FMI-based Model Exchange for Aircraft Energy Systems

Modelica FMI Tutorial

Dirk Zimmer
German Aerospace Center (DLR)

Tim Giese
Airbus Operations GmbH

Matthieu Crespo
Liebherr-Aerospace, Toulouse

Sébastien Vial
Airbus Operations SAS

Motivation: Why using Model Exchange?

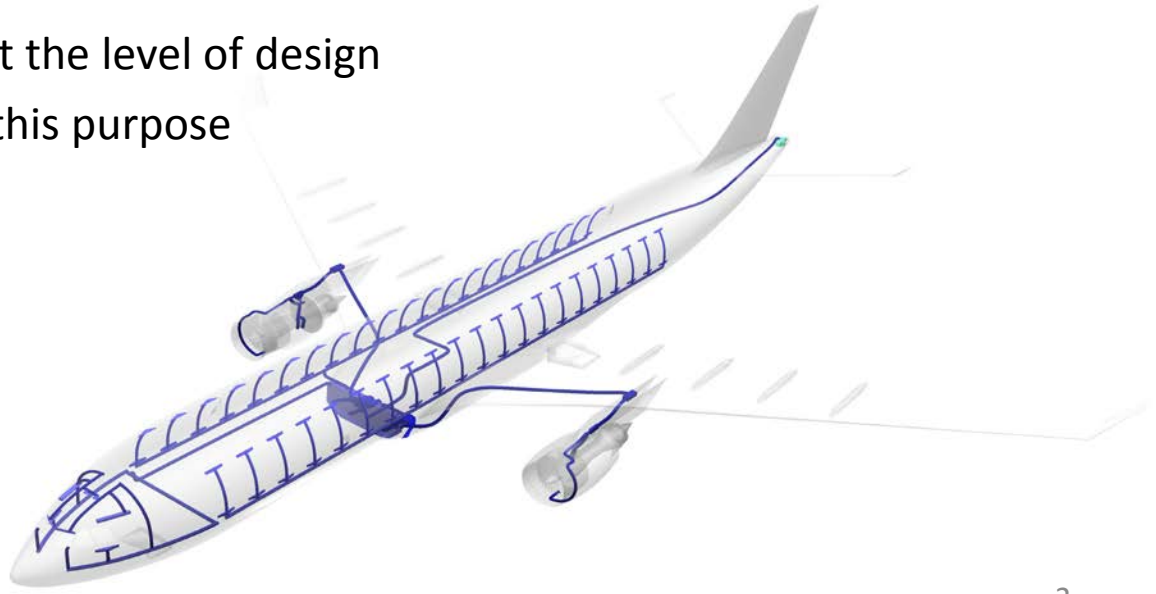
Current Situation:

- Already now, component developers and system integrators engage heavily in creating mathematical models of their systems.
- Popular Tools: Modelica tools or Matlab Simulink, etc…
- In a globalized world, collaboration is needed already at the level of early design

Future Goal:

- Enable better cooperation at the level of design

➔ Exchange these models for this purpose



Applications of Model Exchange

Overall Integration and Optimization

- Centrally integrate all models
- Enable a total system simulation at early design stage
- Enable optimization at early design stage

Executable Specification

- Express the requirements in form of a model
- Exchange models together with formal requirement documents

Distributed (Virtual) Testing of Components

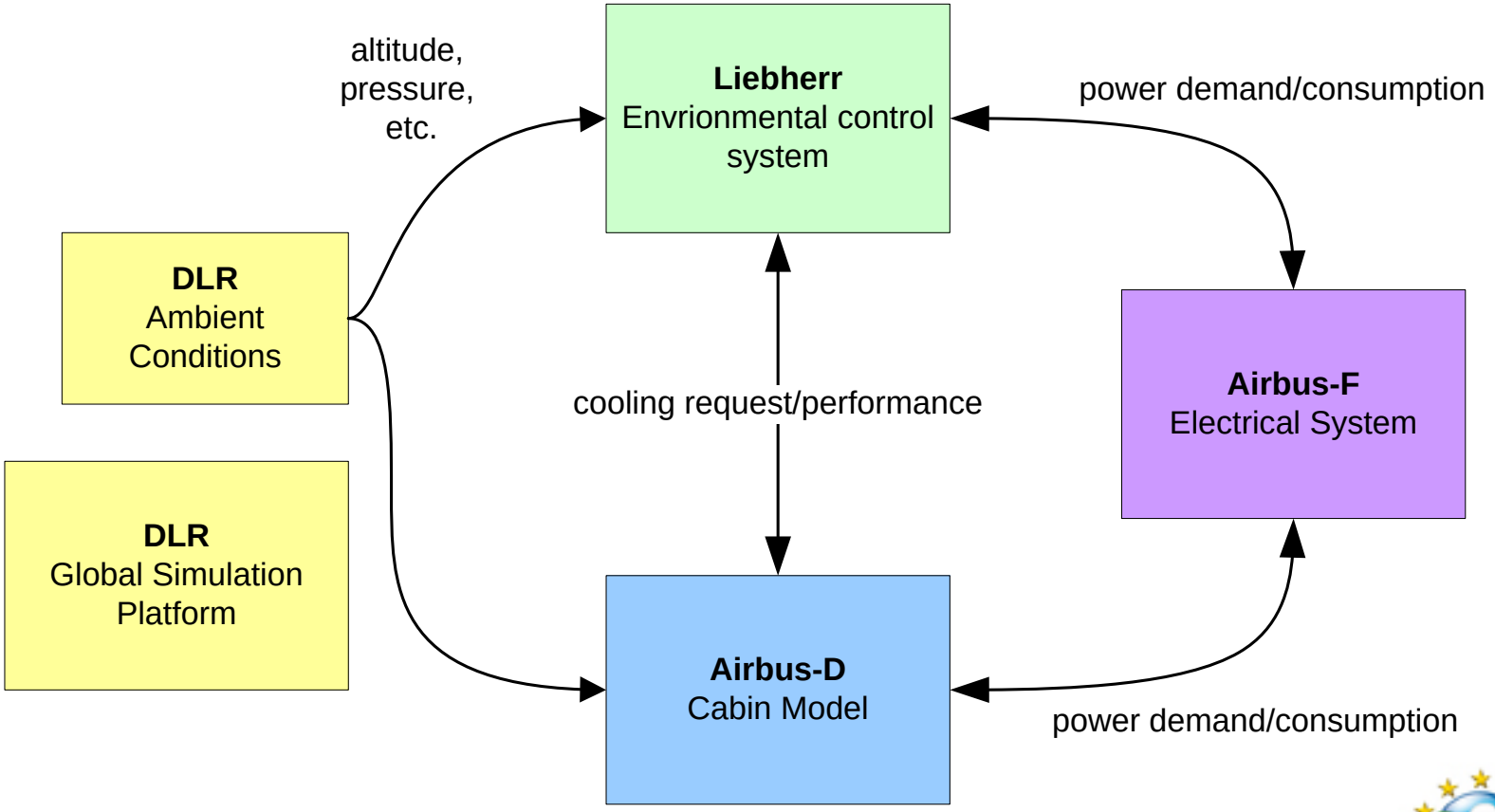
- Use model exchange to allow for virtual testing
- Manufacturer may send model of the environment to component supplier
- and vice versa….

Our Use Case: Energy System

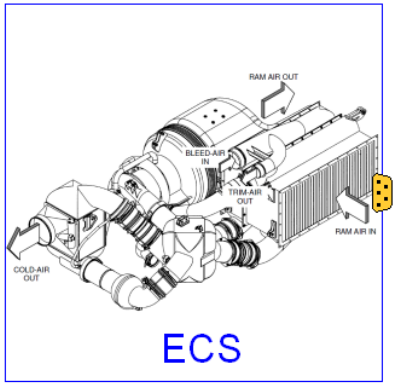
- An Aircraft contains many sub-systems.
- Three of them have been selected for our use case
- Within SGO WP2.1, we aim at a system model for:
  - Cabin,
  - ECS,
  - Electric Power.
- Each component is separately developed.

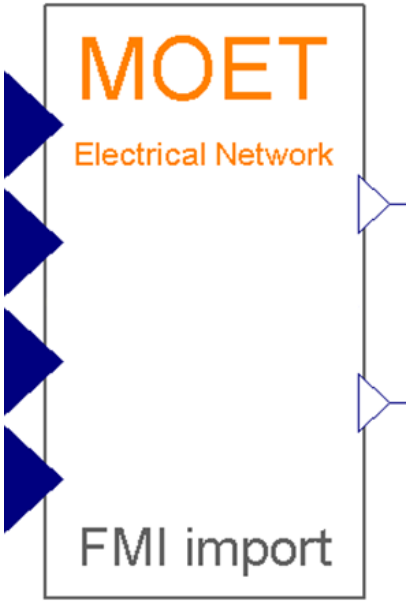
www.cleansky.eu CLEANSKY

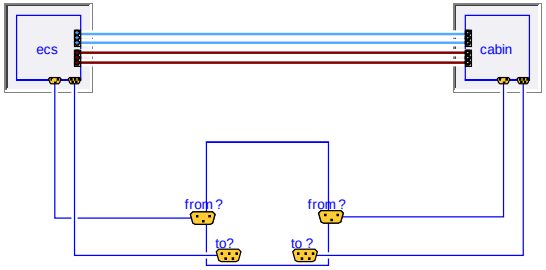
Our Use Case: Energy System

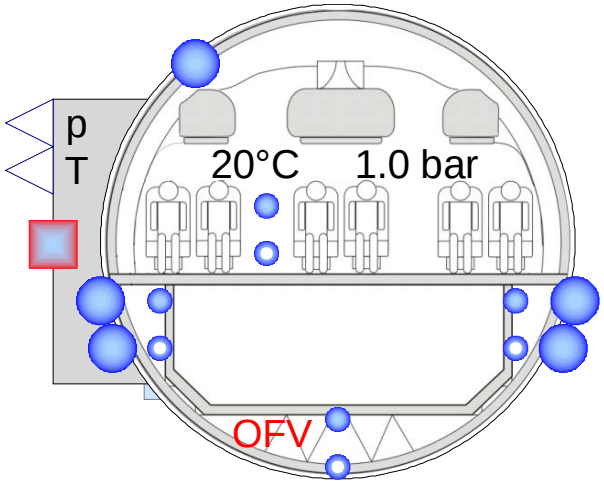


Our Use Case: Energy System



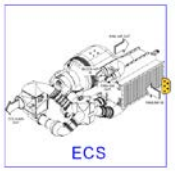






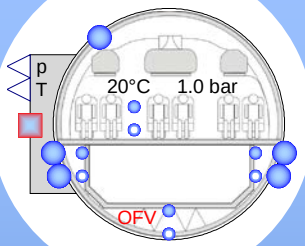
www.cleansky.eu

Our Use Case: Energy System



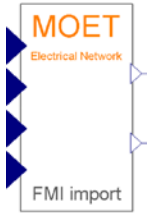
Environmental Control System

- Model exists for right and left pack.
- > 2x 30 interface variables
- >2x 700 equations
- Model contains complex non-linear equation systems



Cabin model

- > 40 interface variables
- > 1000 equations
- Model behavior can be very stiff.

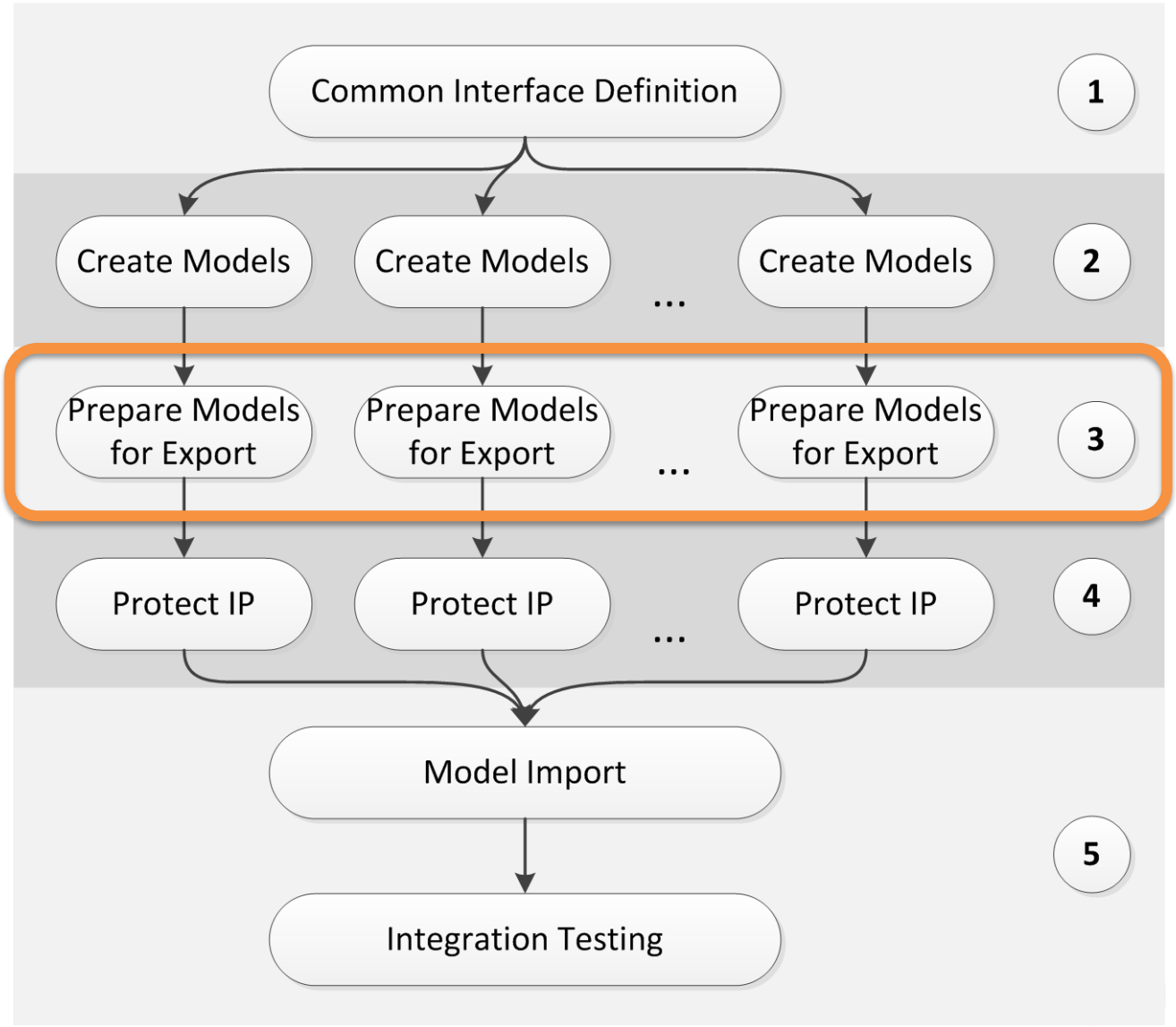


Electric System

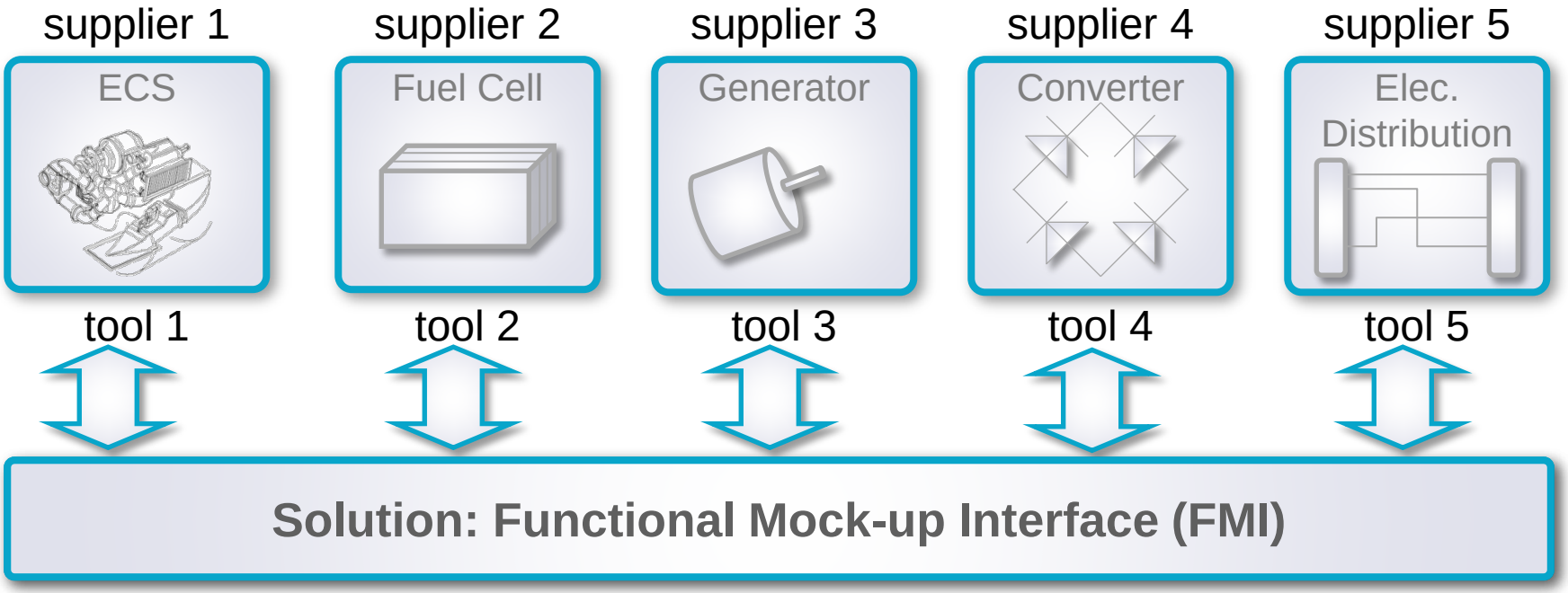
- > 35 interface variables
- > 3000 equations
- Model contains discrete events

The models are quasi steady-state models

Model Exchange Process



Step 3: Export Models



- For Export a cross-tool standard is required.
- The Functional Mock-up Interface offers this standard.

Step 5: Integration (c), (d)

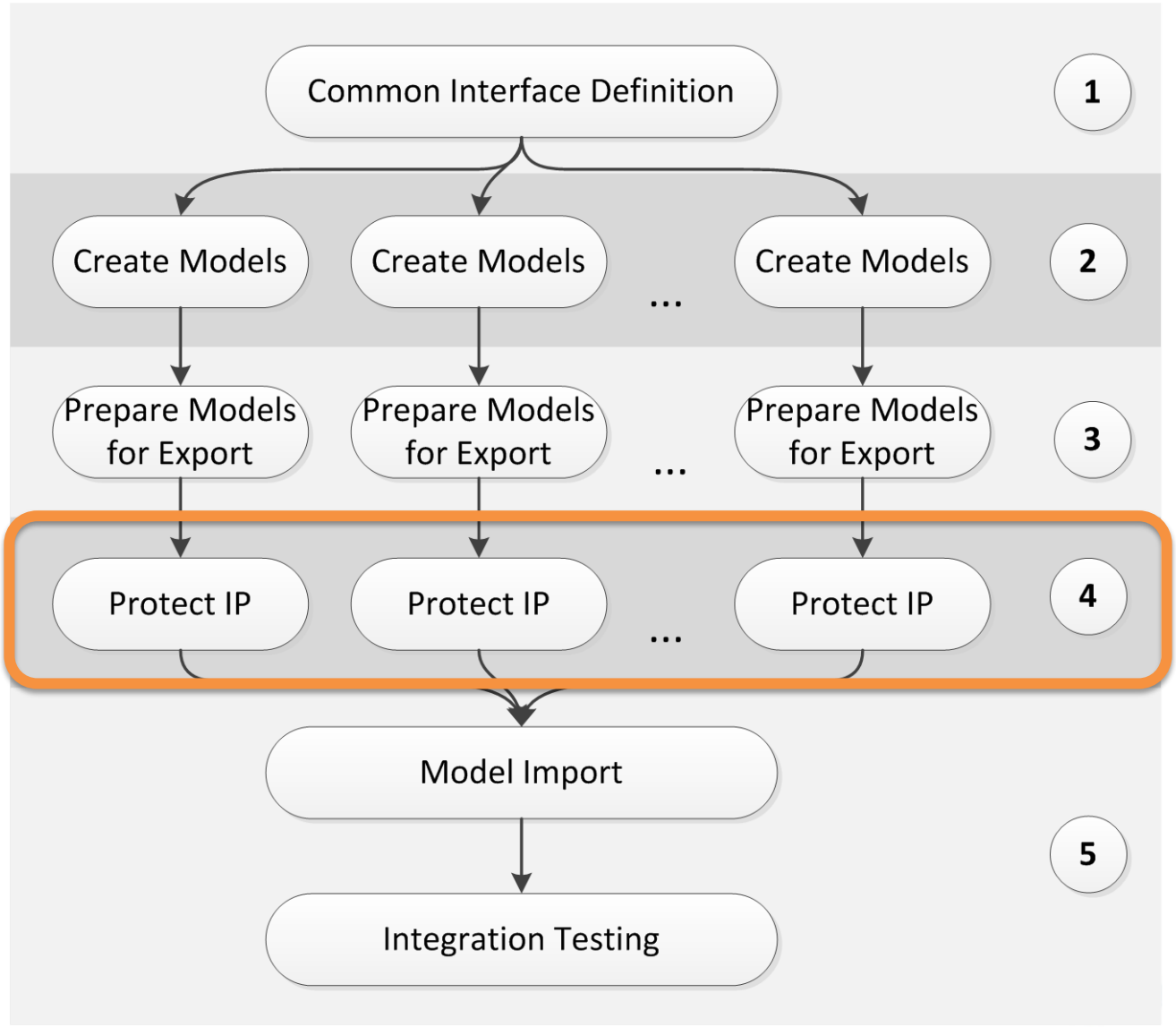
FMI for Model Exchange	FMI for Co-Simulation
• $dx/dt = f(x,u,t)$	• $x_{t+h} = f(x_t,u,t)$
• does not include ODE-solver	• includes ODE solver

- Both version of FMI support hybrid systems with discrete events.

Step 3: Export Models

- FMI Standard: www.fmi-standard.org
- Tutorial on how to use FMI
- A Functional Mock-up Unit (FMU) consists in two parts:
  - An XML Description of the Model inputs/outputs, state-variables and parameters
  - Executable or compilable Code that implements a given set of functions defined in the FMI standard.
- Further publications:
  - Torsten Blochwitz et al., *FMI 2.0: The Standard for Tool independent Exchange of Simulation Model*
  - Sofia Gedda et al., *Derivative-free Parameter Optimization of Functional Mock-up Units*.
  - Manuel Gräber, *Using Functional Mock-up Units for Nonlinear Model Predictive Control*.
  - From Proceedings of the 9th International Modelica Conference, September 2012, Munich, Germany.

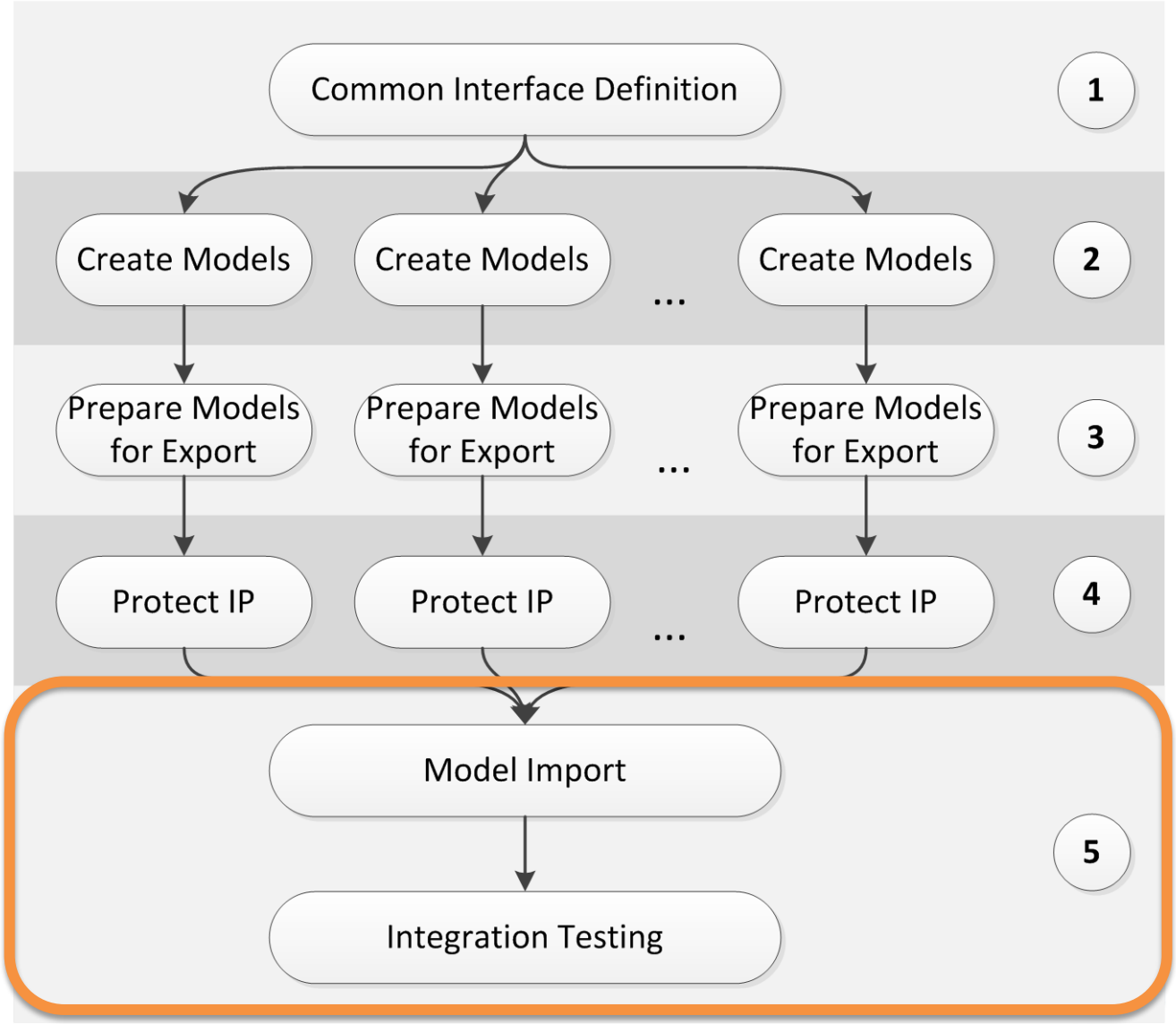
Model Exchange Process



Step 4: Protect Intellectual Property

- When exporting an FMU, the intellectual property must be protected.
- Part 1 is the XML-Description. Crucial parameter values or revealing variable names that expose the internal model structure are removed by a a Python Script.
- Part 2 is the actual code. Binary code provides a sufficient degree of obfuscation. For C-code, already existing tools for source-code obfuscation can be applied such as:
  - www.stunnix.com/prod/cxxo/overview.shtml
  - www.morpher.com/
  - www.pcsentinelsoftware.com/products/mangleit

Model Exchange Process

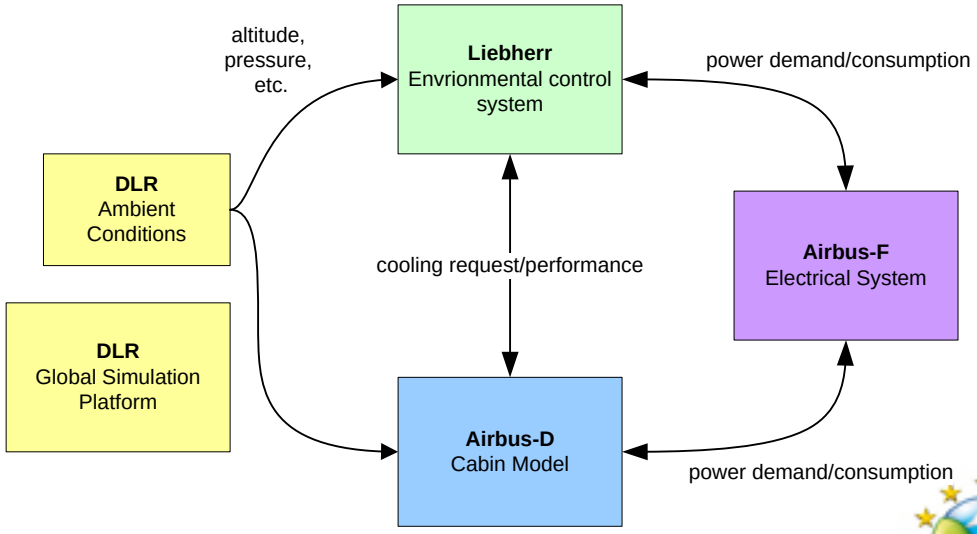


Step 5: Integration

- Given a proper definition of the interface from step 1 that defines the connection pattern, the integration should work seamlessly in theory.

- In practice, however, there are a number of problems that can occur. The main problem fields reveal to be:

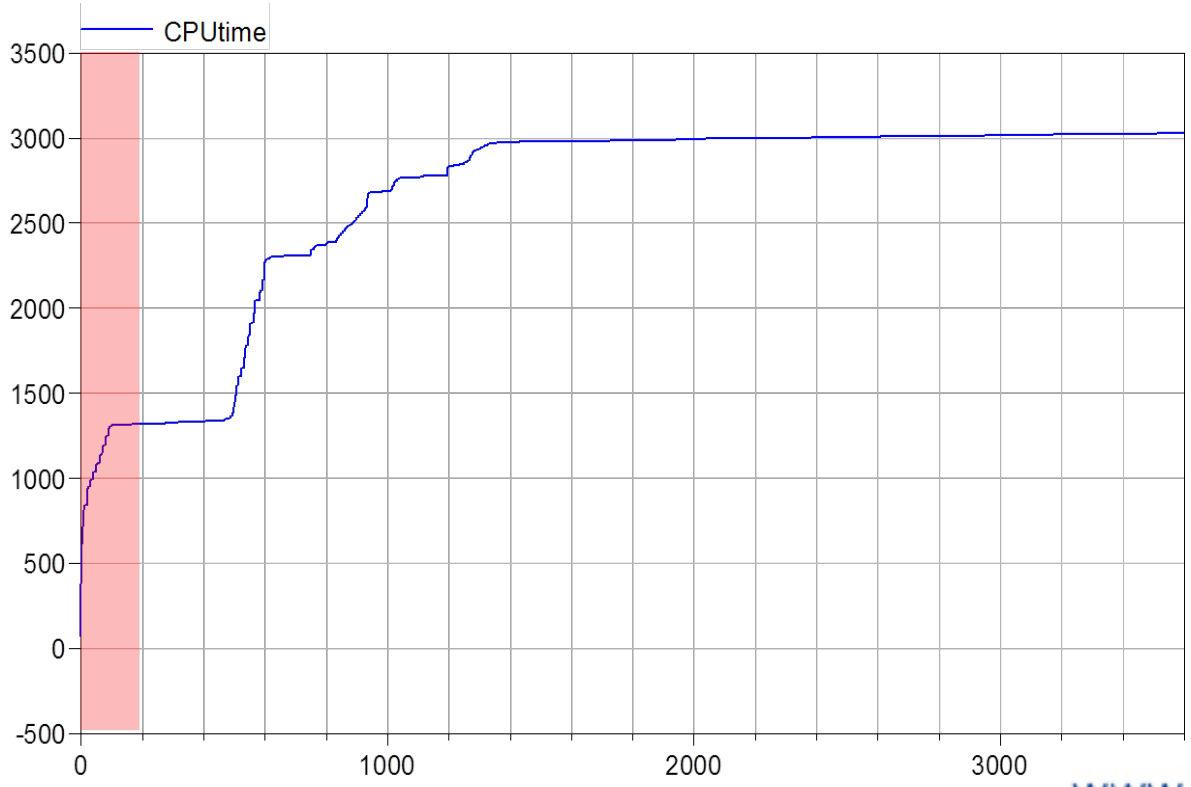
a) Initialization
b) Performance
c) Robustness
d) Partial Integration Capability



Step 5: Integration (a) Initialization

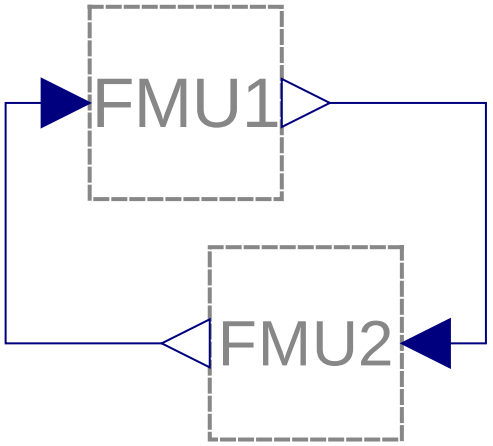
- At the beginning the ECS and Cabin model undergo fast transients to approach their steady state

- Initialization is a critical and computationally demanding phase.



Step 5: Integration (a) Initialization

- **Method 1:** Avoid algebraic couplings between FMUs that would create larger non-linear systems.

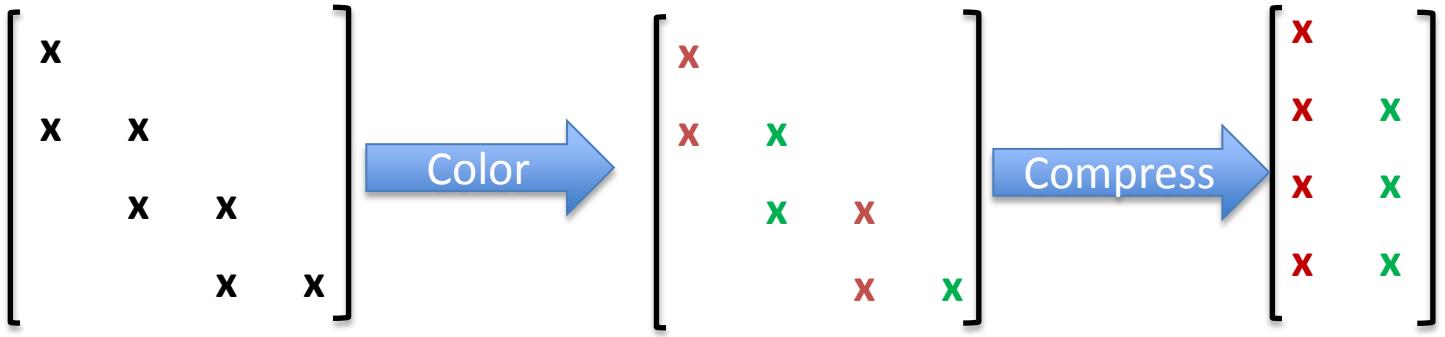


- **Method 2:** Use safe default values for inputs at t=0 and then later couple the sub-systems by using a form of homotopy

$$u = \lambda \cdot y + (1 - \lambda) \cdot d$$

Step 5: Integration (b) Perfomance

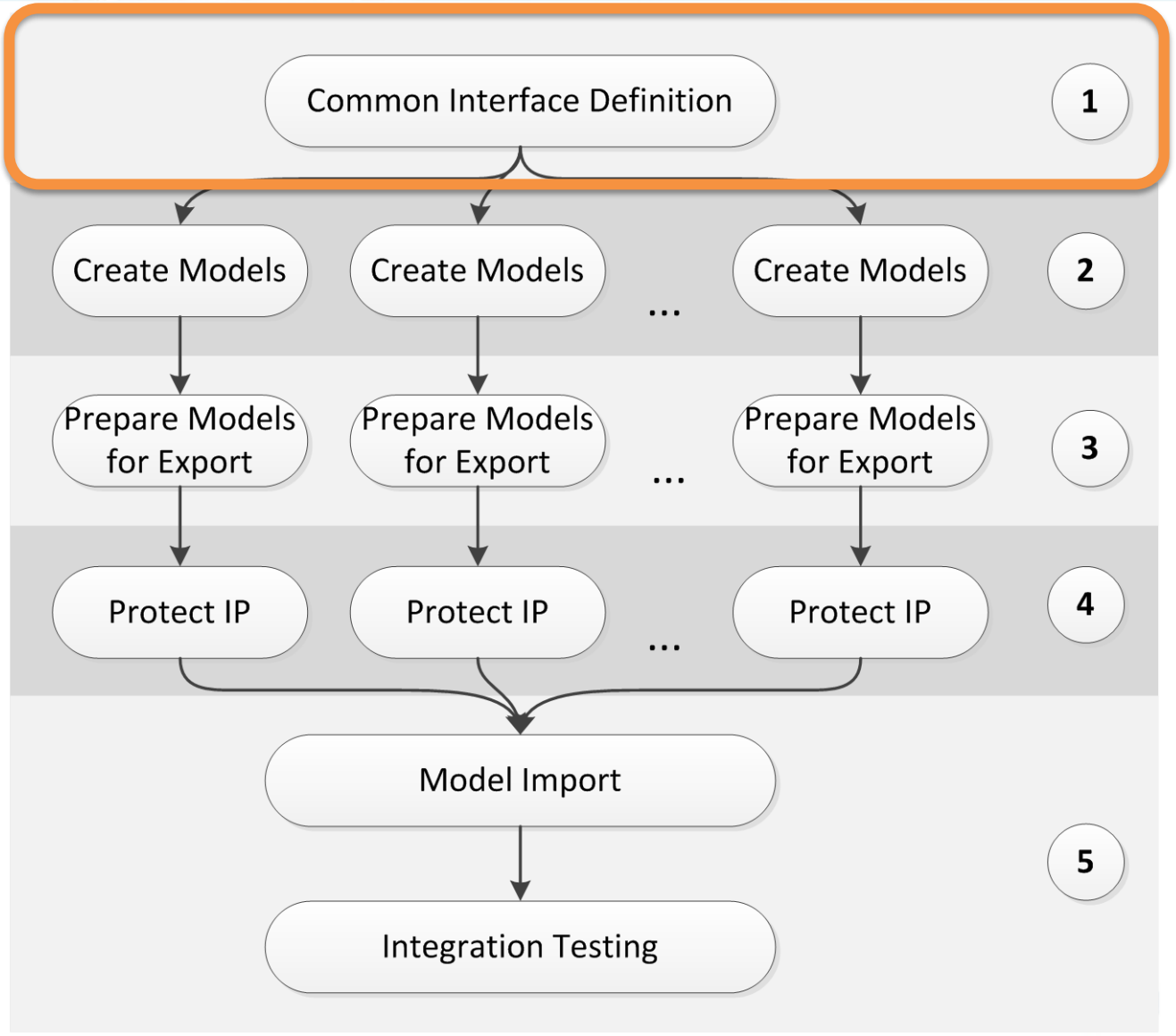
- Simulating with FMUs is often slower than simulating the model directly.
- In FMI v1.0, there was no support for sparse Jacobians (needed for implicit solvers such as DASSL). Yet the performance in our use-case was acceptable.
- FMI v2.0 now supports sparse Jacobians and is expected to improve performance. Currently only a factor of 2 could be measured.



Step 5: Integration (c), (d)

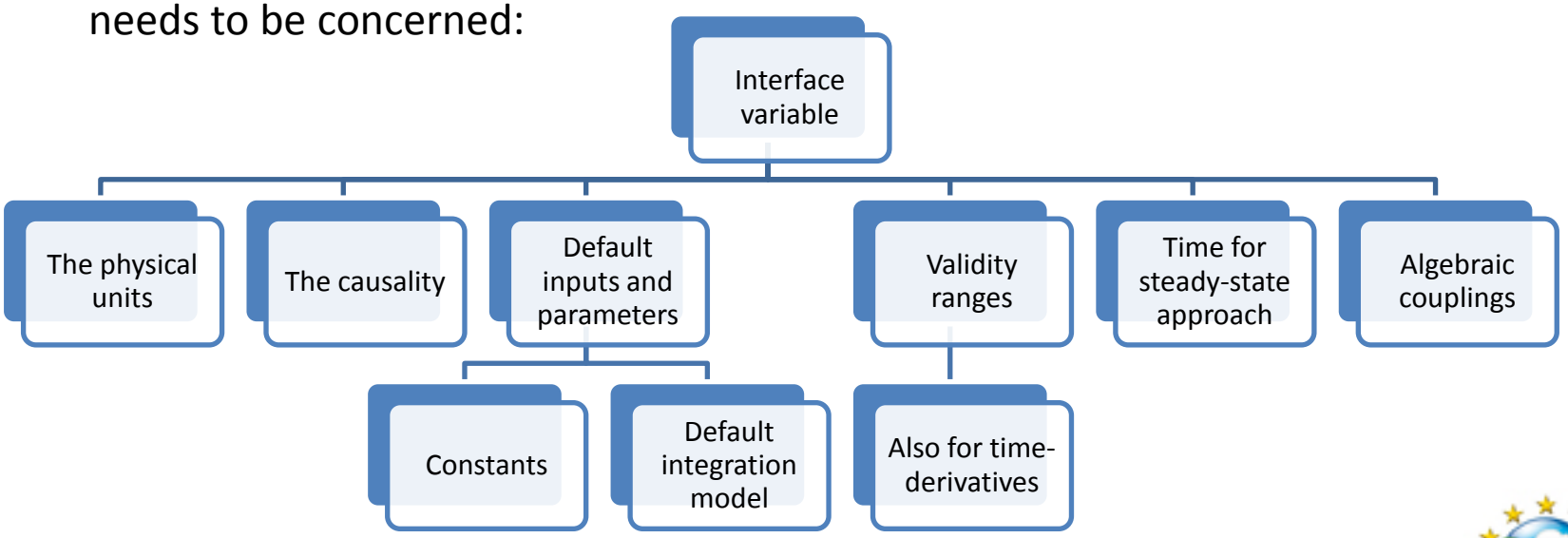
Robustness	Partial Integration Capability
• Validity Ranges for inputs required • Validity Ranges for time-derivative of inputs required	• Partial Integration meaningful for testing and analysis • Default inputs (constants or models) required

Model Exchange Process

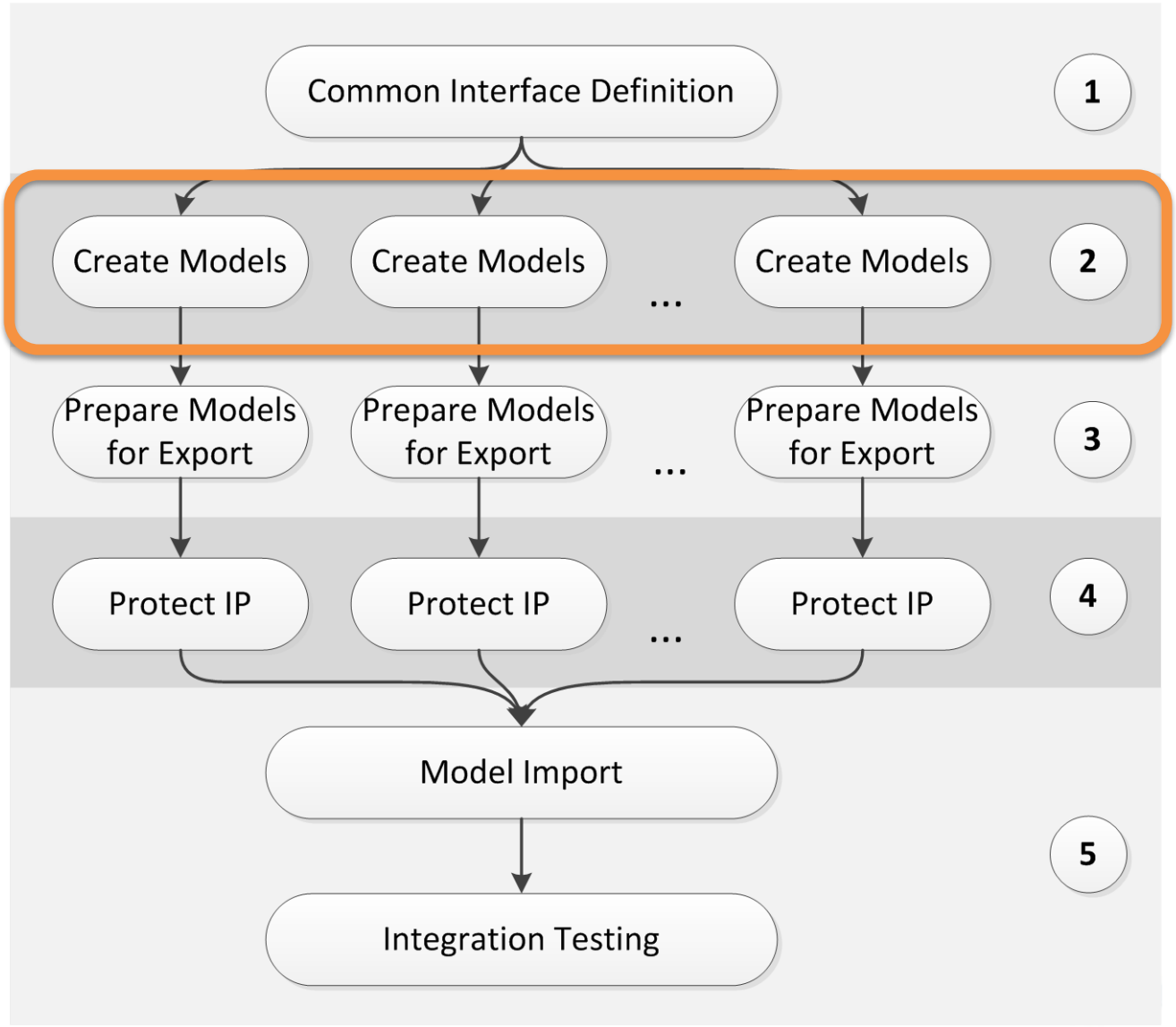


Step 1: Common Interface Definition

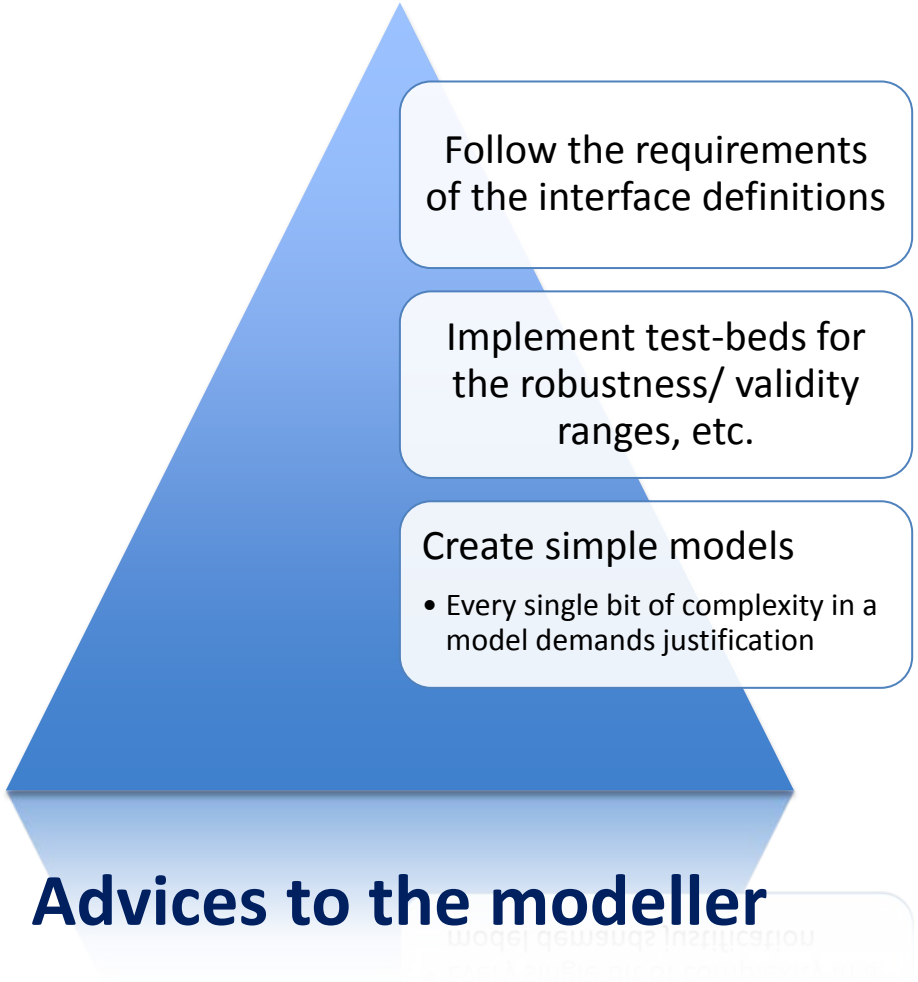
- The definition of interfaces consists in the definition of the physical boundary variables connecting the corresponding sub-systems.
- However the model exchange process requires more than just the definition of this set.
- For each interface variable in the set, a number of additional information needs to be concerned:



Model Exchange Process

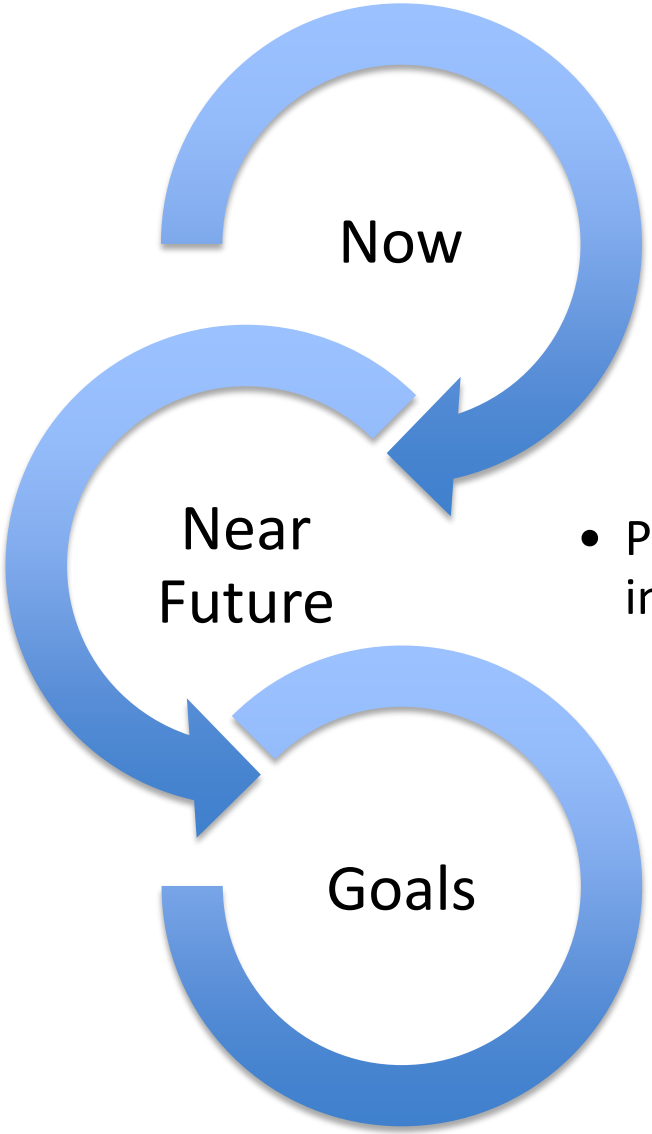


Step 2: Model Creation (or Adaption)



Advices to the modeller

Conclusions



- First cross-platform, cross-company integration of a complete energy system with plausible results
- Collected a lot of experience
- Technology is principally mature enough

- Provide a detailed guideline that incorporates our experience

- Use of Model Exchange for Executable Specification
- Use of Model Exchange for Virtual Testing

Recommendations for tool providers

- It is important that validity ranges can be easily specified and exported via modelDescription.xml. Also validity ranges on the rate or dependent on the state are needed.
- In practice, not only FMUs are exchanged but FMUs with test beds.
- A tool should offer means to perform local model robustness tests. For instance, performing tests by perturbing input according to validity ranges.
- The integrator should have means that enable him to see whether algebraic loops are created over an FMU.
- Having the option to automatically connect via naming conventions would be helpful. Also partial integration is a frequent case.