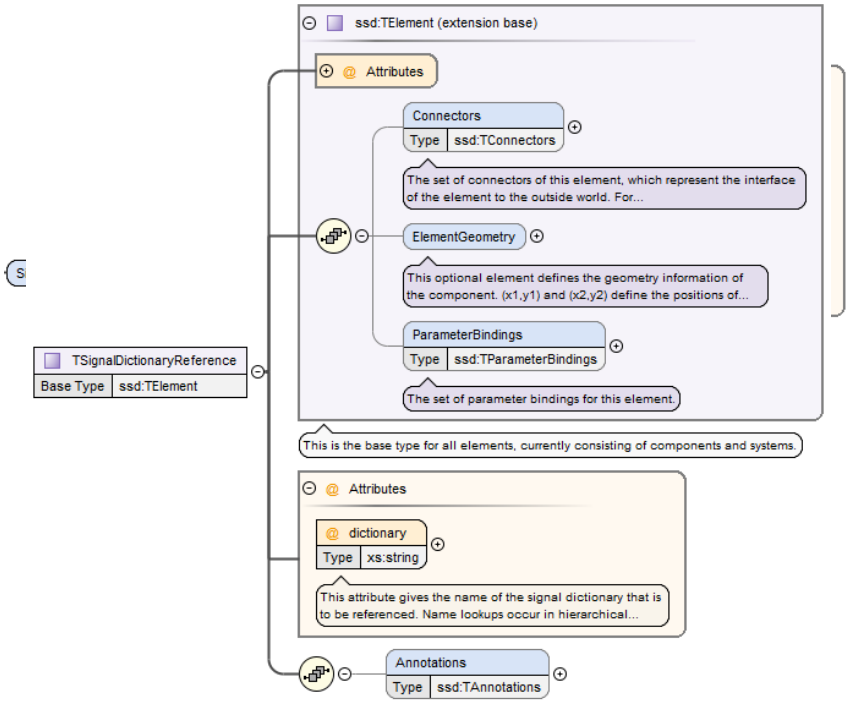




XML Schema Description - System Structure Definition

```
<ssd:Elements>
    <ssd:System name="SubSystem">
        <ssd:Connectors>
            <ssd:Connector name="LocalIn1" kind="input"><ssd:Real unit="m/s"/></ssd:Connector>
            <ssd:Connector name="Out1" kind="output"><ssd:Real unit="m/s"/></ssd:Connector>
            <ssd:Connector name="Out3" kind="output"><ssd:Real unit="m/s"/></ssd:Connector>
        </ssd:Connectors>
        <ssd:Elements>
            <ssd:SignalDictionaryReference dictionary="MyDictionary" name="MyDict">
                <ssd:Connectors>
                    <ssd:Connector name="Var2" kind="inout"><ssd:Real unit="m/s"/></ssd:Connector>
                    <ssd:Connector name="Var4" kind="inout"><ssd:Real unit="m/s"/></ssd:Connector>
                </ssd:Connectors>
            </ssd:SignalDictionaryReference>
            <ssd:Component name="FirstFMUInstance1" source="resources/FirstFMU.fmu" type="application/x-fmu-sharedlibrary">
                <ssd:Connectors>
                    <ssd:Connector name="In1" kind="input"><ssd:Real unit="m/s"/></ssd:Connector>
                    <ssd:Connector name="Out1" kind="output"><ssd:Real unit="m/s"/></ssd:Connector>
                    <ssd:Connector name="Out2" kind="output"><ssd:Real unit="m/s"/></ssd:Connector>
                </ssd:Connectors>
                <ssd:ParameterBindings>
                    <ssd:ParameterBinding source="resources/SampleSystemParameterValuesFirstFMU.ssv" type="application/x-ssp-parameter-set">
                        <ssd:ParameterMapping source="resources/SampleSystemParameterMappingFirstFMU.ssm" type="application/x-ssp-parameter-mapping"/>
                    </ssd:ParameterBinding>
                </ssd:ParameterBindings>
            </ssd:Component>
            <ssd:Component name="FirstFMUInstance2" source="resources/FirstFMU.fmu" type="application/x-fmu-sharedlibrary"> [6 lines]
        </ssd:Elements>
        <ssd:Connections>
            <ssd:Connection startConnector="LocalIn1" endElement="FirstFMUInstance1" endConnector="In1"/>
            <ssd:Connection startConnector="LocalIn1" endElement="FirstFMUInstance2" endConnector="In1"/>
```

XML Schema Description - Signal dictionaries



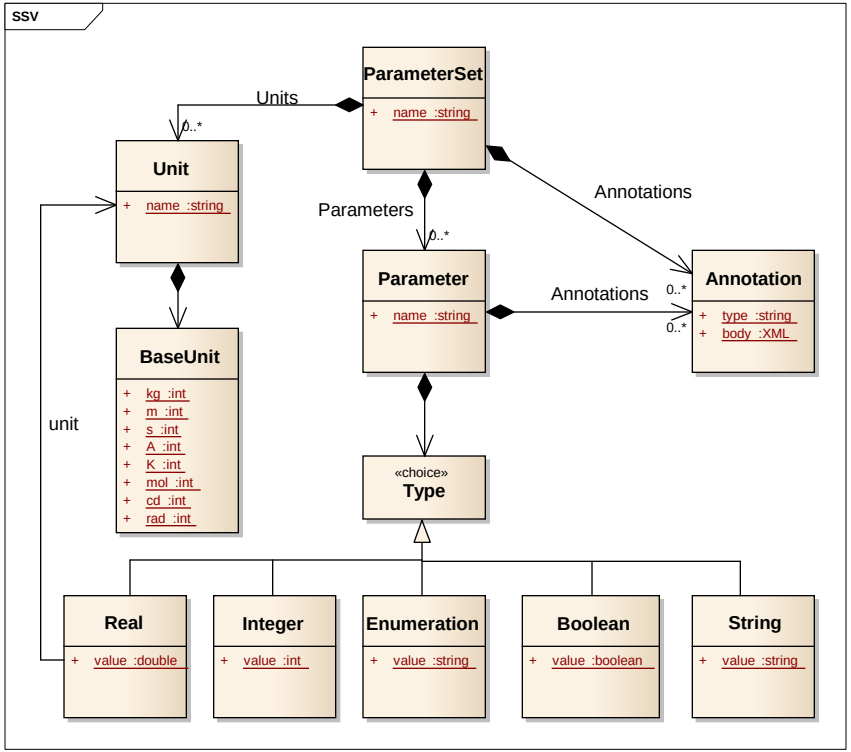
Use cases

- Collecting Control Signals in a Central Location

Features

- Causality is checked by tool automatically
- Crosses hierarchies without need for downward passing
- Well-suited for e.g. ECU control busses

XML Schema Description – Parameter Values Data



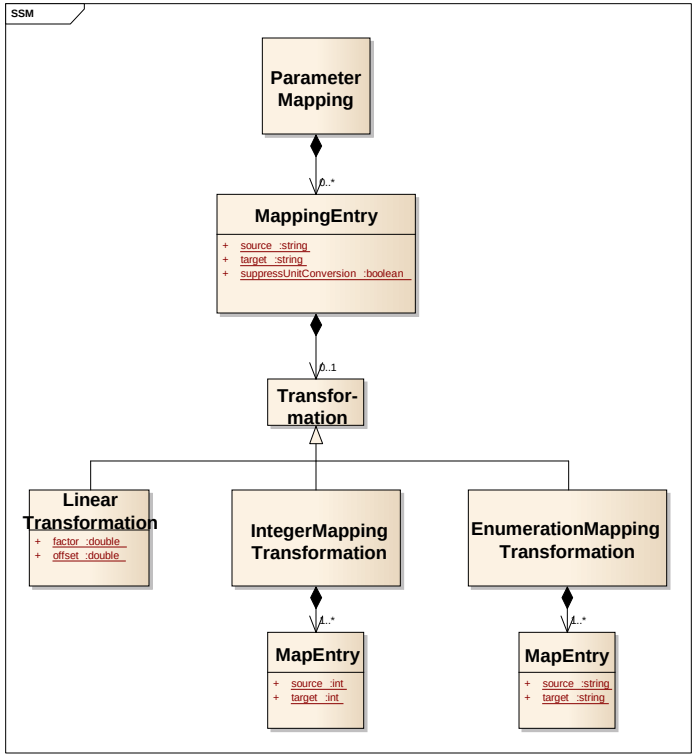
Use case

- Tool-independent Exchange of Parameter Data

Features

- Neutral exchange format between parameter sources
- Compatible to FMI standard
- Provides some meta data
- Access to param DBs via HTTP (-> Parameter API)

XML Schema Description - Parameter Mapping



Use case

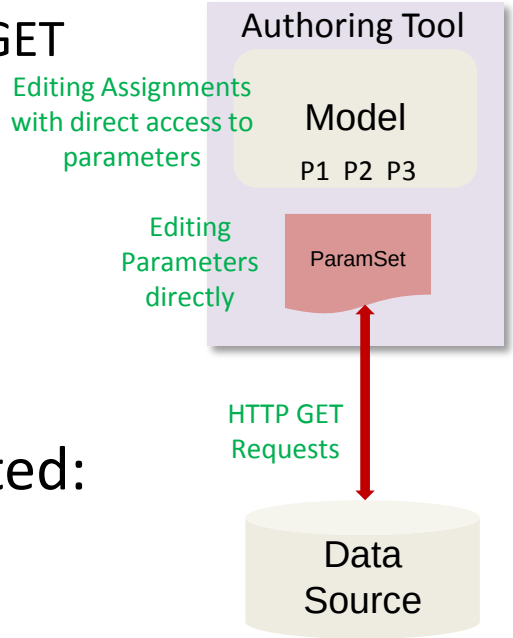
- Mapping Parameters to FMUs when the Parameter Names differ or Parameter Values require Transformations

Features

- Can be stored separately from System Structure and Parameter Data
- Can be inlined into SSD
- Optional manual linear and mapping transformations

Parameter API Get Mechanisms

- General Idea:
  - Access to external parameter sources via HTTP(S) GET Requests
  - Request URI is the source attribute
  - Type attribute passed via accept request header
  - Updates handled efficiently via ETag/Conditional GET/HEAD
- Returns Parameter Data in the format requested:
  - application/x-ssp-parameter-set -> SSV file format
  - Sources and tools can support other formats



Parameter API Get Mechanisms

```
GET /context/ParamSetA HTTP/1.1
Host: pardb.example.com:80
Accept: application/x-ssp-parameter-set
```

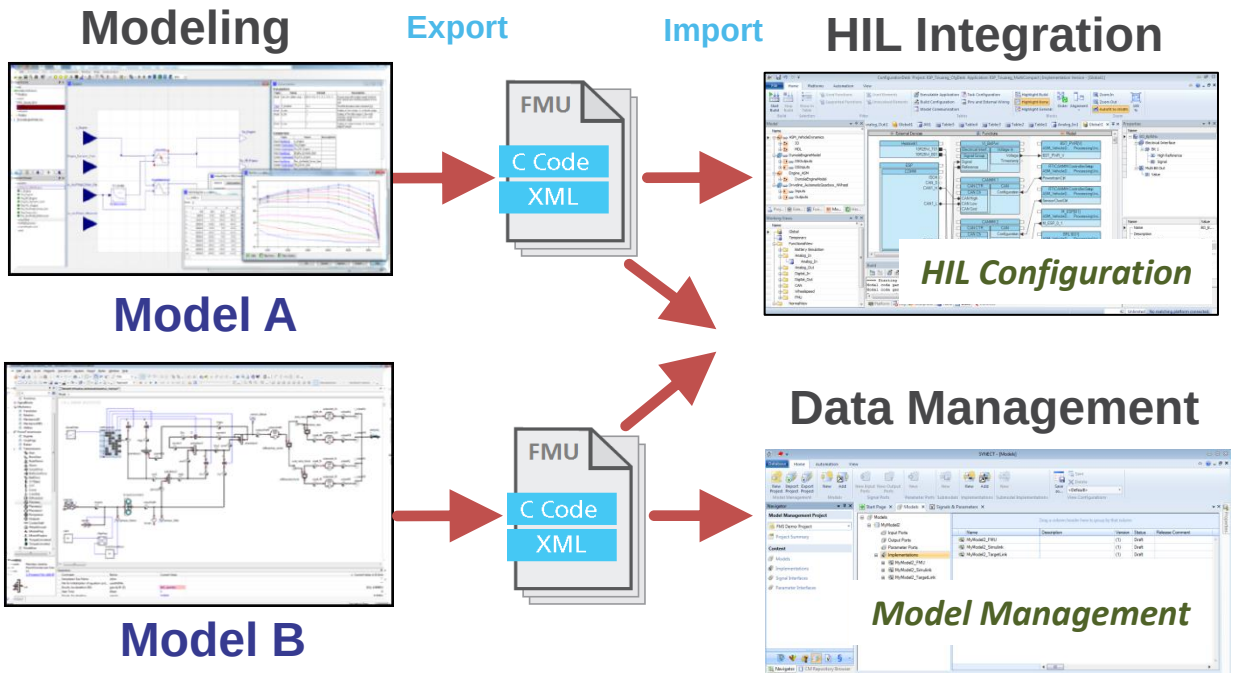
```
HTTP/1.1 200 Ok
Content-Type: application/x-ssp-parameter-set
Content-Length: …
ETag: "3f80f-1b6-3e1cb03b"

<?xml version="1.0" encoding="UTF-8"?>
<ssv:ParameterSet version="Draft20151124"
                  name="SystemParams" …>
    <ssv:Parameters>
        …
```

- Future extension:
  - Request version/ variant Descriptor for Resources
  - Query for alternative versions/ variants based on descriptor with wild-cards
  - Full parameter management API for editing, managing parameters and parameter Sets

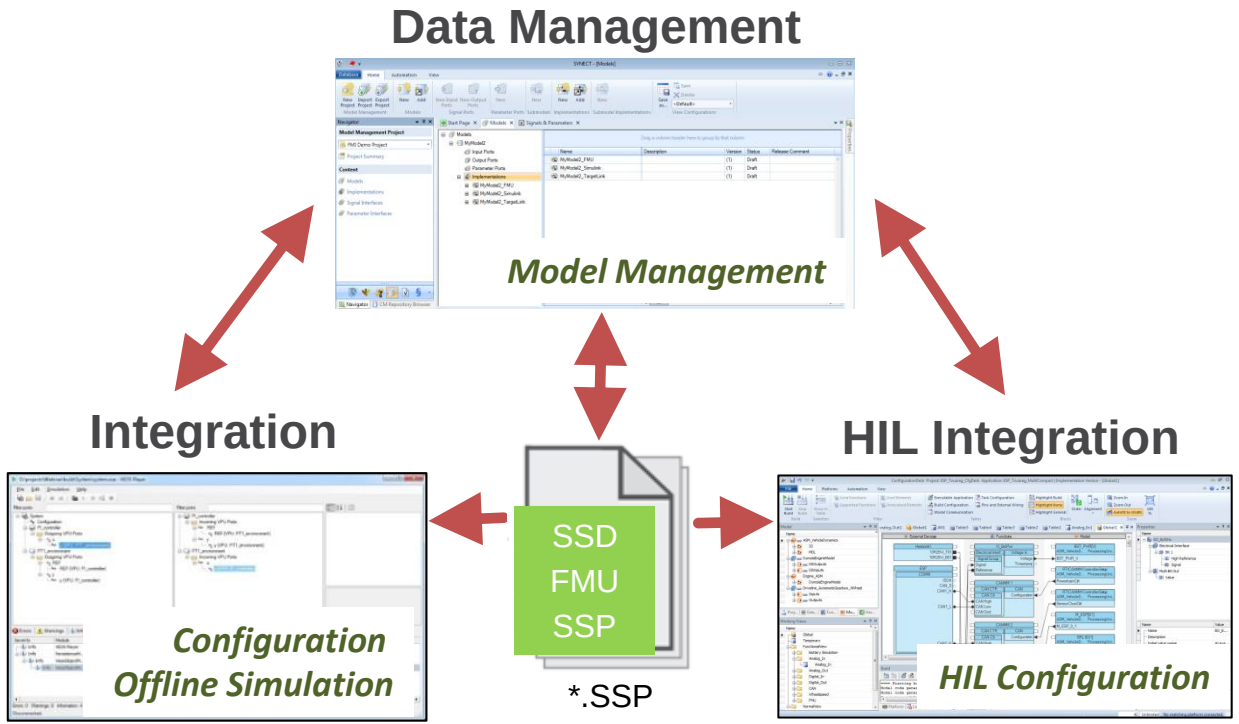
Integration of FMUs for HIL Testing

- HIL configuration tools are importing FMUs to integrate them with other FMUs, Simulink-based models and real ECUs
- Data Management tools are managing the lifecycle of the FMUs



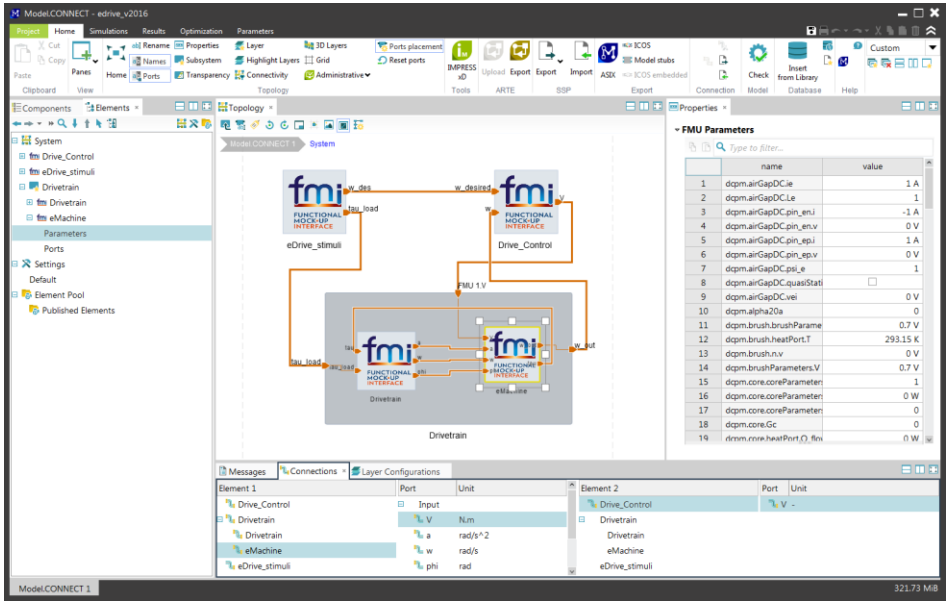
Reuse of the System Structure for SIL, MIL and HIL

- Integration and Data Management tools share a vendor independent system description (SSP)
- Reuse of tools, configurations, models, tests, layouts and parameters at system level is supported



Prototypes – Integration Tool

- Model.CONNECT™ by AVL – Scope:
  - Simulation architecture set-up
  - Model integration
    (FMI and dedicated interfaces)
  - Execution
    (office and lab)
  - Model management
  - Handling system structure and parameter variants

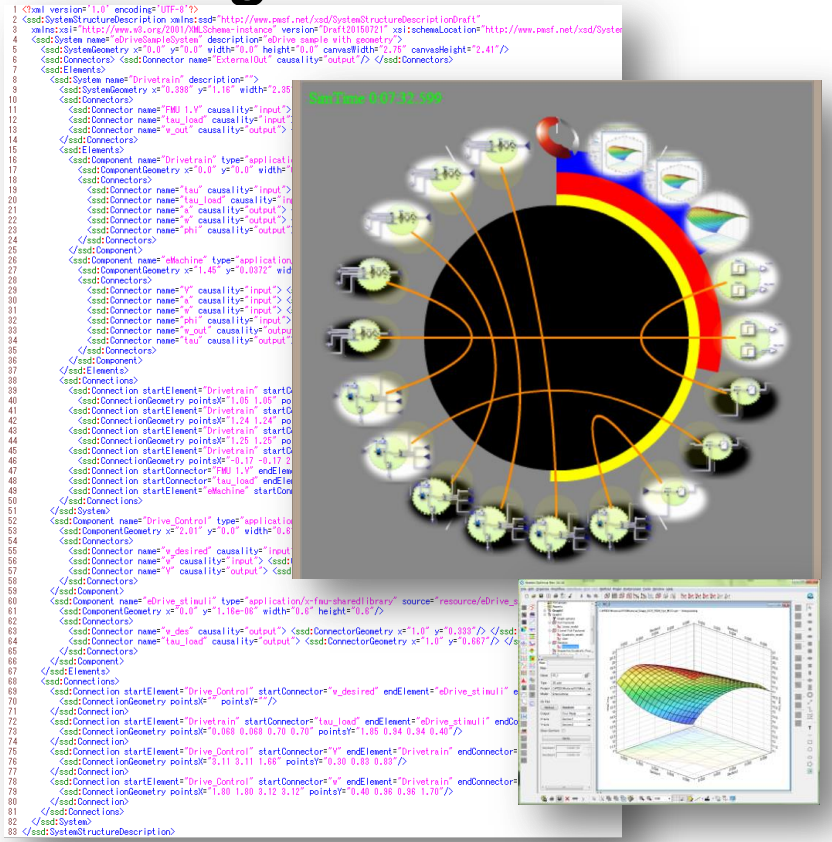


Prototypes – Integration Tool

- Model.CONNECT™ by AVL – SSP prototype:
  - Import and export of system structure (SSP packages)
    - Prototype supports multiple structure variants in the package
    - Mapping between the SSP variant handling and the tool-specific variant handling had to be implemented
    - Import-export roundtrip does not re-produce original ssp content
    - This is a consequence of the deliberately simple SSP variant handling concept
  - Import and export of graphical information
    - Overall layout information can be transferred via SSP. Intention is not to have pixel-by-pixel reproduction in any tool

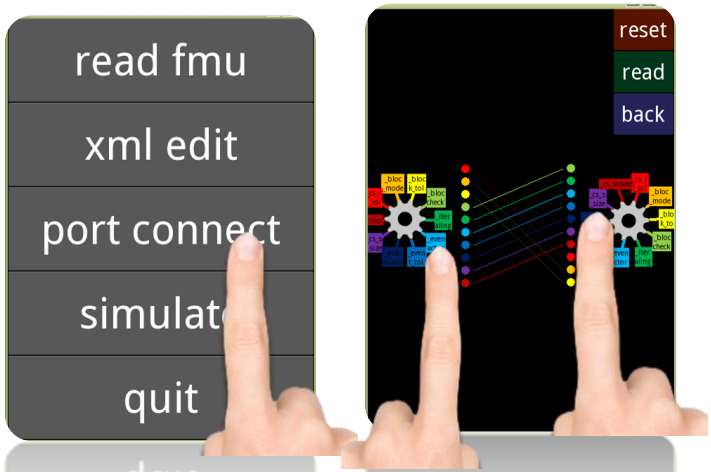
Prototypes – Online Testing Tool

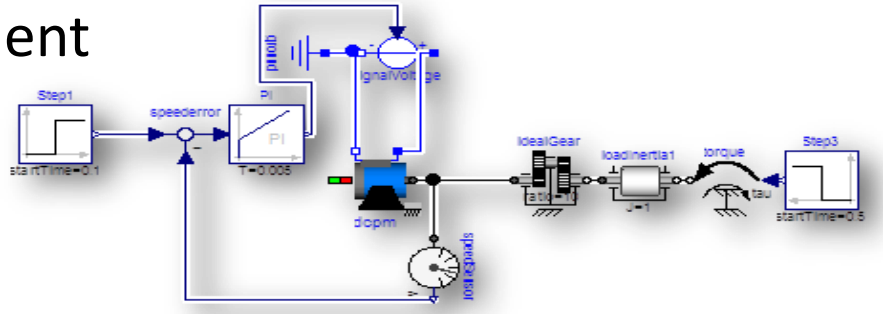
- Scalablility of <ssd:Connectors>
  - Ring configuration at a glance
  - 3D Flash UI for <ssd:Component>
- Time integration control master
  - Unit Test with default parameter
  - Synchronized Co-Simulation Test
- Parameter database as FMU
  - FMU of (sqlite.DB + sql.DLL)
  - exported by Optimus®

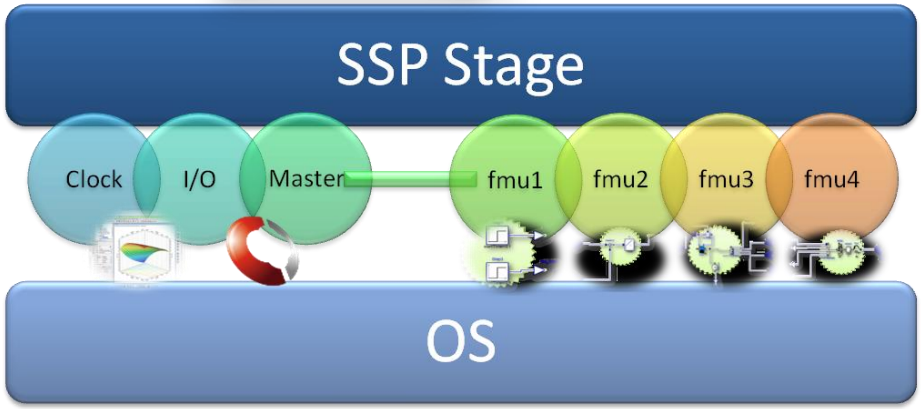


Prototypes – “Co-Simulation Browser” concept

- Mobile co-Simulation environment
- SSP(.zip) as online content
- Minimal GUI

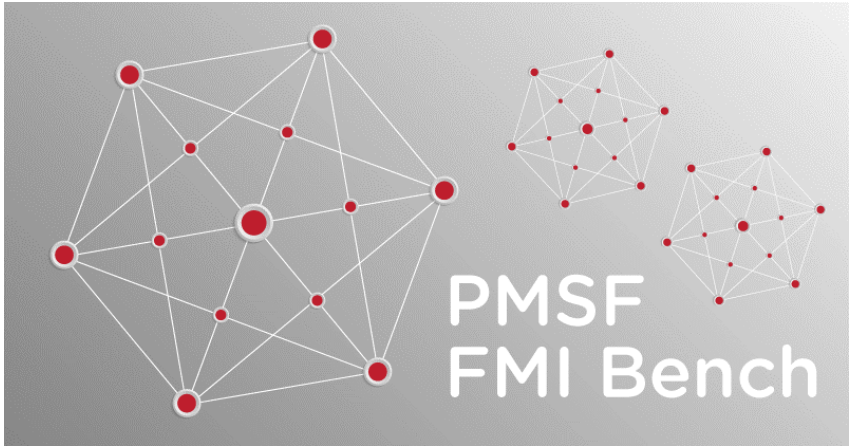






Prototypes: Integration Tool FMI Bench

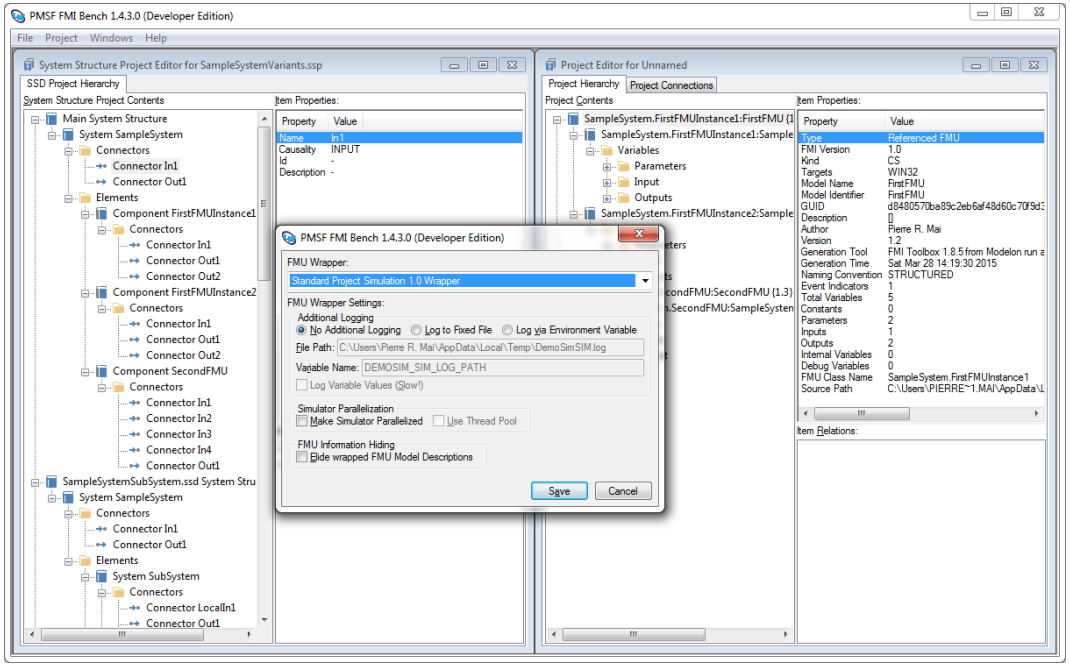
- FMI Bench by PMSF: Workbench for FMUs
  - FMU Inspector & Editor
  - FMU Profiling and Debugging
  - FMU Integration
  - Automated Workflows
  - Export FMU Networks as Integrated FMUs or Stand-alone Simulators



  - Supports Remote FMU Execution, FMU-internal Parallelization

Prototypes: Integration Tool FMI Bench

- FMI Bench SSP Prototype
  - Direct Editing of SSDs, SSPs, incl. Variants
  - Generation of Native FMI Bench Projects from SSP Projects
  - Generation of FMU or Stand-alone Sim. from SSP
  - Parallelization



Future work / Outline

- Further Development of API for parameter handling
- Try to involve providers of simulation data management systems in this project
- Evaluate approaches with „real-world examples“
- Publish first release soon

- Any contribution is very appreciated!