OpenScaling Deliverable 1.1 Requirements and scope of OpenSCALING solution

Version 1.01

Date : 2024-11-15

Requirements and scope of OpenSCALING solution

Description

This document describes the requirements and scope of the overall project, based on the analysis of different stakeholders such as OEMs, tier-1 suppliers, service providers and tool vendors, with respect to industrial application scenarios in the smart energy, smart building, aviation, and automotive domains.

The document starts with the *Scope* of the project, which is a collection of Demonstrator Analyses which have been conducted for all demonstrators. The demonstrators have also specified their requirements, and these requirements have been further discussed within various work groups where they have been compiled. The final set of requirements is presented in section *Requirements*. Primarily, the requirements are sorted with respect to the project's Innovations.

Scope

Following are the Demonstrator Analyses for each demonstrator.

Demonstrator Analysis: Large-scale green hydrogen production

General information

What is the demonstrator to be used for?

The demonstrator is used to model large-scale hydrogen production plants. These models are then used in a model-predictive control scheme.

What does the overview structure look like?

1. Model the large-scale hydrogen production plant in OpenModelica Connection Editor (OMEdit) with OpenSCALING solutions
2. Test different optimization solvers
3. Test that the technology and standards work
4. Compare results without OpenSCALING solutions

What is the goal of the demonstrator?

The demonstrator is aiming to showcase how the new developments from the research project allow to generate larger Modelica models still usable in optimization problems.

How do you measure the success of the demonstrator?

The demonstrator is evaluated by measuring

- the achieved simulation and optimization speed-up and
- the reduced memory requirements

in the target environment using the new features developed in WP3.

Solutions for which challenges are demonstrated?

The following challenges shall be addressed:

- How to generate large-scale models
- Modeling of media properties using PENodes

Technical information, relation to WPs

What information and data goes in and out and in what formats?

Inputs of "Large-scale green hydrogen production":

- Basic component models with parameters
- Simulation forecasts / data

Outputs of "Large-scale green hydrogen production":

- Large-scale green hydrogen energy management system demonstrator

Which tools and IT infrastructures are used?

OpenModelica Connection Editor (OMEdit) is used to create the model and export the FMU. The FMU is used in ABB OPTIMAX to run a model-predictive control algorithm. Machine runs on Microsoft Azure Cloud.

Which methods, delivery items from OpenSCALING are used and why?

Related deliverables:

- WP3: Large Scale System (LSS) Models
  - Treat large scale models with arrays of components. Such models don't work with current Modelica tools. Need to preserve arrays during model translation.
- WP4: Physically enhanced Neural ODEs (PeNODEs)
  - Model media properties with PeNODE, in particular when facing mixtures with impurities. Reduce complexity of model code, increase accuracy and reduce computational time.

Demonstrator Analysis: Virtual testing of heat pump system using surrogate models

General information

What is the demonstrator to be used for?

The demonstrator is used to significantly accelerate the simulation of a heat pump system model such that it can be used for control design, controller optimization, and virtual testing of the controlled system.

What does the overview structure look like?

Looking at the overall engineering process the demonstrator is focused on the step from *Plant Optimization* to *Control Design*. The plant model created from component models by a system simulation engineer is transferred to a control engineer. This plant model is used in the control design environment to test and optimize the controllers operating the heat pump system. The plant model is used in a defined operation range. The design variants are restricted to a defined set of design parameters.

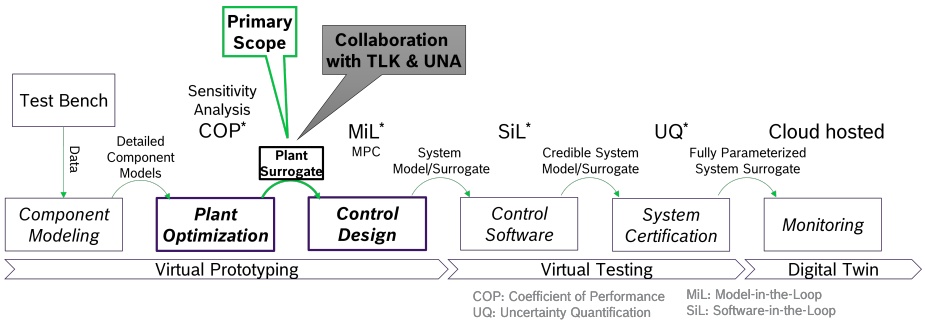

Figure 1: Engineering Process Chart

What is the goal of the demonstrator?

The demonstrator is aiming to showcase how a system simulation engineer is enabled by surrogate modeling techniques to generate fast running plant model from his detailed system model that meets the model quality requirements and can be used in the control engineering environment.

How do you measure the success of the demonstrator?

The demonstrator is evaluated by measuring the achieved simulation speed-up in the target environment comparing the surrogate against the state-of-art solution.

Solutions for which challenges are demonstrated?

The following challenges shall be addressed:

- Which method can be used to generate a surrogate for the complex dynamics of this system?
- How to make this method accessible to the simulation engineer such that he can define the constraints and requirements for the generated model without deeper knowledge about the training method itself?
- How can the model quality of the surrogate model be proven?
- How can the model be deployed in a way that the control engineer can easily integrate it into his control engineering environment?

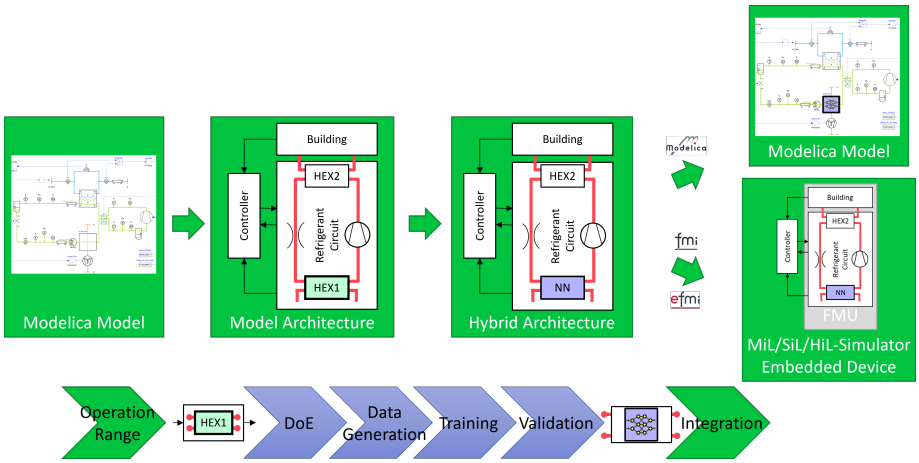

Figure 2: Workflow

Technical information, relation to WPs

What information and data goes in and out and in what formats?

Inputs of "Plant surrogate generation":

- Original model of the plant (Modelica model)
- Accuracy requirements (format to be defined, e.g., eFMI/Behavioral Model)
- Expected operation range (format to be defined)
- Expected parameter ranges (format to be defined)
- Training data (format to be defined)
- Validation data (format to be defined)
- Test data (format to be defined)

The original model is provided as Modelica model. The accuracy is defined by a error metric w.r.t. to validation and test data. The operation range is defined by value ranges or a reference trajectory for the inputs. The parameter ranges are defined by uncertainty attributes in hte Modelica parameter declarations.

The training/validation/test data are defined by DoE of the original model or reference solutions as time series data.

Outputs of "Plant surrogate generation":

- Surrogate model (Modelica model or FMU)
- Credibility (format to be defined)

What information, data is used internally?

Configuration data and meta parameters of the surrogate model generator.

Which tools and IT infrastructures are used?

Original model is created in Dymola. The surrogate generation method is implemented in Julia or Python. The control engineering environment is Matlab/Simulink.

Which methods, delivery items from OpenSCALING are used and why?

- WP2: uncertain parameter declaration in Modelica and FMI
- WP3: array preserving compilation
  - to avoid recompilation of heat exchanger models, when changing the discretization
- WP3: acausal FMU
- WP4: surrogate training method

Demonstrator Analysis: Pre-compiled flexible models for HVAC of buildings

General information

What is the demonstrator to be used for?

The demonstrator is used to significantly increase the reuse of pre-compiled acausal models in selecting/simulating components in, Air Handling Units (AHU). Also capabilities to allocate arrays of datastructures during runtime for FMUs will be addressed.

What does the overview structure look like?

Looking at the overall engineering process the demonstrator is focused on three steps:

1. Make simulations using acausal FMUs

- Test that the technology and standards work

2. Test different 'solvers' i.e. programs running FMUs
   - The HVAC model FMU created from component models by different tool providers are tested if they work together.
   - The design variants are restricted to a defined set of design parameters.
3. Different aspects of runtime allocatable arrays will be tested within the chosen HVAC model.

What is the goal of the demonstrator?

The demonstrator is aiming to showcase that it is possible to simulate a system with pre-compiled physical model, HVAC components, which have acausual ports and arrays that can change size. Also the improved speed of using pre-compiled models will be measured in comparision to using non-pre-compiled models.

How do you measure the success of the demonstrator?

Using FMUs:

1. The models can be used in an acausal manner, e.g. solving for the flow using certain input and also solving for the required input given the flow.
2. Demonstrate that dynamic memory allocation can be used for array data structures during runtime.
3. Using a mixture of different tool providers together.

The demonstrator is successful if computations using acausal FMUs (from preferably different FMU generators) are verified to yeild the 'same' result as non-FMU based AHU computations. Furthermore the dynamic array allocation must be working.

Solutions for which challenges are demonstrated?

The following challenges shall be addressed:

- Usage of acausal FMUs
- Allocatable arrays

Technical information, relation to WPs

What information and data goes in and out and in what formats?

Inputs of component models for AHU design:

- Original model of the components that make up an AHU (Modelica models)
- Accuracy requirements (format to be defined)
- Expected operation range (format to be defined)
- Expected parameter ranges (format to be defined)
- Validation data (format to be defined)
- Test data (format to be defined)

The original model is provided as Modelica model but not publicly available. The accuracy is defined by a error metric w.r.t. to validation and test data. The operation range is defined by value ranges or a reference trajectory for the inputs. The parameter ranges are defined by uncertainty attributes in the Modelica parameter declarations. The training/validation/test data are defined by DoE of the original model or reference solutions as time series data.

Outputs of FMU component models for an AHU:

- Component models (Modelica and FMU model)

What information, data is used internally?

Configuration data for the Modelica component models

Which tools and IT infrastructures are used?

Original model is created in Dymola/OpenModelica. Testing is done using Python

Which methods, delivery items from OpenSCALING are used and why?

- WP3: array preserving compilation
- WP3: acausal FMU

Demonstrator Analysis: Fuel Cell Propulsion Subsystems

General information

What is the demonstrator to be used for?

The demonstrator describes parts of a fuel cell driven aircraft propulsion system, namely the fuel cell and thermal management sub-systems. The model is used in the system development process for component sizing and integration, architecture optimization and control system design. It is simulated as an integrated part of the total system model directly in one simulation platform or combined with a control system or supplier sub-system as co-simulation FMU. In both cases the complexity of the component interactions, the required level of detail and some component's spatial discretization make the model execution too slow for an efficient data generation.

Within the OpenScaling project the demonstrator is used to benchmark the compilation and simulation performance of methods developed for large scale systems within the project such as array handling or the use of surrogate models.

In addition the benefits of improved model handling methods for graphical representation and meta-data transportation are planned to be analyzed in the context of this demonstrator.

What does the overview structure look like?

Looking at the overall engineering process the demonstrator is focused on two steps:

1. From *Component Modeling* to *Plant Optimization* The physical component models created from experimental data are used for plant optimization. The component models are accelerated or modeled more precisely. The plant model is used to design and optimize systems.
2. From *Plant Optimization* to *Control Design* The plant model created from component models by a system simulation engineer is transferred to a control engineer. This plant model is used in the control design environment to test and optimize the controllers operating the heat pump system. The plant model is used in a defined operation range. The design variants are restricted to a defined set of design parameters.
3. From *Control Design* to *Control Software* The plant model is also used for control software testing in a SiL/ HiL environment. Same defined operation ranges and restrictions as in 2. There may be a feedback loop to building the plant surrogate to extend the operation ranges.

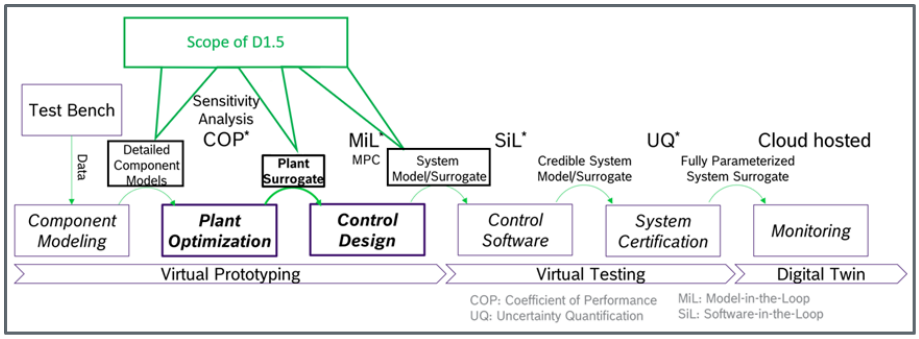

Figure 3: Engineering Process Chart

What is the goal of the demonstrator?

- Compilation and simulation performance improvement on sub-system level (faster simulation/compilation) without losing accuracy
- Model credibility and traceability data generation and usage in FMU delivery and integration workflows
- Deployment of the models on offline simulation platforms and realtime targets with the features included above

How do you measure the success of the demonstrator?

The demonstrator is evaluated by measuring the achieved simulation speed-up in the Modelica simulation tool and also in a SiL/ HiL environment comparing the surrogate (or a system model including a surrogate) against the state-of-art system models. Possible success factors are:

- Successful reduction in discretization in combination with ANN-based calculation of submodels
- Successful combinination of physics-based conservation laws with ML-based representation of physical phenomena
- Accuracy of a 50 cell model in a 5 cell model OR speed of a 50 cell model increased to today's 5 cell model speed
- Speed x10 for system optimization, controller design (HiL)
- Automized sub-system FMU delivery with standardized credibility information (no priority)

Solutions for which challenges are demonstrated?

The following challenges shall be addressed:

- Which method can be used to generate a surrogate for the complex dynamics of this system?
- What kind of Workflow is needed to build a surrogate system model or a system model including a PeN-ODE?
- Which physical model part shall be replaced by a surrogate?
- Shall physical time constants be replaced or complemented by PeN-ODEs?
- How to generate training data that includes necessary dynamics?
- Do PeN-ODEs work with with discontinuous/ discrete switches?
- How to make this method accessible to the simulation engineer such that he can define the constraints and requirements for the generated model without deeper knowledge about the training method itself?
- How can the quality of the surrogate model be proven?
- How can ANN model parts be reintegrated into Modelica without explicit Modelica code?
- Which techniques can tool vendors use to speed up compiling and simulation for discretized component models?
- How do libraries for fuel cell systems need to be changed to make use of these new techniques?

Technical information, relation to WPs

What information and data goes in and out and in what formats?

Inputs of "Plant surrogate generation":

- Original model of the plant (Modelica model)
- Accuracy requirements (format to be defined)

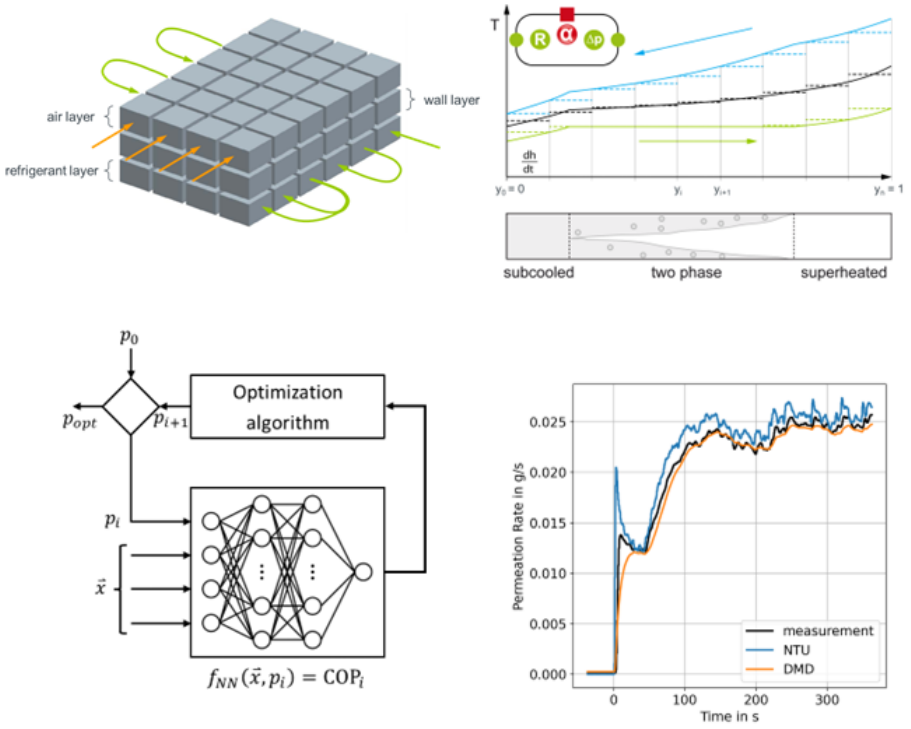

Figure 4: State Of The Art

- Expected operation range (format to be defined)
- Expected parameter ranges (format to be defined)
- Training data (format to be defined)
- Validation data (format to be defined)
- Test data (format to be defined)

The original model is provided as Modelica model but not publicly available. The accuracy is defined by a error metric w.r.t. to validation and test data. The operation range is defined by value ranges or a reference trajectory for the inputs. The parameter ranges could be defined in the Modelica parameter declarations. The training/validation/test data are defined by DoE of the original model or reference solutions as time series data.

Outputs of "Plant surrogate generation":

- Surrogate component models (Modelica model with ext. functions or similar)
- Surrogate plant models (Modelica model with ext. functions or similar or FMU)

What information, data is used internally?

Configuration data and meta parameters of the surrogate model generator.

Which tools and IT infrastructures are used?

Original model is created in Dymola with library TIL. The surrogate generation method is implemented in Python (PyTorch, skikit-learn, tensorflow, etc.). MoBA Automation workflow for surrogate generation and validation. Jenkins and Gitlab CI pipeline for surrogate/ FMU generation and validation

Which methods, delivery items from OpenSCALING are used and why?

- WP3: Array preserving compilation (Innovation 1)
- WP3: Large scale system Modelica model graphical representation (Innovation 1)
- WP4: Surrogate training methods, surrogate integration format (Innovations 4 + 5)
- WP5: Meta-data representation in FMUs (Innovation 3?)

Demonstrator Analysis: Demonstration of Credibility Format in the Aerospace and Automotive Domains

Introduction

In this demonstrator the end users (Saab, VTI, Volvo, and Parker) aims to demonstrate the value of connecting a joint format (deduced in WP2 and WP5) of credibility information to the end use of simulator applications in their respective domains. The contributing tool developers will facilitate the enabling technology to capture, distribute, and communicate, and analyse the aggregated credibility information in order to facilitate decision support in terms of: modelling activities using the latest features of FMI and SSP, model Verification and Validation (V&V) and Uncertainty Quantification (UQ), and model based decision making.

To ensure generalization of methods and tools, the vehicle systems demonstrator is instantiated in two examples; aerospace vehicle systems and automotive vehicle systems respectively. The examples share common technology and methodology.

Stakeholder needs

Aircraft energy management As an OEM Saab want to develop and mature enablers for credbile (including traceability aspects) round trip engineering in order to realize energy managment strategies of aircraft vehicle systems.

Alternative scenarios:

- Evaluate Environmental Control System (ECS) software control strategies with respect to maximizing of available cooling power. Models needed that surround the unit under test are identified and developed in the best suited tool. A suitable scale simulator is developed and deployed in an environment enabling large scale system simulation and rigourous design space exploration of ECS controlling software solutions.
- Evaluate Fuel System (hardware and software architectures) strategies with respect to efficient managment of heat energy in order to maintain feasible fuel temperatures. Focus on early life-cycel stages (conceptual design of fuel system architectures).

Electric drive train design As an OEM, Volvo wants to utilize VTI simulators for energy efficiency and driveability assessment of new electric drive train design alternatives. This use-case covers both single stakeholder case where the OEM is responsible for the integration, and multi-stakeholder case where the integration responisbility is shared, allowing multiple aspects and system level perspectives to be investigated.

Alternative scenarios:

- Volvo provides a complete vehicle model as black-box FMU(s), for integration in VTI simulators.
- Volvo provides a drive train model as black-box FMU(s), for integration with VTI vehicle models in VTI simulators.

- Part of vehicle design includes Parker component(s) such as control valves for rear-axle steering.
- Volvo has a drive train Hardware-In-the-Loop (HIL) test rig that is connected to a VTI simulator using Distributed Co-simulation Protocol (DCP).

Automated driving As an OEM Volvo wants to utilize VTI simulators for virtual verification of Automated Driving (AD) system functionality in complex traffic environments.

Alternative scenarios:

- Volvo provides a complete vehicle model including AD software as black-box FMU(s), for integration in VTI simulators.
- Volvo provides AD software as black-box FMU(s), for integration with VTI vehicle models in VTI simulators.

Credibility assessment for increased quality in model-based decisions As an end user of a simulator application at Saab, Volvo, or VTI, I want to be informed of simulation credibility, to the extent possible based on objective data. This requires accounting for several aspects in building credibile simulations such as validity and traceability.

Alternative scenarios:

- I want to be able to easily express the intended use(s) of the simulation in a standardized format.
- Prior to the simulation I want to be able to check suitability of available simulation models w.r.t. the inted use(s), focusing on standardized static measures.
- During the simulation I want to monitor the overall credibility of the simulation using a standardized format.
- Post to the simulation I want to be able to access an overview of assessed credibility in some suitable format.

Provision and packaging of credibility information As a model responsible at Parker, Saab, Volvo, or VTI, I want to provide models with information on model credibility to customers (end users).

Alternative scenarios:

- I want to provide standardised information of model credibility as metadata in our exported simulation models. Models can represent any system level, from component to system of systems.
- I want to get feedback, after simulation, on the performance of my model in terms of credibility and suitability, and its impact on the complete simulation. Note that this need requires communication between end users and model responisble.

- I want to re-use the same valve component model in the different use cases of the demonstrator.

Overarching Use-Case

Description and Value The overarching use-case aims to demonstrate, in a collaborative setting, the benefit of communicating model meta-data on a standardized format along with corresponding tool support. This use case will demonstrate OpenScaling technology developed in related OpenScaling work-packages, particularly WP2 and WP5. This overarching use-case is tailored and instantiated in specialized aerospace and automotive applications to demonstrate OpenScaling solutions.

The use-case "Main Scenario" is visualized scehmatically in Figure XX. Please note that the scenario captures a sub-set of the artifacts relevant in the aerospaceace and automotive specializations of the overarching use-case.

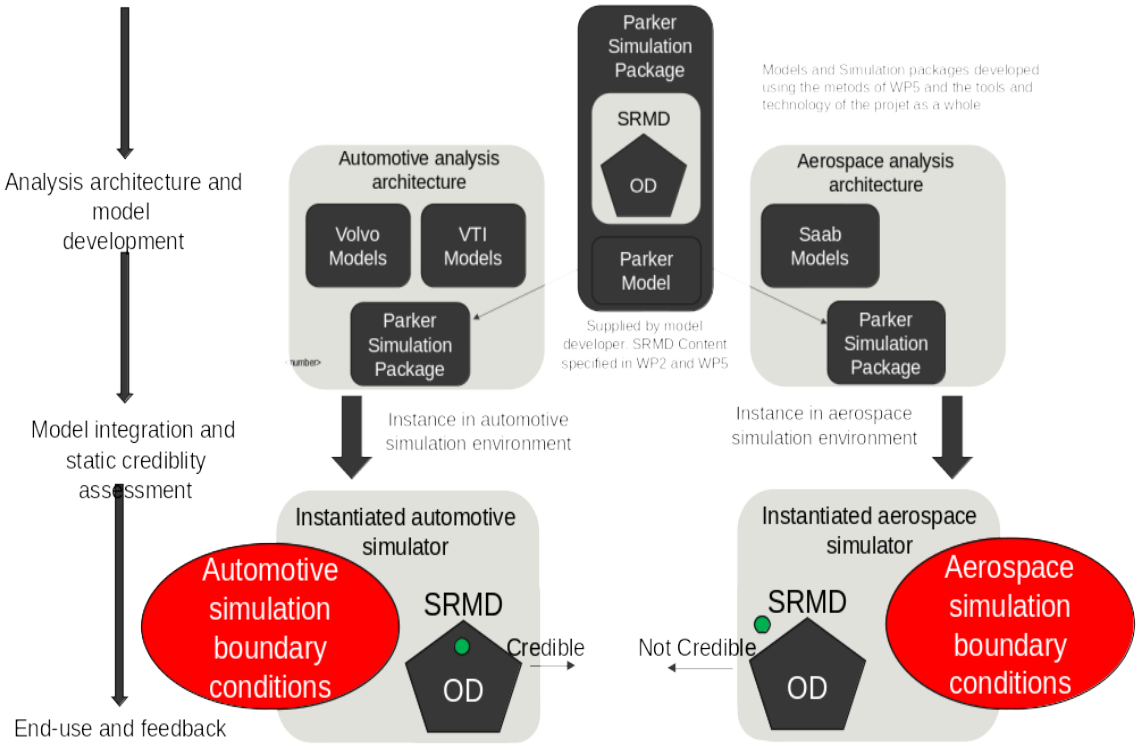

Figure 5: Schematic description of overarching use-case

The Parker Simulation Model is to be used in both the aerospace and automotive

specalised use cases, where the aim is to use the same valve model in both cases, with additional layers modelled as use-case specific FMUs, as illustrated in the figure below.

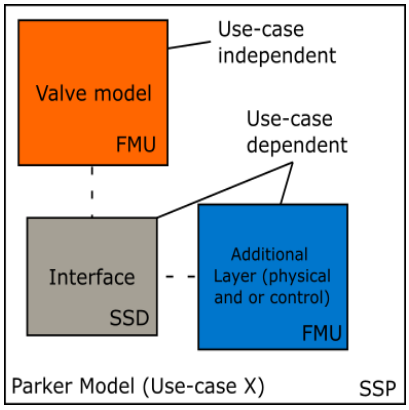

Figure 6: Schematic description of the Parker valve model used in the use-cases

Prerequisites An end-user has a need to analyse and evaluate alternative system realizations w.r.t. product requirements. Support of FMI, SSP, and SRMD in all particpating export and integration tools. Tool functinality and development workflows supporting, and packaging, information related to credibility, for example: means to asess uncertainty, validity, and traceability.

Actors System Architect Analysis Architect Model Responsible Simulation Engineer Model Integrator

Note that one individual engineer may take the role of several actors.

Main Scenario

1) The System Architect and Analysis Architect specifies the overall system architechture and corresponding analysis architecture(s) capturing the unit under test. There may be several different configurations of both system and analysis architectures depending on analysis needs and product requirements.

2) The Analysis Architect extracts and communicates, using the xml formats of the FMI and SSP standards, interfaces of the simulation models that are to be supplied by the Model Responsible of each analysis architecture constituent part. Note that this includes meta-data such as early estimates of intended use that can be lated refined by the Model Responsible.

3) The Model Responsible generates a simulation model according to the specification from the Analysis Architect, including meta-data expressed in

e.g. the SRMD format. This model is delivered, once ready, to the Model Integrator.

4) The Model Integrators investigate the meta-data, prior to integration, to asses the static credibility of the model. If applicable, the model is then integrated in the analysis architecture. If not, then this feedback is provided to the Model Responsible. The resulting executable analysis architecture (parameteraized realization model) is passed on to the Simulation Engineer.

5) The Simulation Engineers execute the virtual test which may be setup as e.g. software in-the-loop, pilot/driver-in-the-loop, and/or hardware-in-the-Loop. The dynamic credibiliy, based on the meta-data, is assessed on-line during simulation or post to simulation. The results of the simulation are packaged and communicated to end-users evaluating product requirements.

Post Conditions Available simulation results tagged with information specifying objectivly deduced credibility of the integrated models; an assessment that may vary from one simulation time instant to the other.

Specialized Use-Case 1: Aerospace Vehicle Systems

Description and Value This is a tailored and instantiated version of the overarching use-case, specialized towards aerospace vehicle systems, demonstrating how to adress stakeholder needs on "Aircraft energy management".

Prerequisites Support of FMI, SSP, and SRMD in all particpating export and integration tools. Tool functinality and development workflows supporting, and packaging, information related to credibility, for example: means to asess uncertainty, validity, and traceability.

Actors System Architect (Saab) Analysis Architect (Saab) Model Responsible (Saab, Parker) Simulation Engineer (Saab) Model Integrator (Saab)

Main Scenario

1) The System Architect and Analysis Architect specifies the overall system architechture and corresponding analysis architecture(s) capturing the unit under test. There may be several different configurations of both system and analysis architectures depending on analysis needs and product requirements. The following configuration of analysis architecture is considered:

   - Environmental Control System (legacy model, Modelica)
     - Parameterized to support two different geometry concepts via two different SSV files supplied from corresponding geometry models.
   - Environmental Control System Software (legacy model, Simulink/Modelica)
   - Consumer of cooling power (legacy model, Modelica)

- Ambient conditions (legacy model, Modelica)
- Fuel System (New, Modelica-ThermoFluidStream)
  - Parameter values extracted from geometry model, integrated as SSV files
- Fuel System Software (New, Tool/Language-TBD)
- Fuel system Heat-sink Shunt (New, Parker tooling)

The models marked as "new" in the configuration description are new in the sense that they have not been integrated in this context in previous work. That being said, models may very well be available that have been used for other applications.

2) The Analysis Architect extracts and communicates, using the xml formats of the FMI and SSP standards, interfaces of the simulation models that are to be supplied by the Model Responsible of each analysis architecture constituent part, see Figure XX for an overview of the aerospace application example analysis architecture.

Note that this includes meta-data such as early estimates of intended use that can be lated refined by the Model Responsible. In the following step, Parker will provide a model that describes the control valve and a software interface layer for the fuel system heat sink shunt. For the Parker model, the following meta-data is of relevance for the final integration and they thus need to be incrementally considered already in this second step of the use-case:

- Intended use
- Uncertainty
- Operational domain

This type of meta-data is relevant for all of the models of the application example and all model responsibles need to quantify the required meta-data for their respective models.

3) The Model Responsible generates a simulation model according to the specification from the Analysis Architect, including meta-data expressed in e.g. the SRMD format. This simulation model generation includes the development and integration of any supporting software layer adapting available valve model interfaces to the fuel system heat sink shunt application.

The supporting software layer will be modelled as a System Structure Description (SSD) file in combination with supporting FMUs all packaged along with the valve FMU in an SSP. This, to maximise reuse and flexibility of the valve FMU. This model is delivered, once ready, to the Model Integrator.

4) The Model Integrators investigate the meta-data of the fuel valve model, prior to integration, to asses the static credibility of the model. If applicable, with respect to the static meata-data of the neighbouring model components, the valve model is then integrated in the analysis architecture. If the valve model is not applicatble, this feedback is provided to the

Model Responsible. The resulting executable analysis architecture (parameteraized realization model) is passed on to the Simulation Engineer. In paralell and iterativly, the fuel system model (and corresponding software) is subjected to the same static integration tasks as this model is novel to this particular application.

5) The Simulation Engineers execute the virtual test in which aircraft energy management is evaluated. The dynamic credibiliy, based on the meta-data, is assessed on-line during simulation or post to simulation. The results of the simulation are packaged and communicated to end-users evaluating and developing energy management strategies.

Post Conditions Available simulation results tagged with information specifying objectivly deduced credibility of the integrated models; an assessment that may vary from one simulation time instant to the other.

Specialized Use-Case 2: Automotive Vehicle Systems

Description and Value This is a tailored and instantiated version of the overarching use-case, specialized towards automotive vehicle systems, demonstrating how to adress multi-stakeholder needs on drive train energy efficiency and driveability, and credibility assessment. The stakeholder need concerning "Electric drive train design" is considered.

Prerequisites Support of FMI, SSP, and SRMD in all particpating export and integration tools. Tool functinality and development workflows supporting, and packaging, information related to credibility, for example: means to asess uncertainty, validity, and traceability.

Actors System Architect (Volvo) Analysis Architect (Volvo, VTI) Model Responsible (Volvo, VTI, Parker) Simulation Engineer (VTI) Model Integrator (Volvo, VTI)

Main Scenario

1) The System Architect and Analysis Architect specifies the overall system architechture and corresponding analysis architecture(s) capturing the unit under test. There may be several different configurations of both system and analysis architectures depending on analysis needs and product requirements. The following configuration is planned initially:
    - Complete Model (what is meant by this, is the the complete executable analysis architechture?)
    - Driver model/user interaction (VTI)
    - Road and scenario models, preferrably developed using standards OpenDrive and OpenScenario (VTI)

- Vehicle model:
  - Iteration no. 1
    - Complete vehicle model by Volvo (Volvo)
    - Integration in VTI simulator (VTI)
  - Iteration no. 2
    - Valve model for rear axle steering (Parker)
    - Complete vehicle model by Volvo with integrated rear axle steering model (Volvo, Parker)
    - Integration in VTI simulator (VTI)
  - Iteration no. 3
    - Valve model for rear axle steering (Parker)
    - Partial vehicle model from Volvo with integrated rear axle steering model (Volvo, Parker)
    - Integration with parial vehicle model from VTI in VTI simulator (VTI, Volvo)

2) The Analysis Architect extracts and communicates, using the xml formats of the FMI and SSP standards, interfaces of the simulation models that are to be supplied by the Model Responsible of each analysis architecture constituent part. Note that this includes meta-data such as early estimates of intended use that can be refined later by the Model Responsible. Parker provides a model that describes the control valve and mechanical control linkage for the rear-wheel steering. For the Parker model, the following meta-data is of relevance:

   - Intended use
   - Uncertainty
   - Operational domain

3) The Model Responsible generates a simulation model according to the specification from the Analysis Architect, including meta-data expressed in e.g. the SRMD format. This process includes the development of any physical and/or control layer needed between the valve model and the steering wheel/vehicle control system. This model is delivered, once ready, to the Model Integrator.

4) The Model Integrators investigate the meta-data of the valve steering model, prior to integration, to asses the static credibility of the model. If applicable, with respect to the neighbouring model components, the valve model is then integrated in the analysis architecture. If the valve model is not applicable, then this feedback is provided to the Model Responsible. The resulting executable analysis architecture (parameteraized realization model) is passed on to the Simulation Engineer.

5) The Simulation Engineers execute the virtual test which may be setup as e.g. software in-the-loop, pilot/driver-in-the-loop, and/or hardware-in-the-Loop. The dynamic credibiliy ,based on the meta-data, is assessed on-line during simulation or post to simulation. The results of the simulation are

packaged and communicated to end-users evaluating product requirements. For the Parker model, the results of primary interest is the valve model's contribution to the complete model's:

- Uncertainty
- Intended use (validity?)
- Operational domain

Post Conditions Available simulation results tagged with information specifying objectivly deduced credibility of the integrated models; an assessment that may vary from one simulation time instant to the other.

Post Conditions Available results of the vehicle simulation tagged with information specifying objectivly deduced credibility of the integrated models; an assessment that may vary from one simulation time instant to the other.

General information

What is the demonstrator to be used for?

A number of different use-cases are associated with the demonstrator. The overarching, and primary, goal of the demonstrator is to assess the value of incorporating standardized meta-data into large scale system simulator models and in general to investigate relevant aspects of model integration and collaboration. The demonstrator aims to strenghten this assessment by means of two different applications, one in the automotive domain and one in the aerospace domain. In summary, the demonstrator has

Two Applications

1) Aerospace

   One Aerospace Configuration

   1) Standard configuration

   One Aerospace Simulation purpose

   1) Assessment of energy consumption

2) Automotive

   Three Automotive Configurations

   1) Truck with Volvo drivetrain model
   2) Truck with VTI drivetrain model
   3) Black box truck

   Two Automotive Simulation purposes

   1) Assessment of energy consumption
   2) Assessment of vehicle behavior in traffic

Two Common purposes

1) Demonstrate feasibility and performance of common simulation model
2) Estimate system simulation credibility based on component credibility

What does the overview structure look like?

The demonstrator is partitioned into two different applications, one automotive application and one aerospace application. The structure of each application is provided in Figure 1 and 2 together with the information provided in the two specialized use-cases.

Aerospace Vehicle Systems example The architecture of the aerospace application example is modeled as SysML v1 block definition diagrams. The top level view is provided in Figure 1 which describes the application example structure.

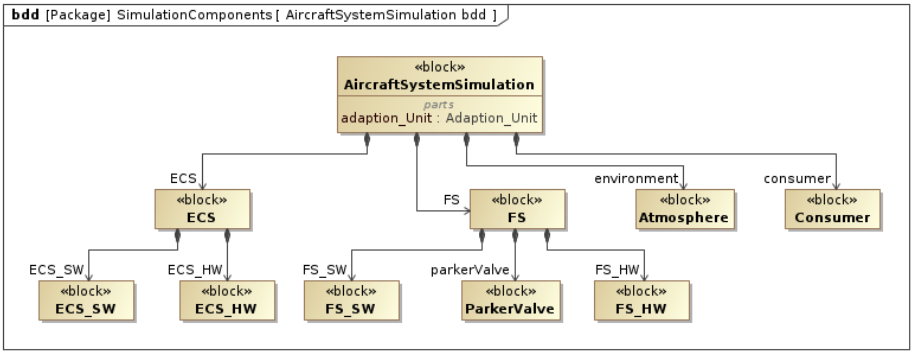

Figure 7: Aerospace application example architecture, top level

The Architecture includes two aircraft general vehcile systems: the Environmental Control System (ECS) and a Fuel system (FS). Both of these subsystem contains modeled hardware and software. Furthermore, the architecture description also includes external entities providing input two, and depending on the output of, the general vechicle systems: an atmosphere and a consumer of cooling power.

Automotive Vehicle Systems example A multi-stakeholder case is demonstrated, integrating models from several stakeholders with differens needs and interests at differens system levels. The demonstrator will showcase simulation of multiple simulation components integrated at system levels in a common simulator platform using open standards. The preliminary architecture of the automotive application is seen below.

The architecture and main features are outlined as follows:

- VTI vehicle model as FMU

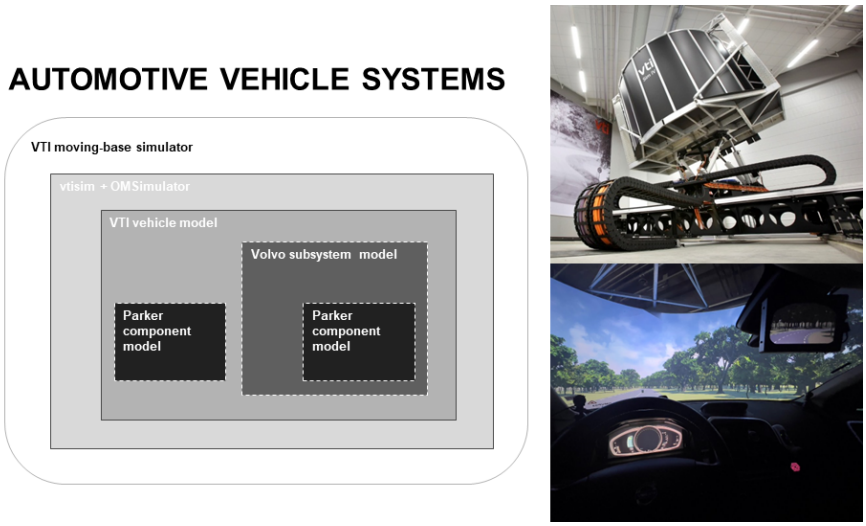

Figure 8: Automotive application example architecture, top level

- Subsystem model replaced by FMU from Volvo, e.g. drive train, combustion engine, electric motor, battery, tyre, etc.
- Component model from Parker integrated as FMU as part of Volvo and/or VTI models, e.g. a control valve as part or the drive train or rear axle steering.
- Possibly individual Volvo FMUs can be PeN-ODE
- Online simulation credibility info using SSP extensions
- Use OMSimulator for import, connection, and efficient simulation

What is the goal of the demonstrator?

The goal of the demonstrator is to showcase the use of providing model meta data, such as credibility information, on a standardised format in submodels that are integrated into larger models. More specifically, information on e.g. model credibility of the larger model is aggregated from its submodels, which will help the user judging the credibility of the simulation results and identify any submodels that poorly match the intended use(s). This is further detailed in the use-cases.

How do you measure the success of the demonstrator?

- How much manual intervention is required on data/model exchange between stakeholders?
- To what degree can credibility of the vehicle simulation be derived from annotations of the component simulation models.
- To what degree can quantified uncertainty of the vehicle simulation be de-

rived from quantified uncertainty of the the component simulation models.
- What number of identified standardization mechanisms can be applied as is in industry.

Solutions for which challenges are demonstrated?

The demonstrator will target the following OpenSCALING research and standardization topics:

- Simulate large scale systems: Demonstrators in large scale simulators, traffic simulations with large number of instances, and ways of working in large heterogenuous organizations.
- Model quality assessment/traceability: Standardized methods for capturing and communicating static and run-time model credibility, including traceability aspects, using MA standards extensions, for example SSP and Simulation Resource Meta-Data (SRMD).
- Harmonize MA standards: Meta data and uncertainty information, contribute to standards enhancements.
- Collaborative Systems Engineering Credible Simulation Process: SSP traceability, SysML interoperability, SysML and/or AUTOSAR/EAST-ADL architectures, standardized model exchange and (co-)simulation using FMI, SSP, and potentially DCP.

Technical information, relation to WPs

- WP2 results on uncertainty quantification and credibility will be applied and assessed
- WP4 results on PeN-ODE technology in general and PeN-ODE FMUs in particular will be applied and assessed
- WP5 workflow will be applied and assessed For each technology application, feedback will be provided to WPs.

What information and data goes in and out and in what formats?

Tool functionality, and knowledge concerning tool capabilities, that support the workflows specified in WP5 serve as demonstrator inputs. The deomstrator will supply examples, motivation of needs, benchmarks, and requirements to the related project workpacakges.

Inputs of "Vehicle system demonstrator":

- Simulation packages developed by appling the workflow of WP5. The packages include models and information describing the models credibility
- Standardized meta-data format originating from WP2.
- Large Scale Modelling technology for fuel system modelling enabled by the ThermoFluidStream Modelica library

Outputs of "Vehicle system demonstrator":

- Simulation results answering questions according to the stakeholder needs.
- Internal Project feedback on applied technologies

What information, data is used internally?

- Definitions of Simulation architecture and Simulation components, including required annotations

Which tools and IT infrastructures are used?

- OpenModelica
- EasySSP
- EATOP

Which methods, delivery items from OpenSCALING are used and why?

The demonstrator contributes to validating the following OpenSCALING solutions and challenges:

- Tools & methods for large scale system simulation.
- Efficient model quality assessment and UQ for large and/or high dimensional systems.
- Meta data enriched model exchange standards.
- Traceable model credibility as integral part of model-based systems engineering
- (Built-in simulation AI support.)
- (From manual modeling to model generation.)

Demonstrator Analysis: PeN-ODEs for Autonomous Vehicles

General information

What is the demonstrator to be used for?

This demonstrator illustrates the workflow of training, deploying, and adapting PeN-ODE based control. The process begins with the training of a PeN-ODE in the Julia programming environment. The trained model is then transferred to a simulation platform such as Dymola or Simulink and embedded into the control structure. Finally, it is exported as an eFMU and deployed on a real-time target. A key goal of this demonstrator is to enable the update of trainable components, i.e. the reparametrization during runtime, allowing for real-time

adaptation to changing conditions or the rapid test of multiple variants in a time critical situation.

Outline

First, an appropriate PeN-ODE based algorithm is trained in Julia. In this phase, an exchange with the methodological developments of WP 4 is crutial. On the one hand, available solution can be incorporated in the training and on the other hand, experiences are shared and might influence the results of WP4 (e.g. in D4.3 Best-Practices for PeNODE modelling). After training, the trained components must be transferred to the simulation tool and be embedded in the overarching control structure of the experimental platform. To this end, solutions of DLR-SR and the MILEA-Library from Bosch are compared and possible extensions to support standards like ONNX are evaluated. Then, the eFMI-export is used to deploy the algorithm on an real-time target. Finally, an update of the trainable components during runtime should be performed, using the enhancements of the eFMI standard that will be developed in OpenSCALING. To do so, we rely on the expertise of Dassault Systèmes that will work on the standard enhancements for FMI and eFMI.

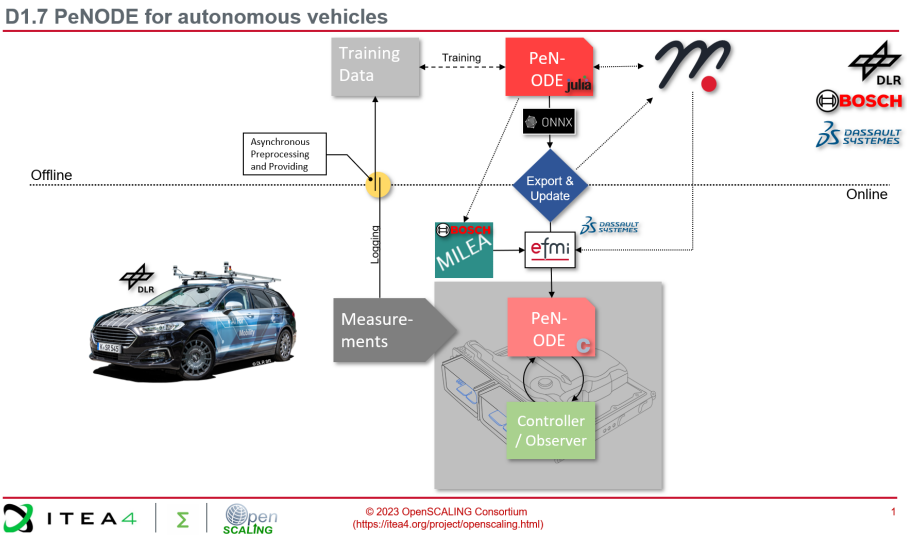

Figure 9: Demonstrator overview

What is the goal of the demonstrator?

This demonstrator showcases the entire workflow to implement a PeN-ODE based algorithm on a real time target enables the online adaption of trained components using eFMI.

How do you measure the success of the demonstrator?

The crucial parts that define the success of this demonstrator are:

1) The successful training of a PeNODE in Julia. The convergence and improvement of the training will be show using the training-, test- and validation error that is common practice in machine learning.
2) The successful integration of the PeNODE within the overarching control system of the experimental platform. The success is determined by validation of the PeNODE outputs with suitable benchmark data generated in the training environment.
3) The deployment of the control system with the integrated PeNODE on a real time target using eFMI. The benchmark from step 2) should then be repeated on the target.
4) The update of the PeNODE parameters without recompilation. The updated PeNODE will again be benchmarked with appropriate benchmark data.

Solutions for which challenges are demonstrated?

The following challenges are addressed:

1) Training of a PeN-ODE in Julia
2) Transfer of trained components from the training- to the simulation environment
3) Export of a PeN-ODE based algorithm as eFMU and deployment on an embedded target
4) Update of trainable parameters during runtime

Technical information, relation to WPs

What information and data goes in and out and in what formats?

Inputs of "PeN-ODEs for autonomous Vehicles":

- Basic control algorithm concept (algorithm)
- Prediction model of the vehicles vertical dynamics (ODE)
- Training Data (time series measurements)

Outputs of "PeN-ODEs for autonomous Vehicles":

- PeN-ODE based control algorithm in simulation tool (simulation model)
- PeN-ODE based control algorithm as eFMU on embedded target (compiled software)

What information, data is used internally?

- Representation of learned components (e.g. via ONNX)
- Export / import mechanism for learned components

- Logging and preprocessing and providing of training data

Which tools and IT infrastructures are used?

- **Julia** is used to train the PeN-ODE. To perform the training, HPC of DLR can be used.
- **Dymola** and/or **Simulink** are the targeted simulation environments.
- ONNX is evaluated as an open standard for the model exchange of trainable components
- eFMI is the enabler to bring the PeNODE to the embedded system
- An Rapid Prototyping embedded system on the research platform AFM is the target for deployment.

Which methods, delivery items from OpenSCALING are used and why?

- WP4: Representation of PeN-ODEs in modelling standards
- WP3: Acausal FMI component models
  - "[...] it is possible to provide runtime adaptable digital twins, because no recompilation is needed in order to change causality or array sizes."

Demonstrator Analysis: Hybrid modelling and part-specific performance prediction of advanced steering systems

General information

This demonstrator will be removed from the project and is planned to be replaced by another. Since this process is not finished (Nov. 14th, 2024), and since the requirements which stem from this demonstrator are still considered for the planned innovations, the demonstrator analysis is kept in this document for reference.

What is the demonstrator to be used for?

The demonstrator is used to develop a workflow to combine first principles models of a steer by wire actuator with data from a test rig to a hybrid model (PeN-ODE). This is done to achieve a higher level of detail when simulating the actuator behaviour. This resulting PeN-ODE shall allow to further reduce the number of prototypes and hardware tests.

Furthermore, data-driven probabilistic modeling approches for parameter and input identification are evaluated, which allow for part-specific fingerprinting and reject-cause detection of steering system during prodution.

What does the overview structure look like?

The process of creating and training PeN-ODEs starts with a first principle model. A steer by wire actuator is created as a mechatronic system model in Matlab Simulink. Therefore it contains mechanical as well as electrical and system control components.

The system model is exported as an FMU to be handed over to Julia development environment and integrated to a PeN-ODE. The PeN-ODE is then trained with measurement data of predefined test scenarios gathered on a dedicated test rig.

The trained PeN-ODE is reimported to it's native development environment.

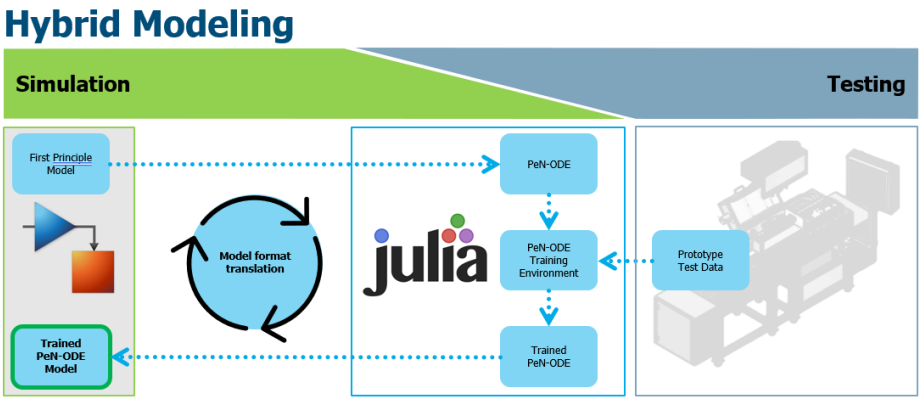

Figure 10: Process Chart

Development and application of PeN-ODEs The demonstrator focuses on the required steps from first principle model to hybrid model (PeN-ODE). Included parts are preparation of models for training (e.g. export to training environment), setup of the actual training environment, training strategies and feeding the PeN-ODEs or other artifacts back to the original development environment.

Development and application of normalizing flow methods for finger-printing In order to uniquely and reliably distinguish devices through their fingerprints, a probabilistic modeling method (BayesFlow) based on normalizing flows are to be applied. For reject-cause detection, i.e., pinch detection, a method IMFlow is to be further developed specifically for inverse system identification. For training the neural networks, sufficient data in various scenarios should be prepared with both the simulation model and the test bench. The demonstrator includes presentation of concrete finer-printing use cases and the results of identification, explaination of applied methods, displaying pinch occurrance and the detection results, and evaluation of performance of applied methods.

What is the goal of the demonstrator?

The demonstrator is aiming to show the capability to model effects, that are not part of physical models and therefore improve the quality of modeling. Using that kind of enhanced models will also enable the use of machine learning techniques for fingerprinting the characteristics of individual parts and therefore help to develop towards industry 4.0.

Development and application of PeN-ODEs As physical models in most cases concentrate on main effects, some – partly not well understood – minor effects are not modelled. PeN-ODEs will help to include those effects in models and therefore improve the quality of prediction.

Development and application of fingerprinting The goal of fingerprinting is to show, that machine learning algorithms (including PeN-ODEs in combination with further ML approaches), will allow to fingerprint individual part characteristics (e.g. dimensional tolerances) out of test rig measurement data.

How do you measure the success of the demonstrator?

The demonstrator will be rated against KPIs showing the prediction capability of the enhanced model and the capability to detect anomalies in part characteristics.

Development and application of PeN-ODEs The demonstrator is evaluated by measuring the achieved simulation fidelity in the target environment, comparing with measurements and first principle models for given reference scenarios. The models enhanced prediction capability will be quantified using statistical KPIs like R^2 and RMSE.

Development and application of fingerprinting Different actuators should be identified correctly. Specific abnormal behaviors of the steering actuator should be identified correctly. Overall 5 common abnormalities will be deliberately introduced in the actuator, e.g. missing lubricant. Uncertainty of identification should be quantified. (see FPP USP5)

Solutions for which challenges are demonstrated?

Development and application of PeN-ODEs The following challenges shall be addressed:

- How to execute workflow for generation and application of PeN-ODEs
- How to train PeN-ODEs; Overall best practices for working with PeN-ODEs
- How to make the method accessible to simulation and project engineers, but also how to deplay towards customers
- How can the model quality of PeN-ODEs be proven?

Development and application of fingerprinting The following challenges shall be addressed:

- How to identify different actuators though BayesFlow?
- How to detect pinch occurrance through IMFlow?
- In what conditions can the methods be applied and how well do they perform?

Technical information, relation to WPs

What information and data goes in and out and in what formats (PeN-ODE)?

Inputs of "PeN-ODE generation":

- Original model of the system (Simulink model)
- Accuracy requirements (format to be defined)
- Expected opration range (format to be defined)
- Corresponding Training data from test rig (format to be defined)
- Reference scenarios (format to be defined)
- Validation and Test Data (format to be defined)

Outputs of "PeN-ODE generation":

- Hybrid model or Neural Network for usage in Simulink (FMU or standardized NN-format, parameterfile, etc.)
- Information on model fidelity and application area

What information and data goes in and out and in what formats (fingerprinting)?

Inputs of "fingerprinting methods":

- Simulation model of steer-ny-wire
- Dataset of system parameters - outputs for fingerprinting
- Dataset of system inputs - outputs for pinch detection

Outputs of "fingerprinting methods":

- trained neural networks for fingerprinting and explaination
- trained neural networks for pinch detection and explaination
- validated methods and corresponding constraints

What information, data is used internally?

PeN-ODEs Training data, simulation model, model parameter, network parameter, hyper parameter

fingerprinting

- Simulation model for generating "easier and cleaner" data for training neural networks
- Relationship between different system variables (parameters, states, inputs, outputs) for better sensitivity analysis for inverse problems

Which tools and IT infrastructures are used?

PeN-ODEs

- original model is created in Matlab/Simulink
- model is exported as Functional Mockup Unit for application in Julia programming language
- PeN-ODE development/training is performed in Julia using FMI.jl and FMIFlux.jl
- PeN-ODE is exported as FMU or NN-data is stored for usage
- Application environment of PeN-ODEis Matlab/Simulink

fingerprinting

- framework of machine learning: PyTorch
- hardware for training neural networks: GPU
- different developed and well implemented normalizing flow architectures

Which methods, delivery items from OpenSCALING are used and why?

PeN-ODEs

- WP4: Representation of PeN-ODEs in standardized ways
- WP4: Enhancements for PeN-ODE training

fingerprinting

- WP2: The instinct uncertainty quantification of normalizing flow methods should be compatible with the standardized UQ representation

Demonstrator Analysis: Resource-aware embedded function of an orchestrated distributed control system

General information

What is the demonstrator to be used for?

The demonstrator is used to illustrate the benefits of a resource-aware function in a distributed control system. The benefits shall be evaluated in terms of:

- economic utilization of the available computational resources,
- maximizing the number of applications running on the same device,
- reduction of compute cost (in a pay per use setting),
- gained flexibility to develop a functionality agnostic of its target environment,
- robustness of the overall system against varying availability of dedicated computational resources, e.g., due to changing allocation of applications.

What does the overview structure look like?

The resource-aware control function shall be deployed as ROS node in a ROS2 environment, controlling a dynamic process, as depicted in the figure below. The ROS2 environment is setup such that nodes can be added and removed at runtime. The amount of computational resources consumed by individual nodes can be varied during runtime.

The resource-aware function node, as depicted below, communicates incoming and outgoing control signals with other nodes. The communication is managed in a asynchronous fashion. Information about the actual workload of the device and the actual execution time needs to be accessible by the resource-aware function. How to provide and manage this information is one of the research topics to be addressed. The resource information shall be used by a node providing an "Adaptation Mechanism" using the resource information to determine a trigger signal to the RAWF-node. The trigger signal is used by the RAWF-node to adapt its demand of computational resources.

There are currently three approaches how the resource demand of the RAWF could be scaled.

Approach A: Adapt MPC design parameters The first approach is based on the idea to adapt the computational demand of the MPC function. This could be realized, e.g., by relaxing constraints or adjusting the tolerance to influence the number of iterations of the optimizer.

Approach B: Switch MPC implementations The second approach is assuming that multiple MPC realizations exist that have a different computational foot print due to a different trade-off between accuracy and effort. The different implementations may differ by, e.g., a different prediction horizon, time step size or the expression of constraints and model.

Approach C: Adapt model interfaced with MPC The third approach fixes the controller structure and assumes that multiple interchangeable models exist that require different computational resources to be evaluated. Alternatively the model itself is scalable by means of runtime parameters influencing the computational effort required to solve the problem.

What is the goal of the demonstrator?

By means of an MPC control function, as example of an adaptive function, realized in a ROS2 environment, it shall be demonstrated that a physical process can be controlled despite of changing resource demands of simultaneously running nodes.

Different approaches to realize a resource-aware function shall be evaluated in terms of feasibility, performance, applicability and realization effort.

In terms of feasibility and performance the different approaches shall be manually implemented. For the most promising approaches it is desired to apply a development workflow based on open model exchange standards and code generation techniques. Dependent on the availability of respective tool support a workflow to realize realtime capable model-based control functions in a partially automated tool chain shall be applied and tested. Tools and methods to plug the prediction model into an MPC function as embedded code is expected to be provide by the tool vendors.

The integration into ROS2 will focus on providing an interface description utilized in a manual process to show case the capability.

After the project the concept shall be transferred to the application domains.

- In the automotive domain autonomous driving applications in distributed system with v2x (vehicle to x) communication or applied to new E/E zone architectures shall be improved in terms of runtime adaptability.
- In the field of industrial automation zero down-time updates and more cost efficient utilization of computational resources shall be enabled.

How do you measure the success of the demonstrator?

The success of the demonstrator is evaluated by showing that the proposed resource-aware distributed control system is able to adapt under short-term lack of computational resources and communication delays. The cost saving potential of the proposed resource-aware function shall be quantified by the resource utilization compared against state-of-the-art static allocation based on worst-case assumptions.

Solutions for which challenges are demonstrated?

The following challenges shall be addressed:

- Which interfaces have to be provided by the environment to enable the envisioned adaptability of the application?
- How to realize a resource-aware adaptable function in ROS2?
- How to realize an MPC as resource-aware function?
- How to leverage Modelica/BaseModelica/FMI/eFMI and CasADi/ACADOS to realize an MPC in ROS2?
- How to generate embedded code from a model preserving the structural parameters as runtime parameters?

Technical information, relation to WPs

What information and data goes in and out and in what formats?

Inputs of "Resource-aware function development process":

- Signal interface of the control function with the environment (sensors, actuators)
- Prediction model of the physical system (plant) to be controlled
- Requirements (constraints/objective) to be satisfied by the control function
- Control law implementation (MPC)
- Resource-aware function mechanism to influence the computational foot print
- Resource-aware-function interface (trigger to change modes)
- Adaptation mechanism (decision making)
- Resource information interface

Outputs of "Resource-aware function development process":

- RAWF ROS2 node
  - building blocks dependent on the realization approach

What information, data is used internally?

Dependent on the realization approach the prediction model is provided as a single model, multiple interchangeable models or as a single scalable model. The prediction model is to be integrated as C code into the MPC function by providing call back functions to the optimizer. This includes the forward resp. backward sensitivities.

Alternatively the prediction model is provided as eFMI block to be integrated with a generic MPC implementation.

Which tools and IT infrastructures are used?

Plant model, predicton model and test cases shall be created in Dymola. The eFMU's shall be generated by Dymola and the integrated embedded code generator. As AD framework CasADi or ACADOS shall be used. The generated code is manually integrated as ROS2 node in a ROS2 environment.

Which methods, delivery items from OpenSCALING are used and why?

- WP3: array preserving compilation and eFMI generation

Requirements

Large-scale system (LSS) Modelica models (Innovation 1)

Required

- Change not only size of simple arrays (e.g. Real) but also arrays of complex models like TIL Cells (including differential states, event indicators, replaceable models, stream connectors, external C functions, ...) (Demonstrators 1.4 & 1.5)
- Scalable prediction model (Demonstrator 1.9)
  - The prediction model integrated with the MPC control law must be scalable in terms of computational effort at runtime.
  - This could be realized either by switching between alternative implementations of the model or by changing a structural parameter that influences the fidelity of the model, e.g., the discretization of a PDE.
- Standardized BaseModelica output of the prediction model (Demonstrator 1.9)
  - The Modelica compiler shall support the output the prediction model in BaseModelica.
  - This output format serves as input to other frameworks such as CasADi and ACADOS.

Optional

- Ability to compute large scale systems e.g. many HVAC systems in buildings (Demonstrator 1.4)
- Layer visualization of Modelica models, e.g. being able to display only fluid layer, control layer, etc. Option to move the actual connection line (even if made transparent to highlight a different layer) to the background to avoid component icon cluttering. (Demonstrator 1.5)
- Number of ports is dynamically allocatable (Demonstrator 1.4)

Requirements by Standard

eFMI

- Support structural parameters
  - The eFMI Algorithm Code and Production Code derived from the model need to preserve structural parameters as tunable parameters.
- Support switching between models
  - One eFMU shall allow to contain multiple realizations of the same model and provide a tuneable parameter to trigger the switching between these implementations at runtime.

Acausal precompiled FMU component models (Innovation 2)

Required

- Pre-compile acausal components including arrays of complex models like TIL Cells (including differential states, event indicators, replaceable models, stream connectors, external C functions, ...) (Demonstrator 1.5 & 1.4)
- Let user/ library developer define which model parts are pre-compiled. (Demonstrator 1.5)
- Inter-tool operability of FMUs i.e. same FMU should work with multiple tools (Demonstrator 1.4)

Optional

- Limitations on Acausal precompiled FMU should be described, needed by model developers. (Demonstrator 1.4)

Standardized representation of uncertainty information (Innovation 3)

- Model parameter uncertainties shall be quantified by a probability distribution for each parameter, with a parameter range as simplest representation. (Demonstrator 1.6; Parker)
- Aggregated model credibility shall be representated by a scalar number that can vary dynamically (during simulation) due to variations in FMU inputs and parameters. (Demonstrator 1.6; Parker)
- Support for parsing and interpreting aggregated static information of credibility, in a simulator consisting of multiple constiuent models (FMUs, SSPs), originating in the contribution of the individual component/subsystem models. For example, visualizing overlapping regions of the constituent models operational domains. (Demonstrator 1.6; Saab, VTI)
- Tool support to express uncertainty quantification results in a model operational domain (DoV_UQ). (Demonstrator 1.6; Saab, VTI)

Representation of PeN-ODEs in modelling standards (Innovation 4)

The following statements put requirements to tool enhancements to improve the modeling and training of PeN-ODEs. Some of these are only possible with an extension of the modelling standards. For others this has to be determined during the project.

- A PeN-ODE shall be executable as a FMU (ABB, Airbus, Bosch; currently used)
- A FMU shall be usable in a MPC framework(ABB; special needs w.r.t. discrete states)

- An eFMU shall be usable in a MPC framework(Bosch, DLR)
- A PeN-ODE shall be trainable as a FMU (ZF, UNA, TLK, Bosch; model only available as FMU)
- A PeN-ODE with several NNs shall be trainable as a FMU (UNA, TLK, Bosch; model only available as FMU)
- A PeN-ODE shall be defineable in a Modelica Tool (TLK, Bosch; define which submodel shall be replaced etc.)
- It shall be possible to integrate a trained PeN-ODE (as Modelica component or with physical connectors at least) in a Modelica Tool (TLK, Bosch; having the NNs in Modelica w.o. much effort)
- A FMU containing a PeN-ODE shall allow to trace to the training data (Airbus, TLK, Bosch; exchange of PenODEs)
- A FMU containing a PeN-ODE shall allow to trace to UQ or similar activities for credibility assesment (Volvo, Saab, Bosch)
- An eFMU shall be capable of efficiently evaluating a PeN-ODE with the same behavior as in the training environment (DLR, Bosch)
- An eFMU containing a PeN-ODE shall be capable of modifying NN weights during runtime (DLR)
- A FMU containing a PeN-ODE shall be capable of modifying NN weights (give certain parameters of the FMU a meaning)(UNA, Bosch, ZF,. . .)
- Efficient gradient calculation for fast training (DLR; fast gradients from FMUs or Modelica models)

Training methods for surrogates of large-scale systems (Innovation 5)

- A PeN-ODE training shall be configurable for model developers (Airbus, Bosch)
- Realtime capability of the resulting PeN-ODE (ABB; predictive control)
- Change/tune parameters or submodules (ABB, Airbus; compare different electrolysers, sizing)
- Usage of (sparse) real data (ZF)
- Capability for deployment on realtime systems (Airbus, Bosch, DLR)
- Complement methods with best practices and/or FAQ (TLK, DLR)
- Speed-ups 10x to 100x (Bosch;virtual controller testing)
- Methods for data generation (Bosch, TLK; WHAT data shall be generated)
- PeN-ODEs shall be exportable as an FMU (ZF)
- It shall be possible to add constraints in the PeN-ODE training (e.g. energy conservation)
- PeN-ODEs shall be allowed to have discontinuities

Resource aware functions (Innovation 6)

- Workflow to realize prediction model as MPC function
  - The prediction model together with the formulation of the optimization problem considering the constraints and the objective, must be

integrated with an optimizer. A workflow must be provided that allows to generate and integrate the required artifacts applicable to an implementation on ROS2.

- Workflow to integrate prediction model realized as MPC function as node in ROS2
  - The MPC function consisting of the prediction function and the optimizer must be realized as ROS2 node.
  - This integration could be either by providing a generic MPC node calling a standardized interface provided by FMI/eFMI.
  - Or by providing all functions as C code that is individually integrated in a manual process. -Differentiable prediction model
  - The prediction model must be at least one time continuously differentiable.

Requirements by Standard

eFMI

- Support of forward sensitivities
  - The prediction model must provide its sensitivities of the outputs considered in the objective function with respect to the control inputs. This is also referred to in FMI as *directional derivatives*.
- Support of backward sensitivities
  - The prediction model must provide its sensitivities of the control inputs with respect to the outputs considered in the objective function. In the context of FMI this is also referred to as *adjoint derivatives*.

Modelica

- Formulation of optimization problems
  - The optimization problem is characterized by constraints applied to the control outputs and the objective function. The Modelica language shall provide standardized language constructs to define these properties such that they can be provided as additional information to CasADi or other framework to generate the corrsponding MPC function.

Large Scale System workflow requirements

Listed below are the requirements necessary to enhance the Large Scale System workflow.

- Export of system architectures (the architecture of the physical system in focus) shall be done according to System Structure Description (SSD) xml format of the System Structure and Parameterization (SSP) [Demonstator 1.6]

- Import of system architectures shall be done according to System Structure Description (SSD) xml format of the System Structure and Parameterization (SSP) [Demonstator 1.6]
- Export of sub-system interfaces shall be done according to the ModelDescription.xml format of the Functional Mock-up Interface (FMI), or according to the System Structure Description (SSD) xml format of the System Structure and Parameterization (SSP), depending on the granualarity required by the intended-use. [Demonstator 1.6]
- Import of sub-system interfaces shall be done according to the ModelDescription.xml format of the Functional Mock-up Interface (FMI), or according to the System Structure Description (SSD) xml format of the System Structure and Parameterization (SSP), depending on the granualarity required by the intended-use. [Demonstator 1.6]
- Modelling and Simulation tools shall support incremental addition/removal of resources in analysis architectures (FMUs, ModelDescription files, SSPs, SSDs, SSMs, SSVs, SSBs) to an SSP file [Demonstator 1.6]
- Tools shall support the incremental addition/removal of meta-data capturing life-cycle information and uncertainty quantification information. The meta-data is to be expressed on a standardized format (STMD, DTMD, SRMD of the SSP Traceability specification) to fascilitate interoperability towards other exploited standards and for simplified implementation in all tools relevant during development. Examples of meta-data to capture are: artifact generation/modification date, purpose, static metrics of credibility, dynamic metrics of credibility, etc. [Demonstator 1.6]
- Export of analysis architectures (the architecture of the model(s) representing selected aspects of the architecture of the physical system in focus) shall be done in machine readable format according SSP standard. [Demonstator 1.6]
- Tools shall support an agreed upon standardized format for capturing time series data, for example for exchanging simulation boundary conditions and model validation an verification data. [Demonstator 1.6]
- A single component model's properties' impact on the aggregated information of a larger model (that contains the single component model) shall be traceable to the single component model. [Demonstator 1.6]
- Support for authoring, parsing, and executing nested SSPs for maximising simulation artifact reuse and configuration managment.