

Unambiguous Power System Modeling and Simulation using Modelica Tools

Dr.-Ing. Luigi Vanfretti

E-mail: luigiv@kth.se

Web: <http://www.vanfretti.com>



SmarTS Lab
Smart Transmission Systems Laboratory

Docent and Assistant Professor
Electric Power Systems Dept.
KTH Royal Institute of Technology
Stockholm, Sweden



Statnett

Scientific Advisor
R&D Division (FoU)
Statnett SF
Oslo Norway

7th MODPROD Workshop on Model-Based Product Development

Linköping University

Feb. 5, 2013

Acknowledgment

- The work presented here is a result of the collaboration between RTE (France), AIA (Spain) and KTH (Sweden) within the EU funded FP7 iTesla project: <http://www.itesla-project.eu/>
- Previous results from the FP7 Pegase project have been provided by RTE and used in this work.
- The following people have contributed to this work:

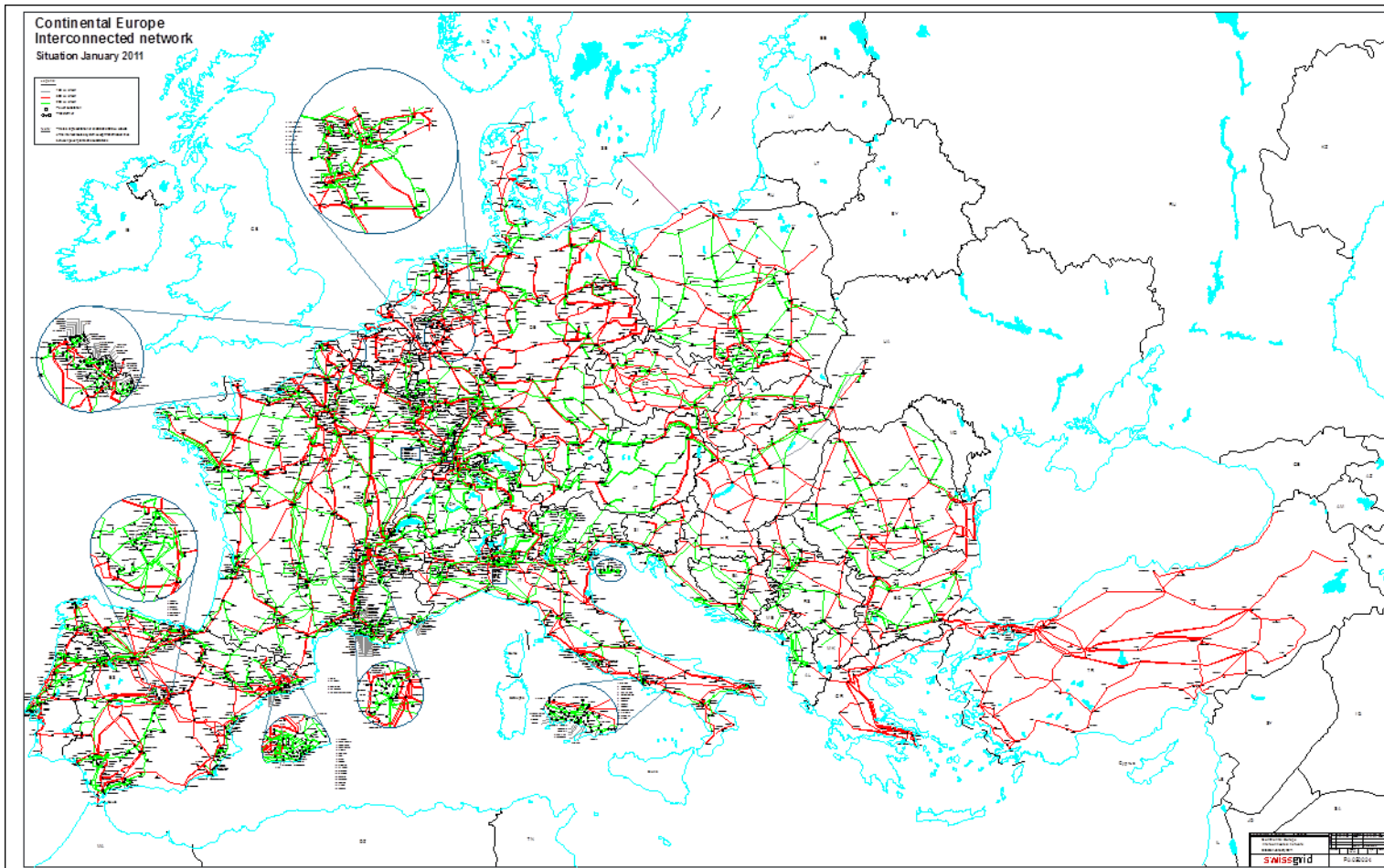


- RTE: Patrick Panciatici, Jean-Baptiste Hyberger, Angela Chieh
- AIA: Gladys De Leon, Milenko Halat, Marc Sabate
- KTH: Wei Li, Tetiana Bogodorova

Outline

- Background: *domain specific* state of the art
 - Power systems dynamics and domain tools for large-scale simulation
 - Modeling limitations, inconsistency and consequences
 - Model exchange in the IEC CIM era
- Unambiguous power system modeling and simulation
 - Building blocks for power system simulation: iTesla PS Modelica Library
 - Consistent model sharing across different modeling tools
 - Adequate Modeling (... a case for controls and protections)
 - Large scale model simulation (... and initialization issues)
 - Using FMUs for model sharing and exploiting generic solvers
- Perspectives on the use of Modelica for
 - Unified CIM-compliant Multiple-Time Scale Power System Modeling
 - Smart Grids: cyber-physical modeling of power systems

Large Scale Power Systems



Large Scale Power Systems

To operate large power networks such as these planners and operators need to analyze variety of operating conditions – both off-line and in *near* real-time (power system security assessment). Different SW systems have been designed for this purpose.

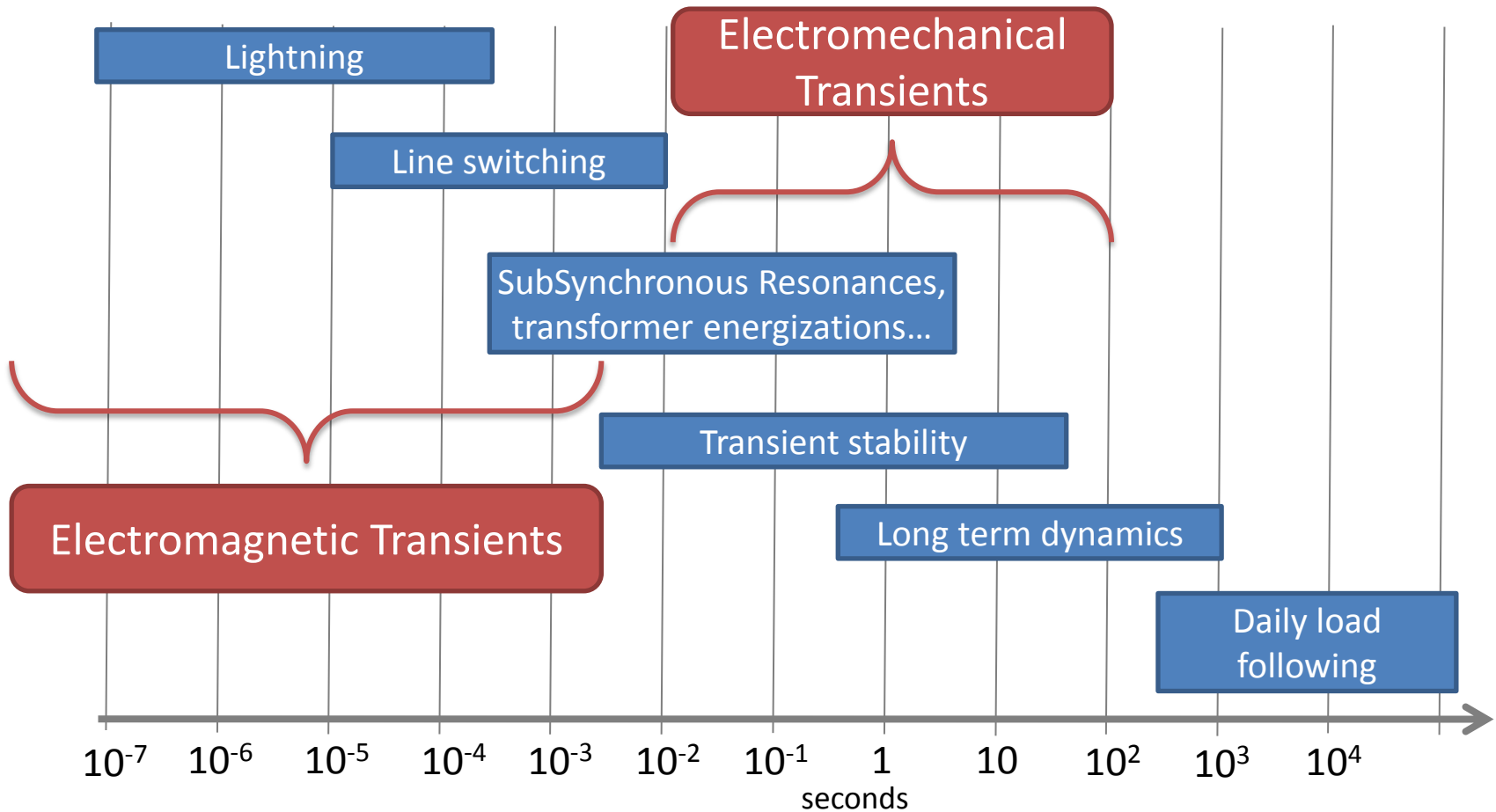
But, the dimension and complexity of the problems are increasing due to growth in electricity demand, lack of investments in transmission, and penetration of intermittent resources. **New tools are needed!**



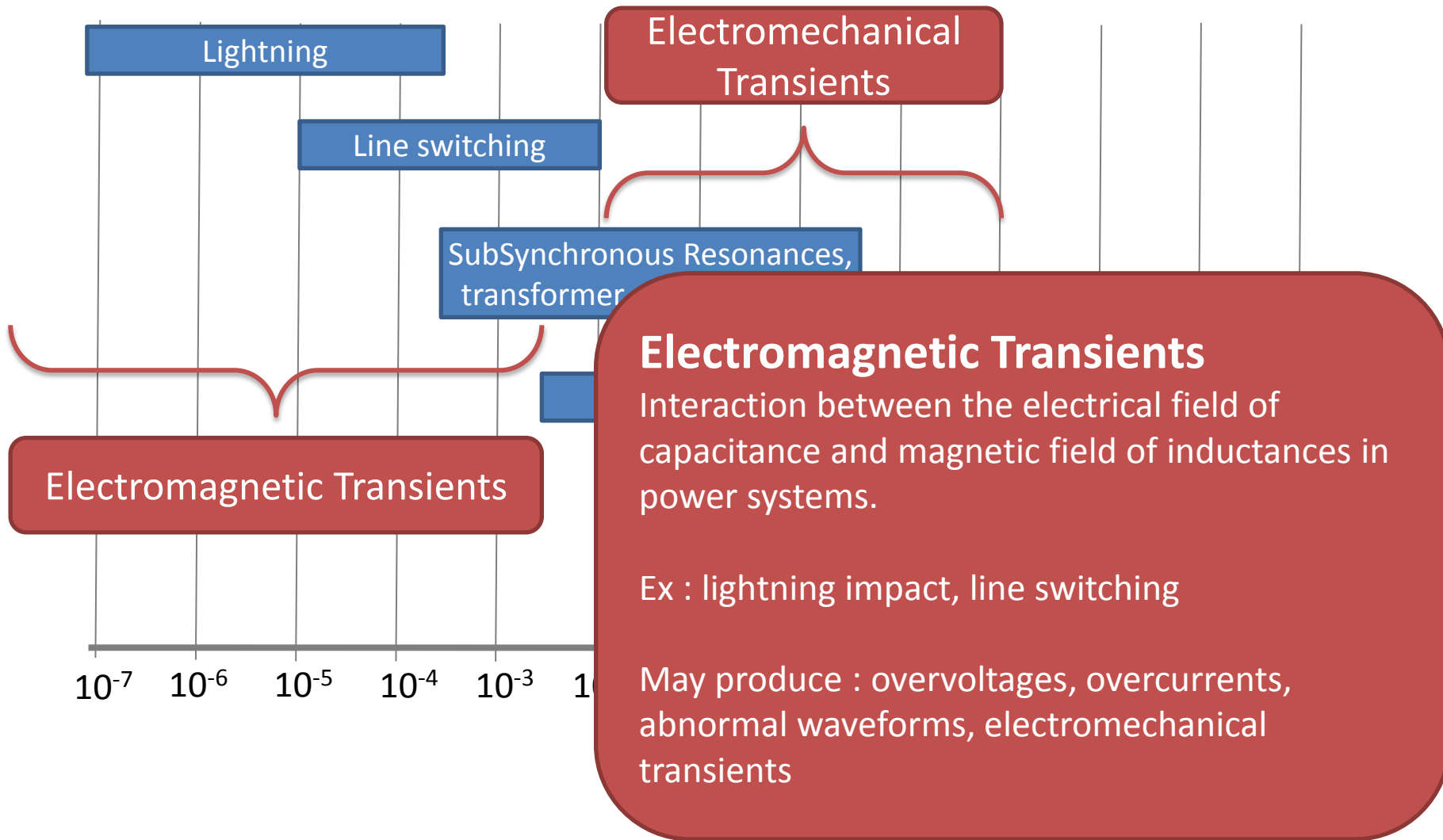
Current and new tools will need to perform simulations:

- Of complex hybrid model components and networks with very large number of continuous and discrete states.
- Models need to be shared, and **simulation results need to be consistent** across different tools and simulation platforms..

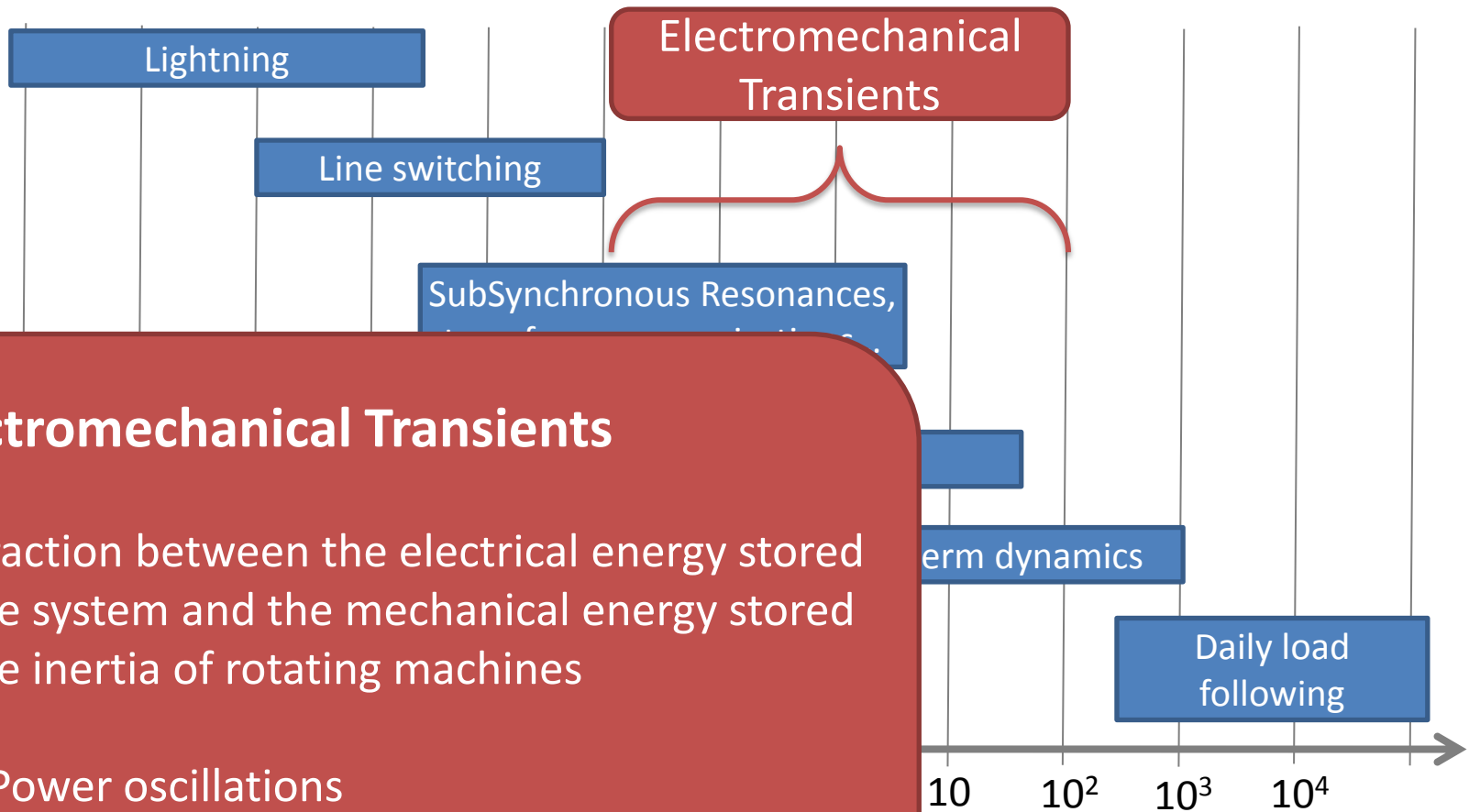
Power system *dynamics*



Power system *dynamics*



Power system *dynamics*



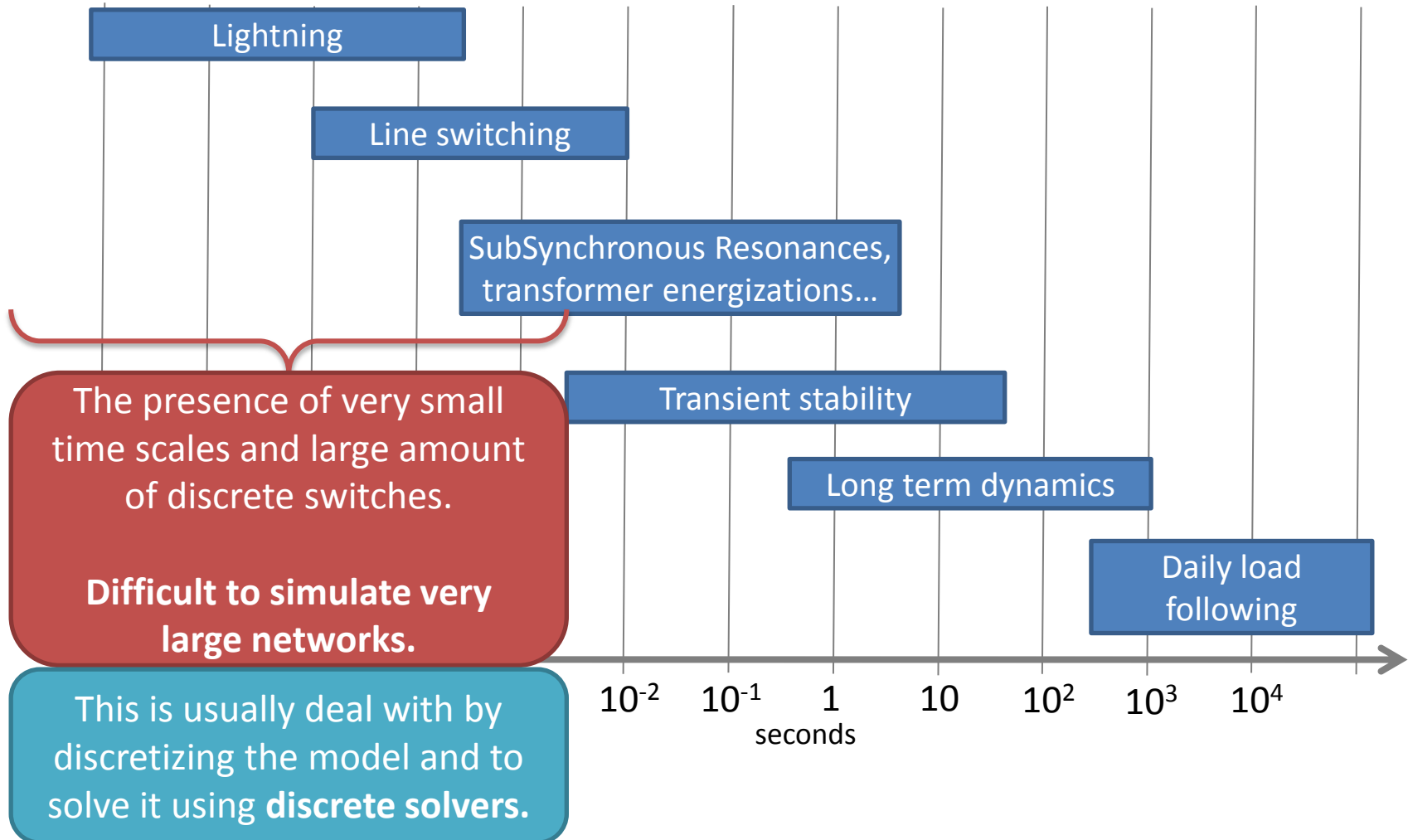
Electromechanical Transients

Interaction between the electrical energy stored in the system and the mechanical energy stored in the inertia of rotating machines

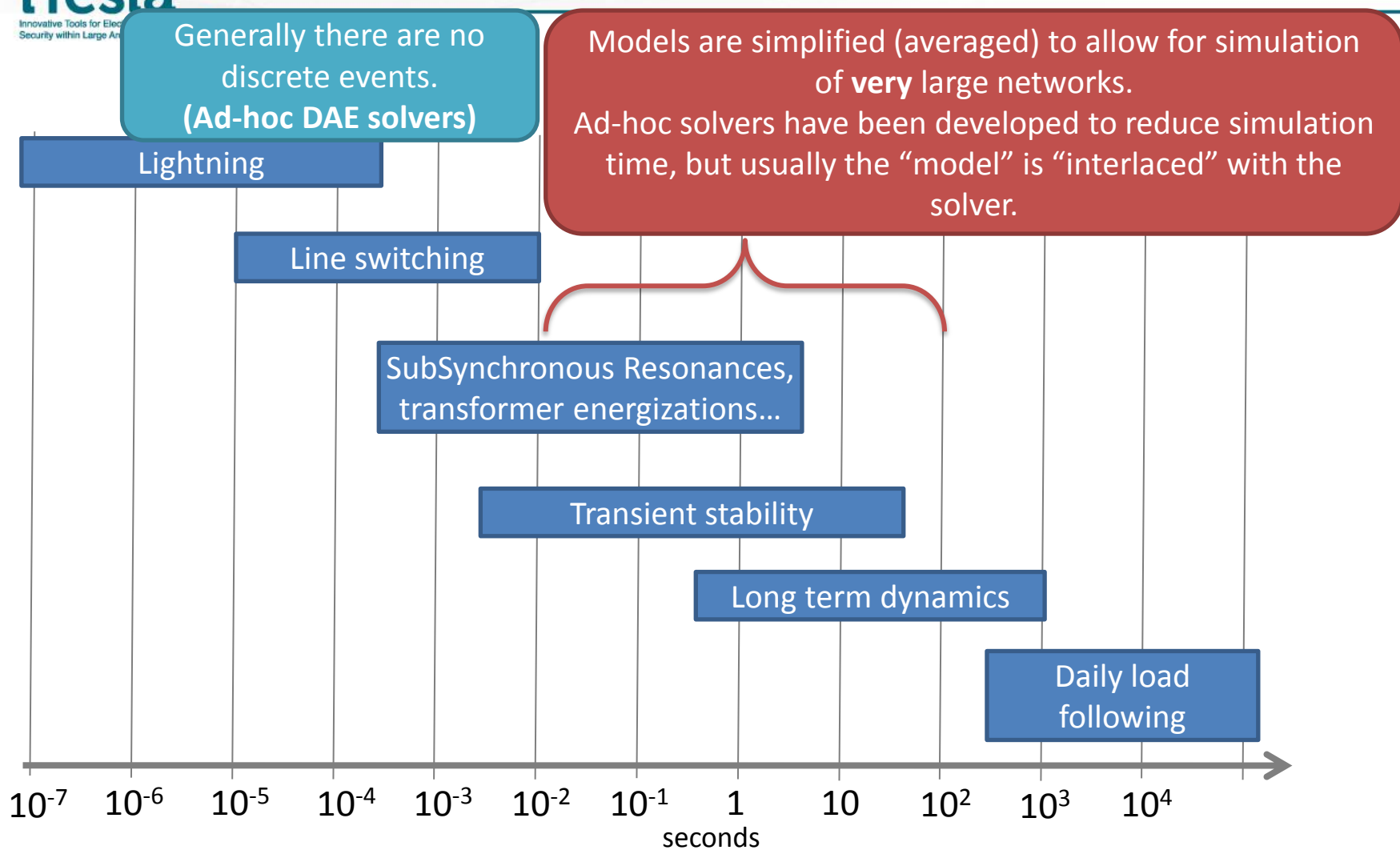
Ex : Power oscillations

May produce: system breakup.

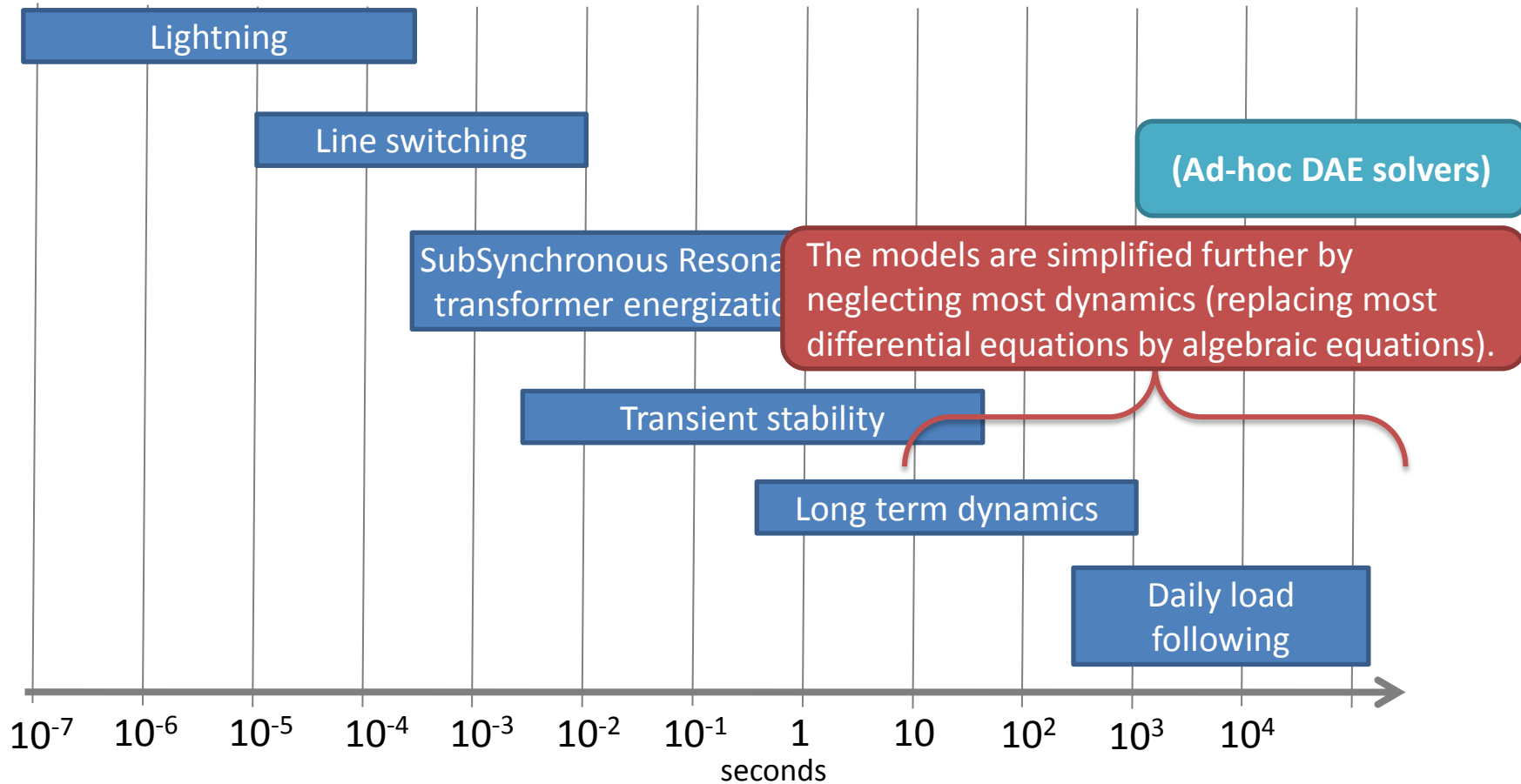
Power system dynamics challenges *for simulation*



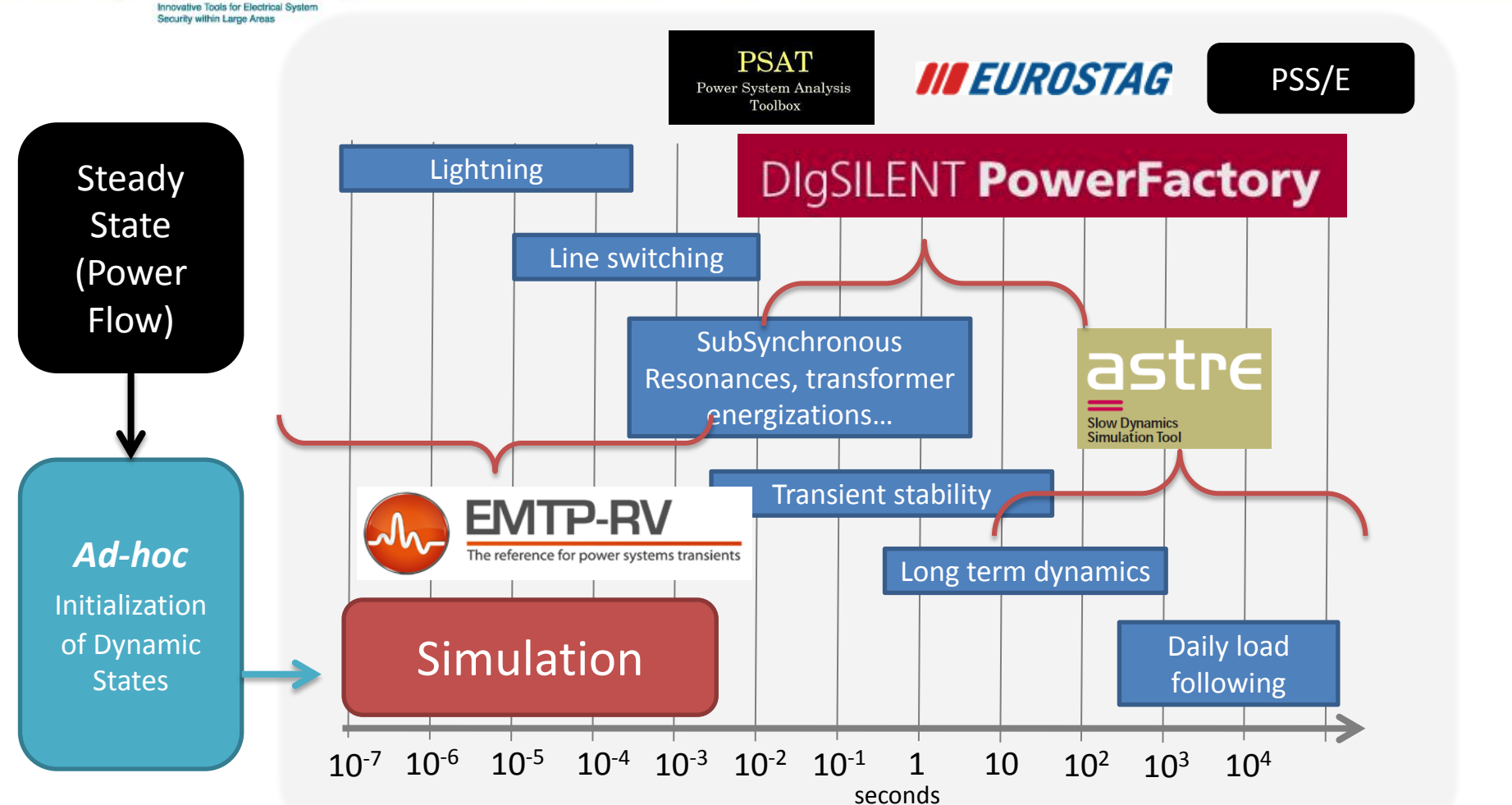
Power system dynamics challenges *for simulation*



Power system dynamics challenges *for simulation*



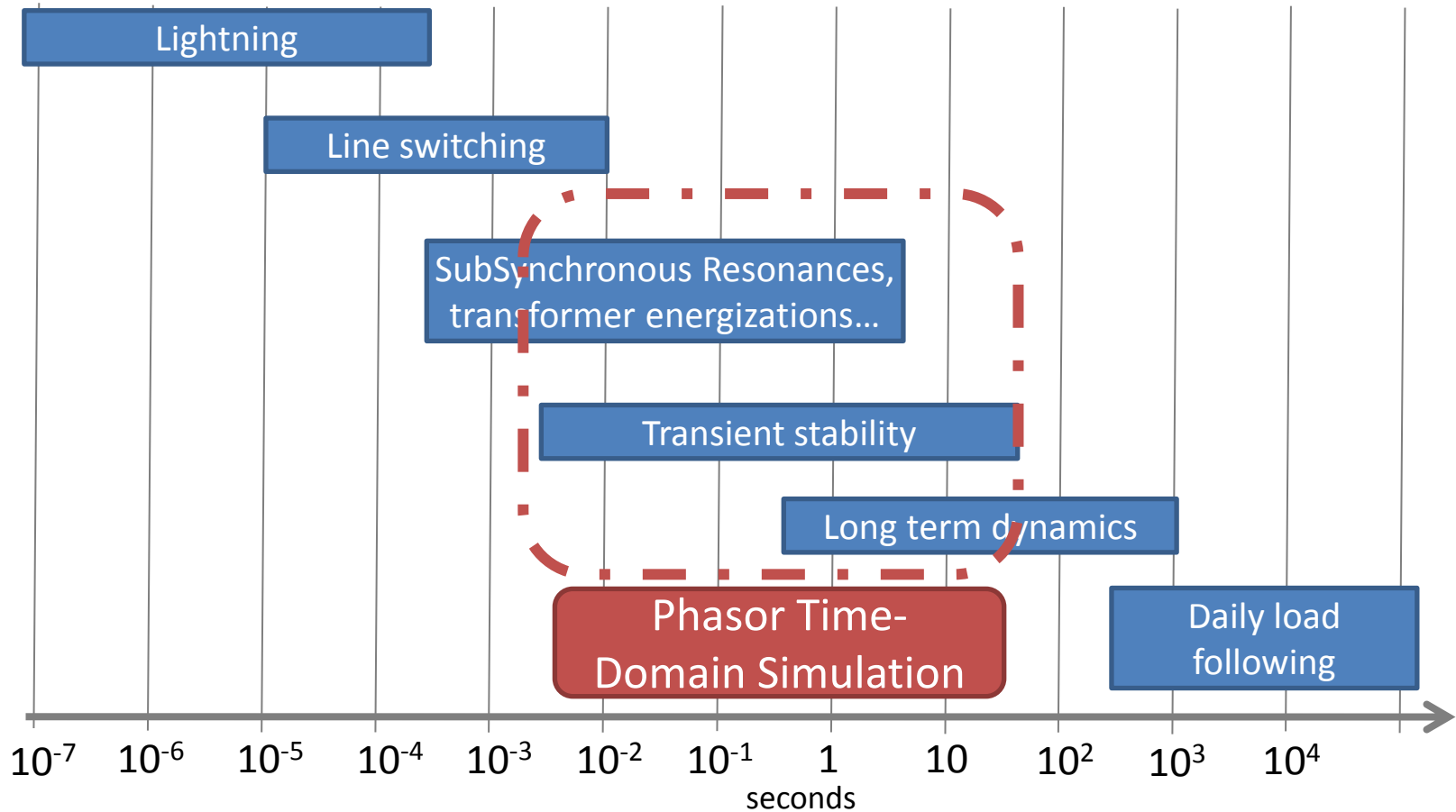
Power system phenomena and domain specific simulation tools



Broad range of time constants results in specific domain tools for simulation.
Non-exhaustive list. There exists other proprietary and few OSS .
Only two proprietary solutions for real-time simulation.

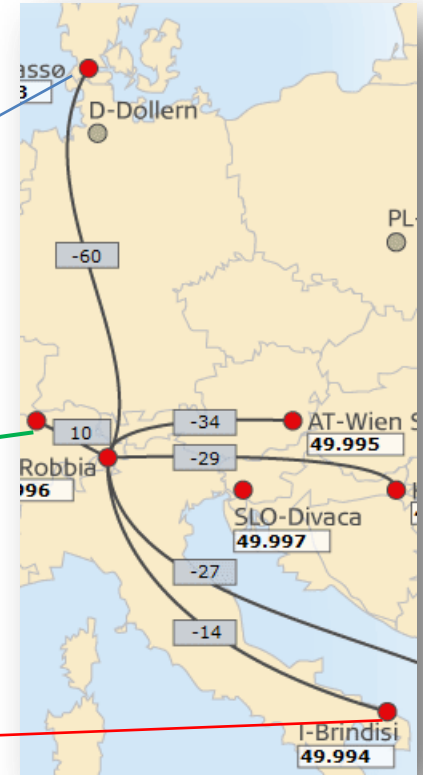
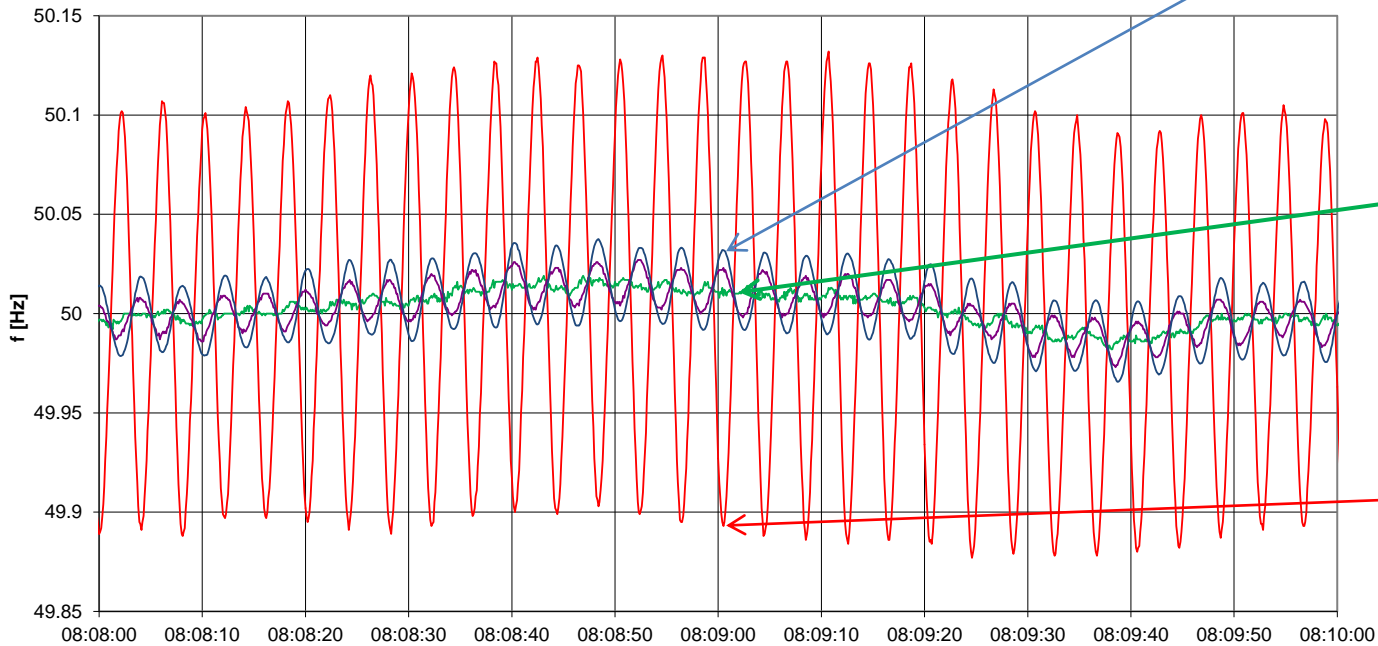
Power System Time-Scale Modeling

from this point on



Synchronzied Phasor Measurement Data

20110219_0755-0825



— Freq. Mettlen

— Freq. Brindisi

— Freq. Wien

— Freq. Kassoe



Power System Modeling

limitations, inconsistency and consequences

- Causal Modeling:
 - Most components are defined using causal block diagram definitions.
 - User defined modeling by scripting or GUIs is sometimes available (casual)
- Model sharing:
 - Parameters of block definitions are shared in a specific “data format”
 - For large systems, this requires “filters” for translation into the internal data format of each program
- Modeling inconsistency:
 - For (standardized casual) models there is no guarantee that the model definition is implemented “exactly” in the same way in different SW
 - User defined models and proprietary models can’t be represented without complete re-implementation in each platform
- Modeling limitations:
 - Most SWs make no difference between “model” and “solver”, and in many cases the model is somehow *implanted* within the solver
- Consequence:
 - It is almost impossible to have **the same model** in different simulation platforms.
 - This requires usually to re-implement the whole model from scratch (or parts of it) or to spend a lot of time “re-tuning” parameters.

Power System Modeling

limitations, inconsistency and consequences

- Causal Modeling:
 - Most components are defined using causal block diagram definitions.
 - User defined modeling by scripting or GUIs is sometimes available (casual)
- Model sharing:
 - Parameters of block definitions are shared in a specific “data format”
 - For large systems, this requires “filters” for translation into the internal data format of each program
- Modeling inconsistency:
 - For (standardized casual) models there is no guarantee that the model definition is implemented “exactly” in the same way in different SW
 - User defined models and proprietary models can’t be represented without complete re-implementation in each platform
- Modeling limitations:
 - Most SWs make no difference between “model” and “solver”, and in many cases the model is somehow *implanted* within the solver
- Consequence: This is very costly!
 - It is almost impossible to have **the same model** in different simulation platforms.
 - This requires usually to re-implement the whole model from scratch (or parts of it) or to spend a lot of time “re-tuning” parameters.

Power System Modeling

limitations, inconsistency and consequences

- Causal Modeling:
 - Most components are defined using causal block diagram definition
 - User defined modeling by scripting or GUIs is sometimes available
- Model sharing:
 - Parameters of block definitions are shared in a specific “data format”
 - For large systems, this requires “filters” for translation into the target program
- Modeling inconsistency:
 - For (standardized casual) models there is no guarantee that they will be implemented “exactly” in the same way in different SW
 - User defined models and proprietary models can’t be represented in a standard implementation in each platform
- Modeling limitations:
 - Most SWs make no difference between “model” and “solver”, and the model is somehow *implanted* within the solver

The Modelica language and Modelica Tools can help in avoiding all of these issues!

Consequence:

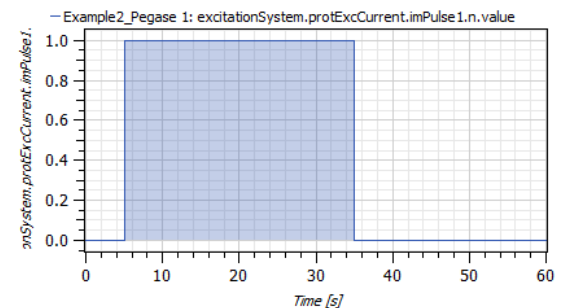
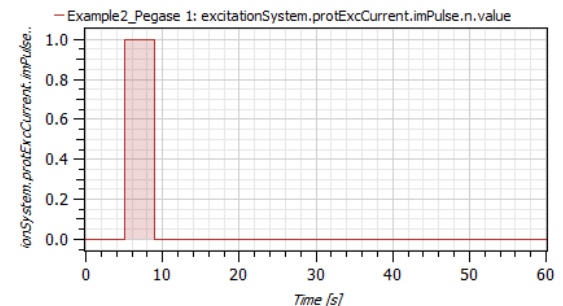
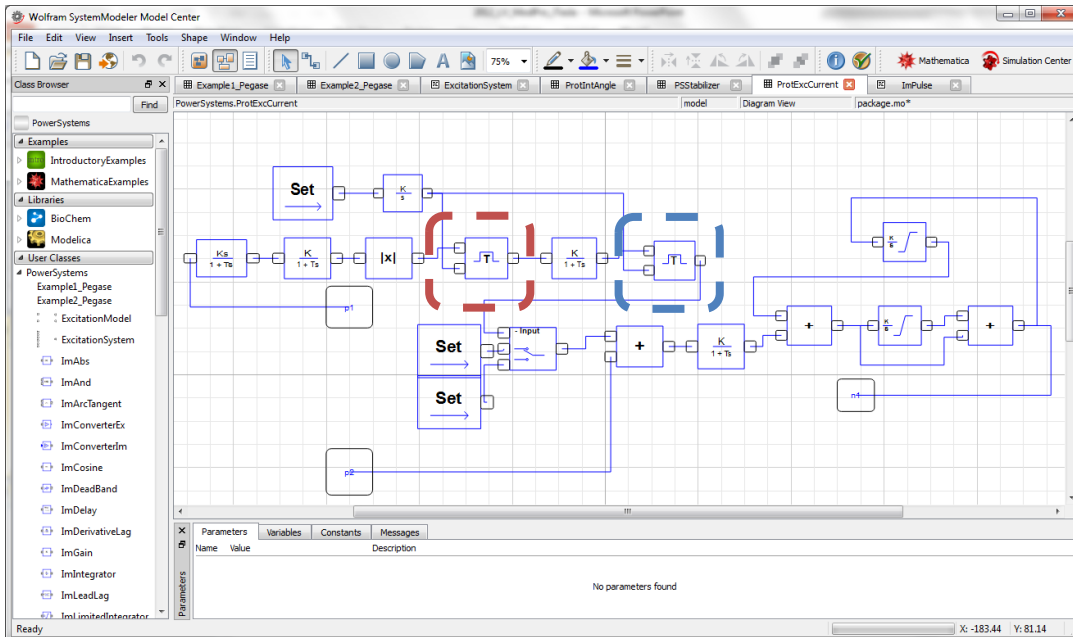
This is very costly!

- It is almost impossible to have **the same model** in different simulation platforms.
- This requires usually to re-implement the whole model from scratch (or parts of it) or to spend a lot of time “re-tuning” parameters.

Power System Modeling

limitations, inconsistency and consequences

- Modeling adequacy:
 - In general, the exclusive use of DAE solvers forces the representation of discrete devices using averaging or with other simplifications/tricks
 - This is an issue with complex controls and protections that require the handling of discrete events: this is not an issue for Modelica models and tools!



IEC CIM

- A set of standards in enable system integration and information exchange based on a common information model
- The CIM standards are based on a Unified Modeling Language (UML) based information model representing real-world objects and information entities exchanged within the value chain of the electric power industry
 - Provides common semantics for all information exchanges
- However:
 - Modeling of dynamics is still in progress (the CIM effort was started in 1996 in IEC) and there are many issues to resolve.
 - The modeling approach selected will use “IEEE Standard” models which are **causal** block-oriented model definitions
 - **An equation-based definition of model dynamics could be more beneficial for model consistency: Modelica is a tangible choice**

Unambiguous

Power System Modeling and Simulation

ambiguous

adjective

an ambiguous explanation: EQUIVOCAL, ambivalent, open to debate/argument, arguable, debatable; obscure, unclear, imprecise, vague, abstruse, doubtful, dubious, uncertain. ANTONYMS clear.

- Modeling and simulation should not be ambiguous: it should be consistent across different simulation platforms.
- For unambiguous modeling, model sharing and simulation, Modelica and Modelica Tools can be used.
- We illustrate how we use Modelica and Modelica Tools towards this goal:
 - Building blocks for power system simulation: iTesla PS Modelica Library
 - Consistent model sharing across different modeling tools
 - Adequate Modeling (... a case for controls and protections)
 - Large scale model simulation (... and initialization issues)
 - Using FMUs for model sharing and exploiting generic solvers

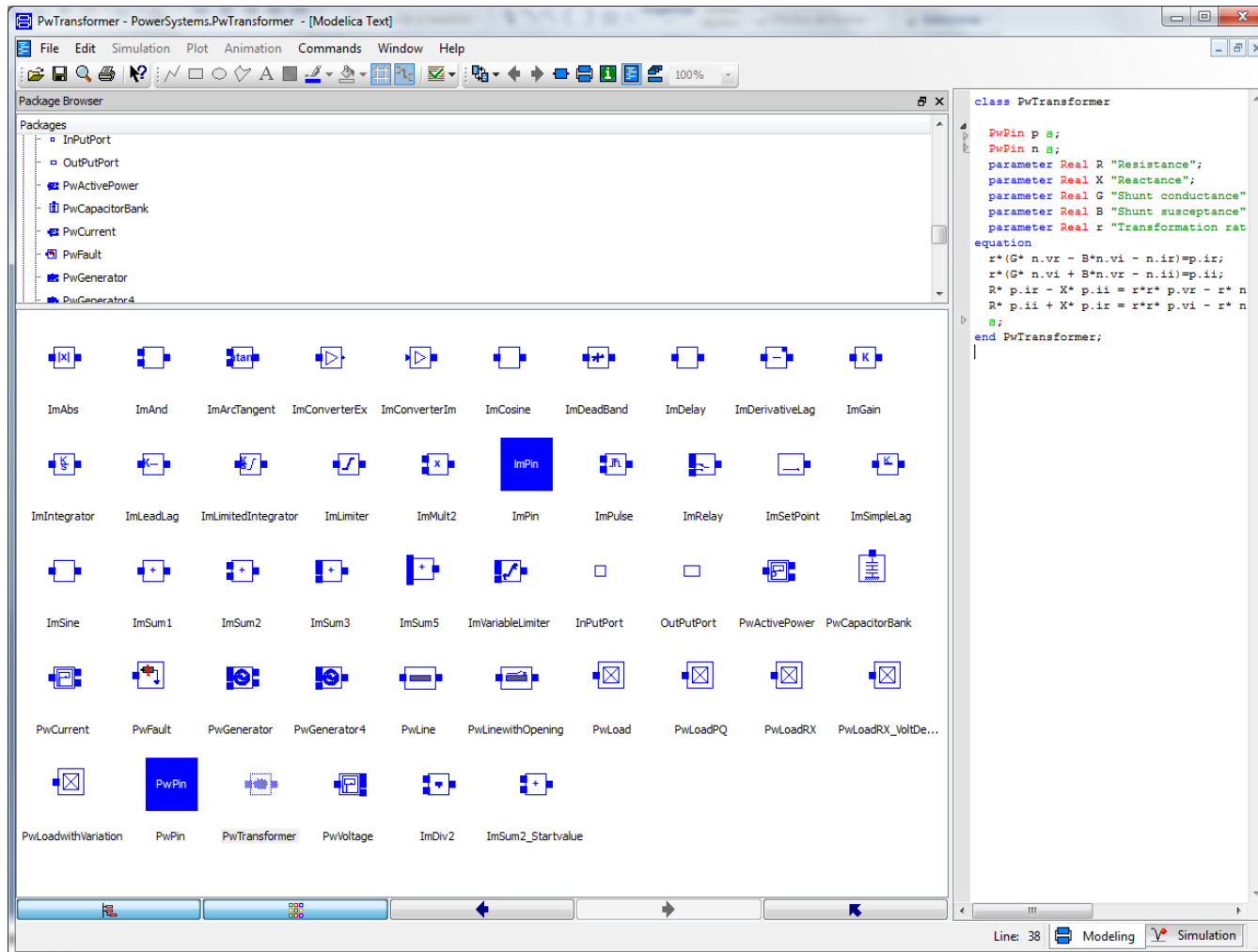


iTesla Power Systems

Modelica Library

- Power Systems Library:
 - A Power Systems library developed during the Pegase project in Scilab/Xcos was converted to Dymola (manually).
 - The library has also been tested (and modified) for OpenModelica and SystemModeler.
 - Components and systems are validated against Eurostag's results.
- New components and time events are being added to this library in order to simulate new systems.
 - Efforts will be put in replicating all of Eurostag's and PSAT models in the library, and adding new models of RES and other power electronic controlled devices
 - Automatic translator from domain specific tools to Modelica will use this library's classes.
- New regulator macroblocks and medium size power systems have been built and simulated in Dymola, OpenModelica and SystemModeler.

- Dymola:



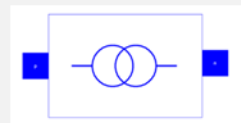
class PwLine

```
parameter Real R "Resistance";
parameter Real X "Reactance";
parameter Real G "Shunt half conductance";
parameter Real B "Shunt half susceptance";
```

equation

```
R*(n.ir-G*n.vr+B*n.vi)-X*(n.ii-B*n.vr-
G*n.vi)=(n.vr-p.vr);
R*(n.ii-B*n.vr-G*n.vi)+X*(n.ir-
G*n.vr+B*n.vi)=(n.vi-p.vi);
R*(p.ir-G*p.vr+B*p.vi)-X*(p.ii-B*p.vr-
G*p.vi)=(p.vr-n.vr);
R*(p.ii-B*p.vr-G*p.vi)+X*(p.ir-
G*p.vr+B*p.vi)=(p.vi-n.vi);
```

end PwLine;



class PwTransformer

```
PwPin p ;
PwPin n ;
parameter Real R "Resistance";
parameter Real X "Reactance";
parameter Real G "Shunt conductance";
parameter Real B "Shunt susceptance";
parameter Real r "Transformation ratio";
```

equation

```
r*(G*n.vr-B*n.vi-n.ir)=p.ir;
r*(G*n.vi+B*n.vr-n.ii)=p.ii;
R*p.ir-X*p.ii=r*r*p.vr-r*n.vr;
R*p.ii+X*p.ir=r*r*p.vi-r*n.vi;
```

end PwTransformer;

New components in iTesla

- Here are some of the new components and time events built in Modelica for the iTesla Project:
 - Capacitor Bank with Modification.
 - Line opening, Load modification

```
class PwCapacitorBank

  PwPin p
  parameter Real nsteps "number of steps";
  parameter Real Go "active losses (p.u.) in each element";
  parameter Real Bo "reactive power (p.u.) in each element";
  parameter Real t1 "time for Bank Modification";
  parameter Real nmod "number of step to swigh on/off (+/-)";
  Real G;
  Real B;
  Real nt;

equation
  if (time > t1) then
    nt = nsteps + nmod;
  else
    nt = nsteps;
  end if;
  G=nt*Go;
  B=nt*Bo;
  p.vr = (p.ir*G + p.ii*B)/(G*G + B*B);
  p.vi = (-p.ir*B + p.ii*G)/(G*G + B*B);

end PwCapacitorBank;
```



```
class PwLoadwithVariation
```

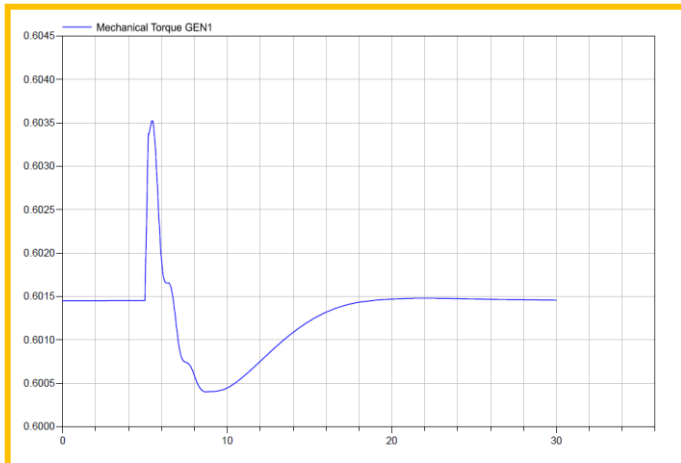
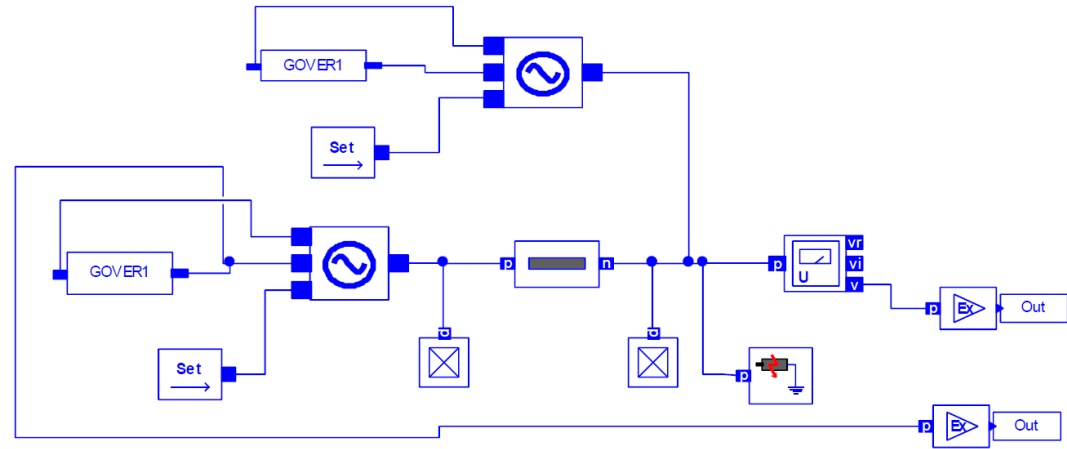
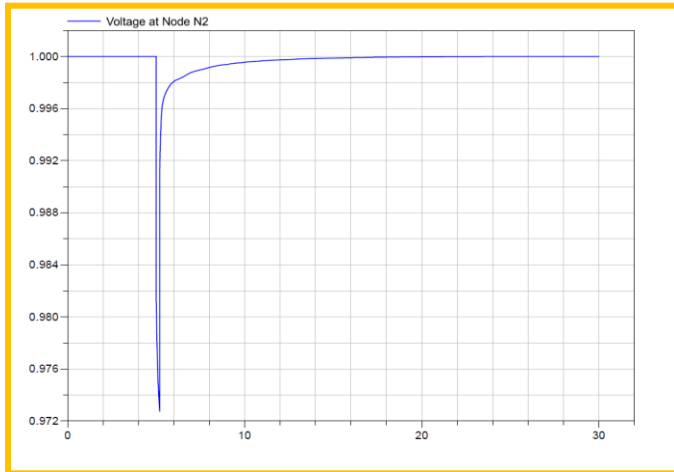
```
  PowerSystems.PwPin p
  parameter Real Vo_real "Initial voltage at node in p.u.";
  parameter Real Vo_img "Initial voltage at node in p.u.";
  parameter Real Po "Initial Active Power in p.u.";
  parameter Real Qo "Initial Reactive Power in p.u.";
  parameter Real t1 "Time of Load variation";
  parameter Real P2 "Active load variation";
  parameter Real Q2 "Reactive load variation";
  Real Vo;
  Real P;
  Real Q;
  Real R;
  Real X;
  Real a;
```

```
equation
```

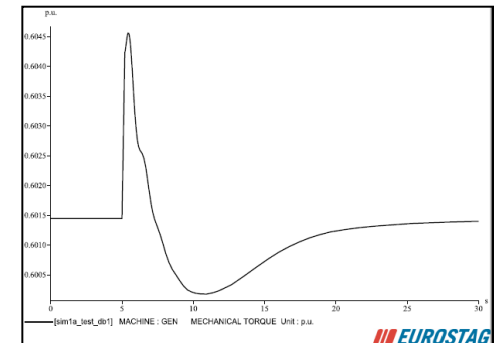
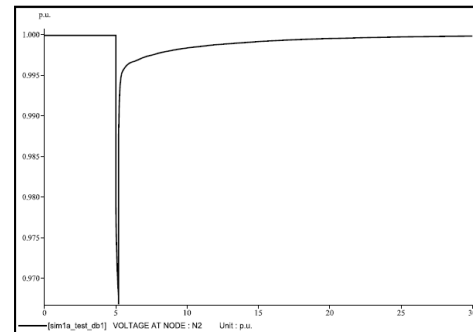
```
  Vo = sqrt(Vo_real*Vo_real + Vo_img*Vo_img);
  if (time > t1) then
    P = Po + P2;
    Q = Qo + Q2;
  else
    P = Po;
    Q = Qo;
  end if;
  a = P/Q;
  R = (abs(Vo)*abs(Vo)/P)*(a*a/(1+a*a));
  X = R/a;
  p.vr=R*p.ir-X*p.ii;
  p.vi=X*p.ir+R*p.ii;
```

```
end PwLoadwithVariation;
```

- Example 1 [from Pegase] in Dymola

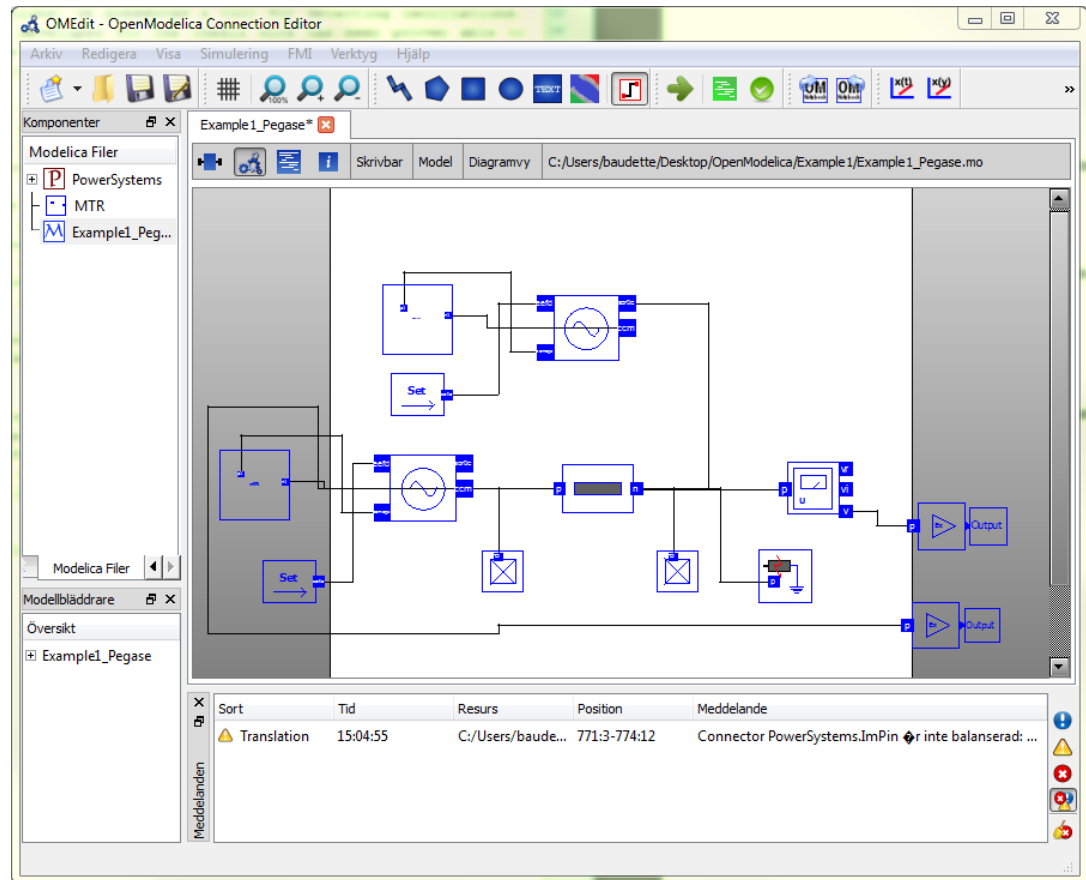
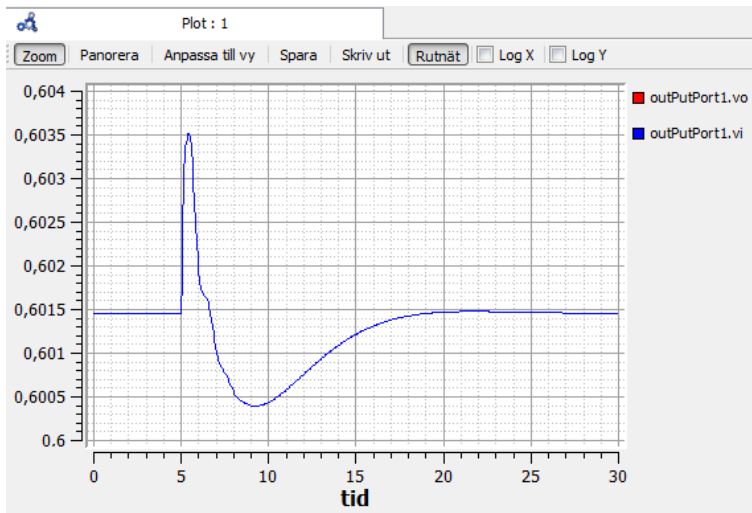
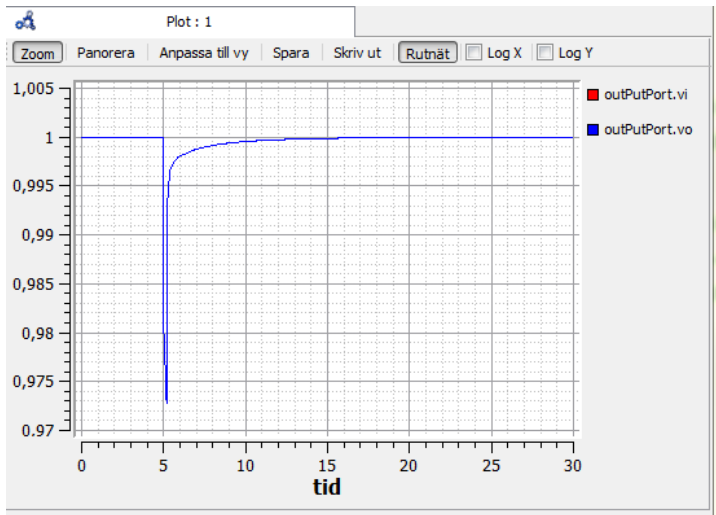


Eurotag



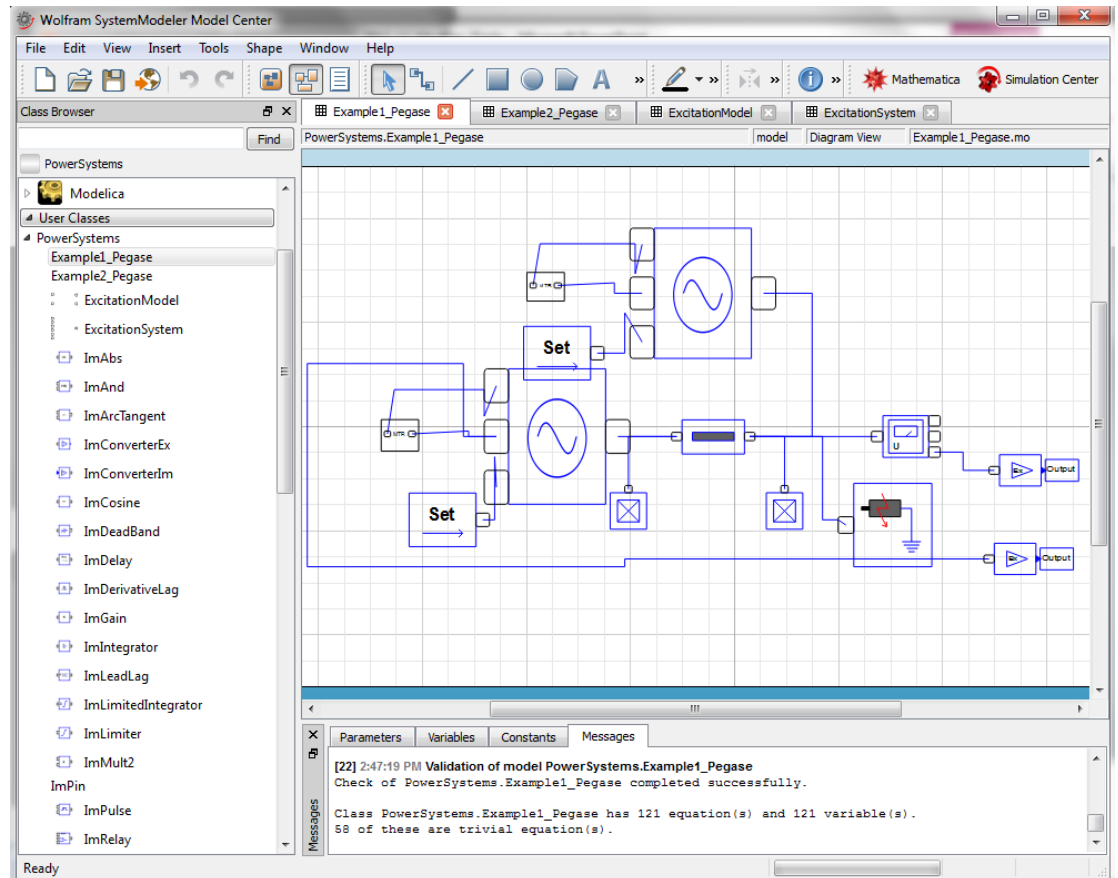
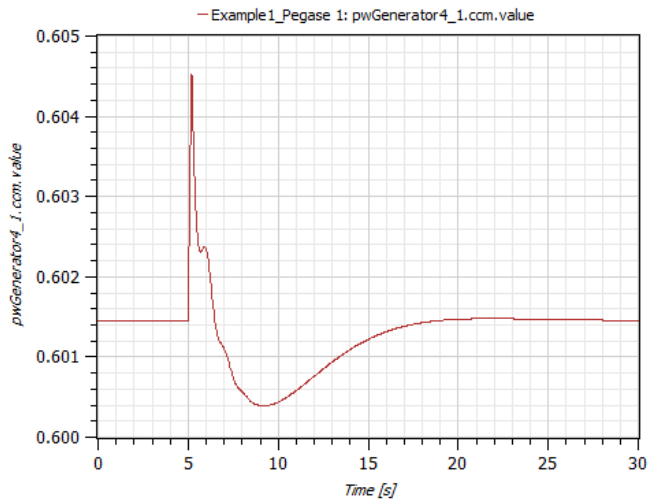
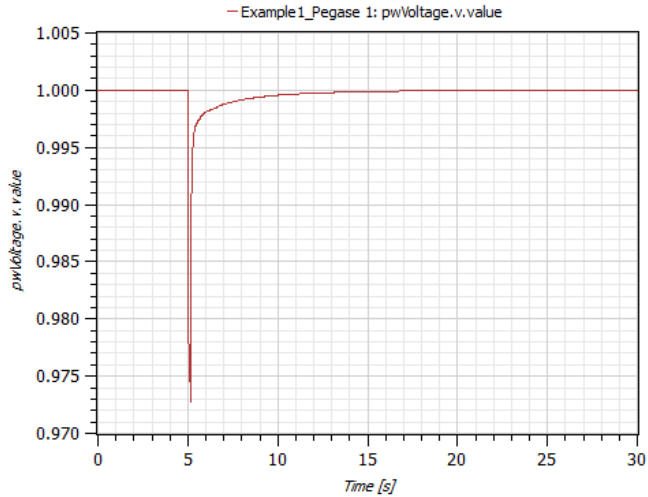
Example 1

OpenModelica



Example 1

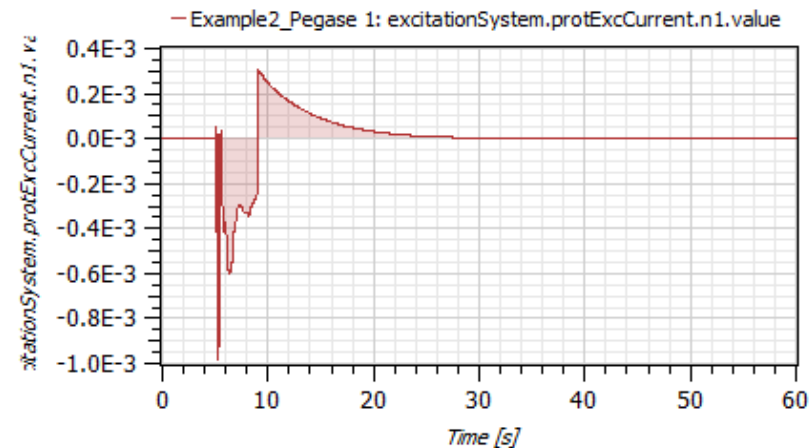
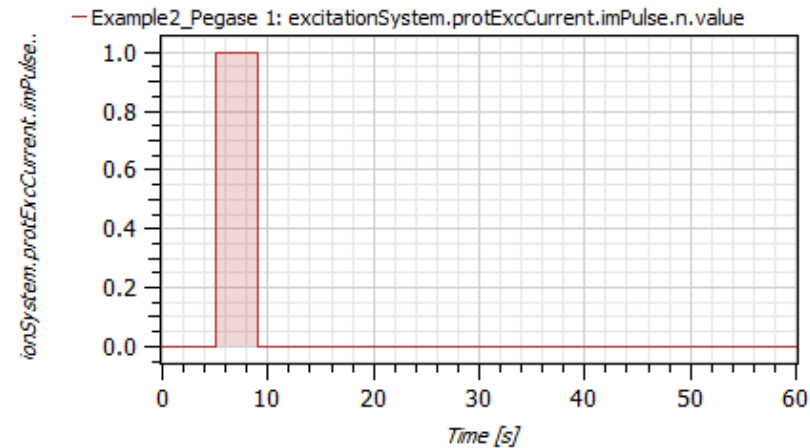
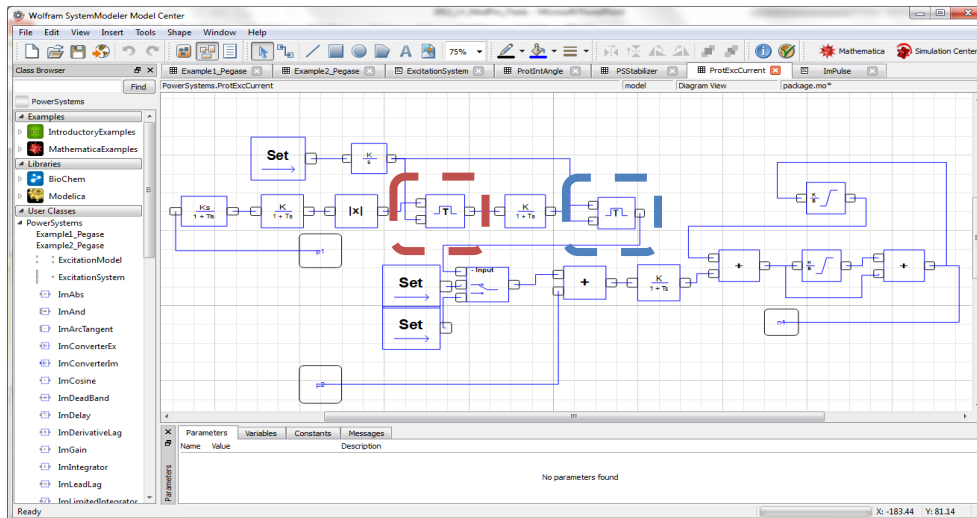
System Modeler



Power System

Modeling Adequacy

- Modeling adequacy: “Example 2”
 - Complex controls and protections may require the handling of discrete events
 - This case illustrates the special protection used in a generator’s excitation current
 - Two discrete time delays are modeled explicitly and sequentially

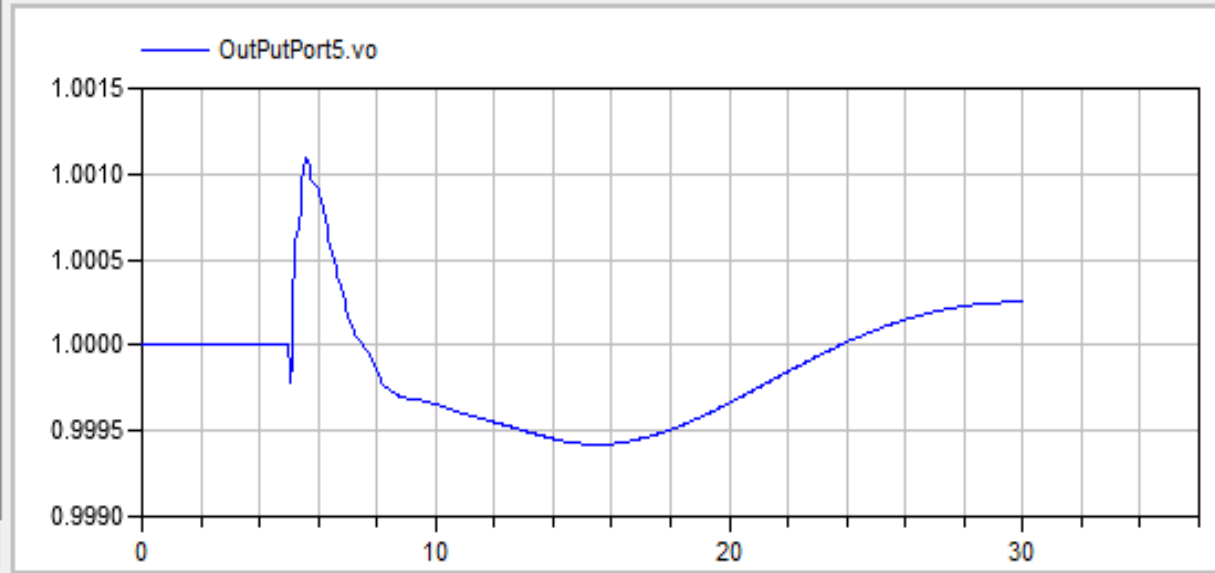
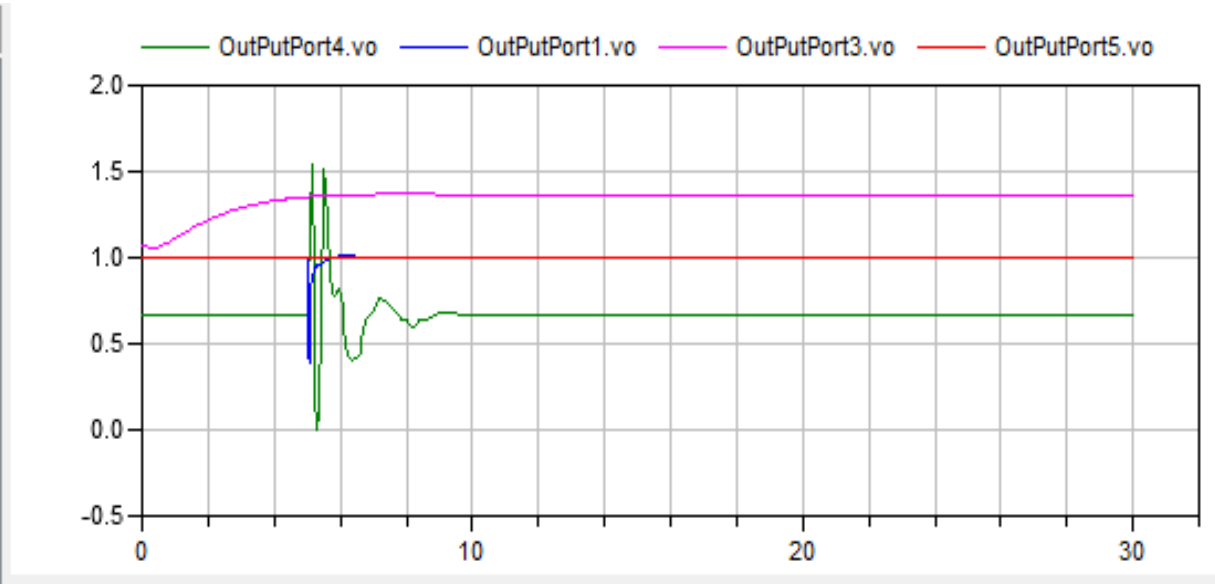


(Larger) Model Simulation

Variable Browser

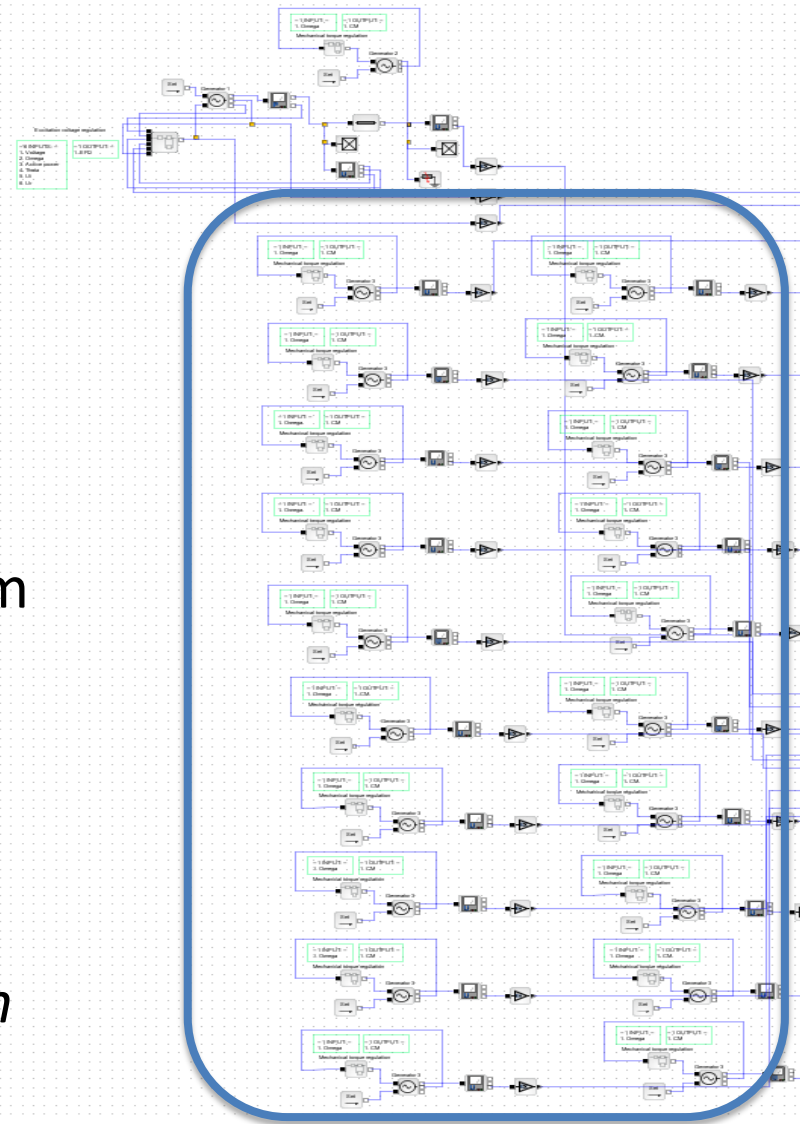
Variables	Values	Unit
ImConverterEx23		
OutPutPort1		
<input type="checkbox"/> vo		
<input type="checkbox"/> vi		
OutPutPort2		
OutPutPort3		
<input type="checkbox"/> vo		
<input type="checkbox"/> vi		
OutPutPort4		
<input type="checkbox"/> vo		
<input type="checkbox"/> vi		
OutPutPort5		
<input checked="" type="checkbox"/> vo		
<input type="checkbox"/> vi		
OutPutPort6		
OutPutPort7		
OutPutPort8		
OutPutPort9		
OutPutPort10		
OutPutPort11		
OutPutPort12		
OutPutPort13		
OutPutPort14		
OutPutPort15		
OutPutPort16		

Advanced



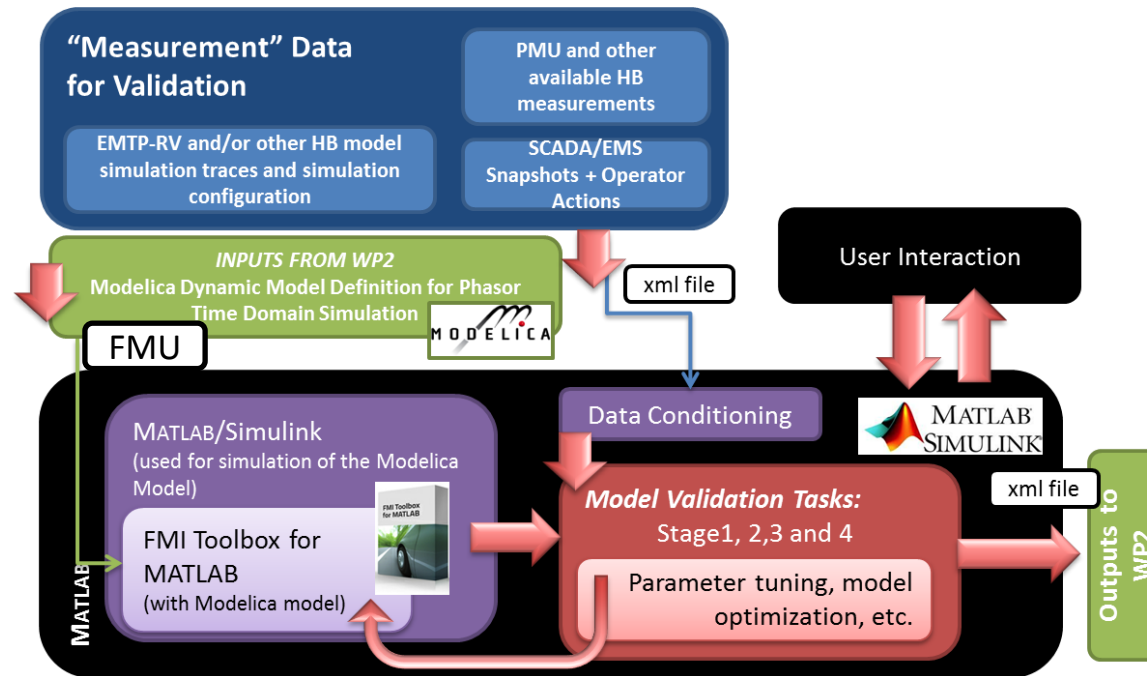
(Larger) Model Simulation

- This figure shows a larger-scale power system network extending “Example2” with 20 more generators.
 - Total of 22 generators + controls *approx. 440 continuous states + approx. 100 discrete states*
- ***Initialization:***
- In the previous the initialization of dynamic states has been extracted from Eurostag.
- Default Dymola initialization is used here. Not possible to initialize/simulate using Scilab/Xcos.
- *Not clear how to deal with initialization for power system models.*



FMUs for Model Sharing and simulation with generic solvers

- Why?:
 - The iTesla project has adopted Modelica as its toolbox internal language for dynamic model description
 - There is a WP on model validation.



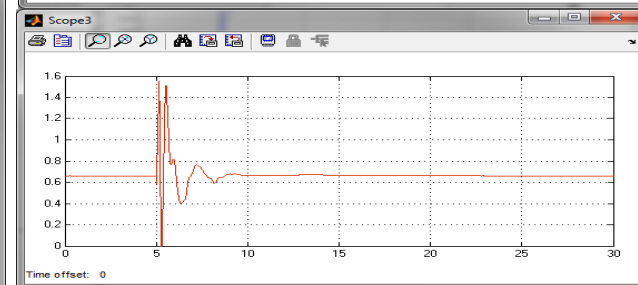
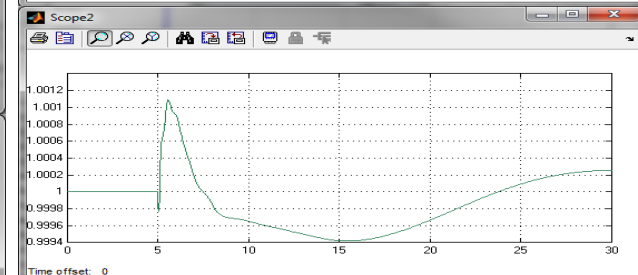
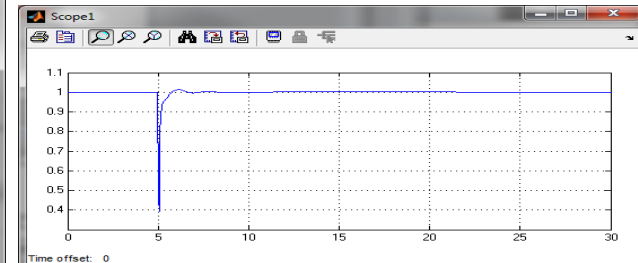
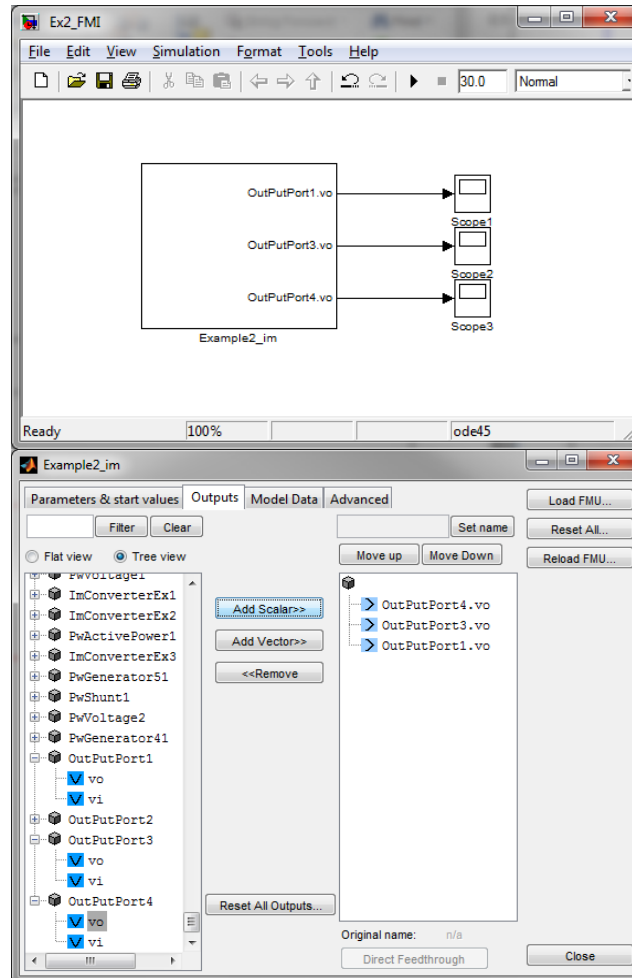
FMUs for Model Sharing

- Some TSOs are not allowed to provide explicit information of models or their description (e.g. wind turbines)
- FMUs could help – two options:
 - FMUs of specific devices are generated, and then incorporated into the overall model.
 - FMUs of a complete model can be generated and shared without revealing the model's internal structure/equations
- We explore the second option.

Sharing FMUs with

MATLAB/Simulink and the FMI Toolbox

- The model validation architecture needs to exploit the sys id. tools in Matlab.
- FMUs generated from Dymola.
- FMUs allow for model simulation in Matlab for self-contained integration of prototype model validation tool.



FMUs for simulation

with generic solvers and in generic purpose environments

- The use of generic solvers is attractive for performing large numbers of simulations of large-scale power system models.
- It is also attractive not to depend on Matlab for the model validation tools
- This example shows the use of Assimulo solvers within the JModelica.org framework
- The JModelica.org framework for dynamic optimization is attractive for this kind of application (model validation and correction)

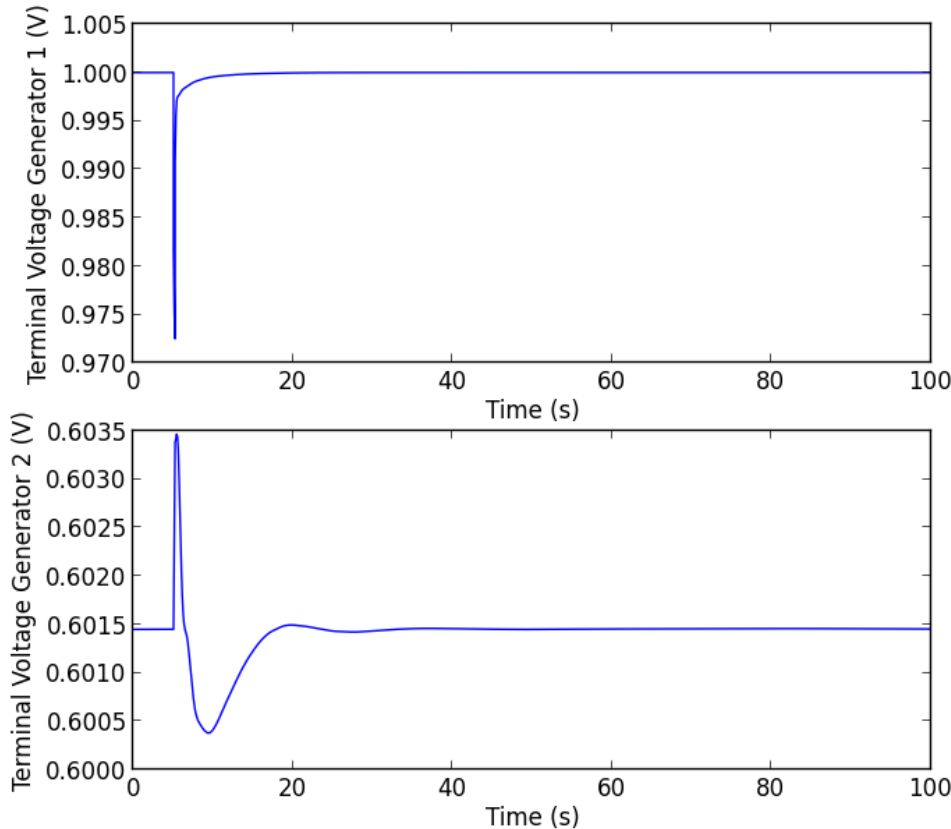
JMODELICA.ORG

```
8 import os as O
9
10 import pylab as P
11 import numpy as N
12
13 from pyfmi import FMUModel
14
15 curr_dir = O.path.dirname(O.path.abspath(__file__));
16 path_to_fmus = O.path.join(curr_dir, 'files', 'FMUs')
17
18 def run_demo(with_plots=True):
19     """
20     Demonstrates how to use JModelica.org for simulation of FM
21     """
22
23     fmu_name = O.path.join(path_to_fmus, 'Example1_0Pegase.fmu')
24     # fmu_name = 'C:\\JModelica.org-1.8.1\\Python\\pyfmi\\examp
25     model = FMUModel(fmu_name)
26
27     opts = model.simulate_options()
28     # opts['Cvode_options']['discr'] = 'Adams'
29     # opts['solver'] = 'Cvode' # Change the solver
30     opts['Cvode_options']['atol'] = 1.0e-3 # specific toleranc
31     opts['initialize'] = True # initialize the model
32     opts['ncp'] = 1000 #changing the number of communication p
33
34     res = model.simulate(options=opts, final_time=100.)
35     print("FMU succesfully simulated")
36
```

FMUs for simulation

with generic solvers and in generic purpose environments

FMI of Example 1 Pegase using JModelica



JMODELICA.ORG

```
Python 1 x pegase_ex01.py x
A discontinuity occurred at t = 5.000000e+00.
A discontinuity occurred at t = 5.200000e+00.
Final Run Statistics: ---

Number of Steps                : 52
Number of Function Evaluations : 74
Number of Jacobian Evaluations : 3
Number of F-Eval During Jac-Eval : 48
Number of Root Evaluations     : 55
Number of Error Test Failures  : 0
Number of Newton Iterations    : 62
Number of Newton Convergence Failures : 0
Number of State-Events        : 0

Solver options:

Solver                : CVode
Linear Multistep Method : BDF
Nonlinear Solver      : Newton
Maxord                : 5
Tolerances (absolute) : [ 0.001 0.001 0.001
0.001 0.001 0.001 0.001 0.001 0.001]
Tolerances (relative) : 0.0001

Simulation interval   : 0.0 - 100.0 seconds.
Elapsed simulation time: 0.4582016 seconds.
FMU succesfully simulated
```

Results & Lessons Learned

No.1:

- A Modelica library has been developed and tested in different Modelica tools.
- Modelica allows for unambiguous model sharing across different simulation software
 - This is natural thanks to the Modelica language

No.2:

- Modeling of complex power system controls and protections can take into account discrete events
 - Flexibility of modeling language and solvers to handle discrete events.

No.3:

- It is possible to simulate large models (although not very large) of power systems.
 - Proper initialization will allow to determine the suitability for real-life networks
 - Automatic conversion from domain specific tool will be required (too time consuming to model from scratch)

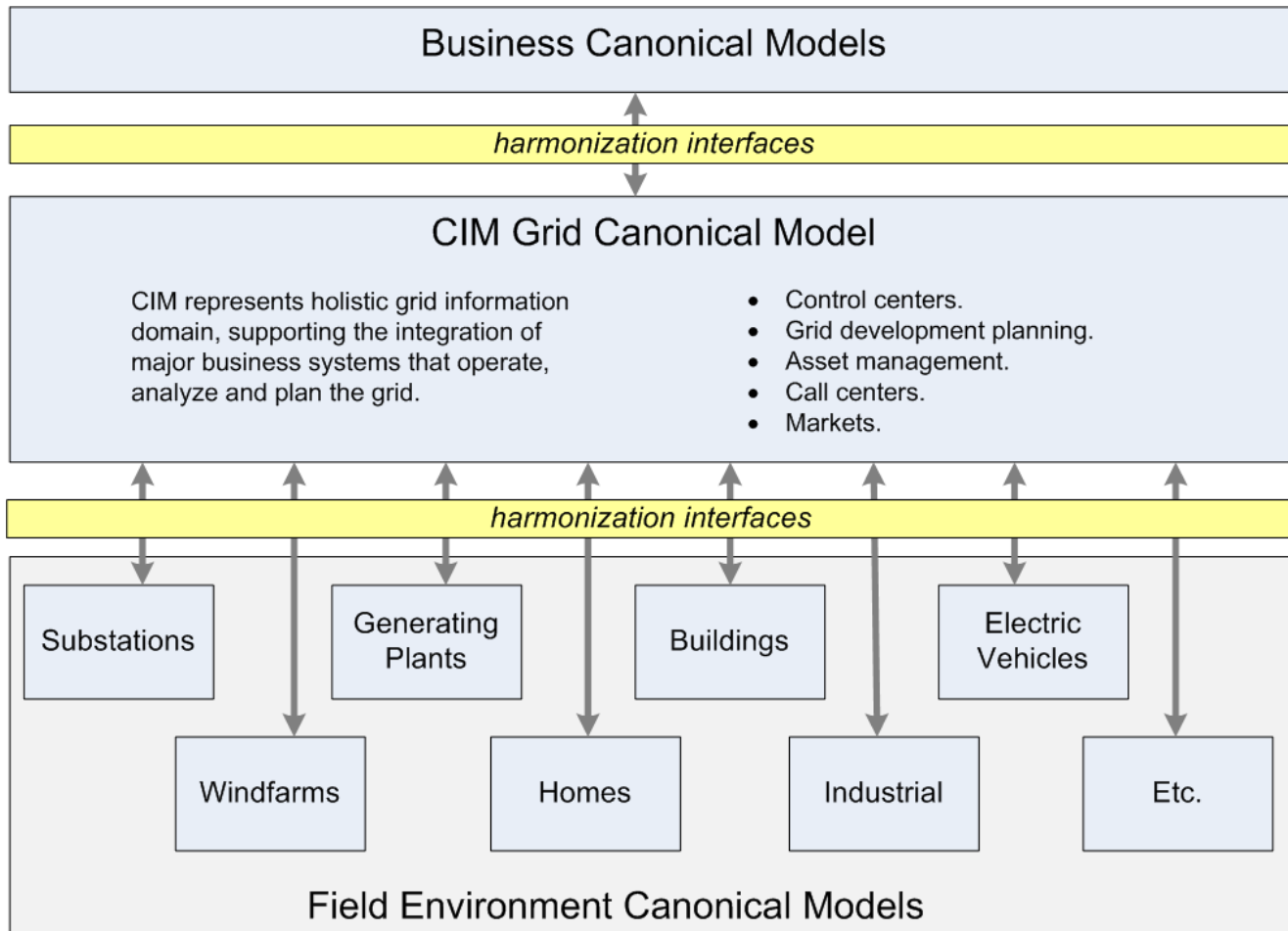
No.4:

- FMUs allow to use general purpose tools for specific power system applications (model validation)
- FMUs allow sharing of models without revealing the internal model definition/structure (useful for manufacturer specific models).

Opportunities for Modelica in Power Systems

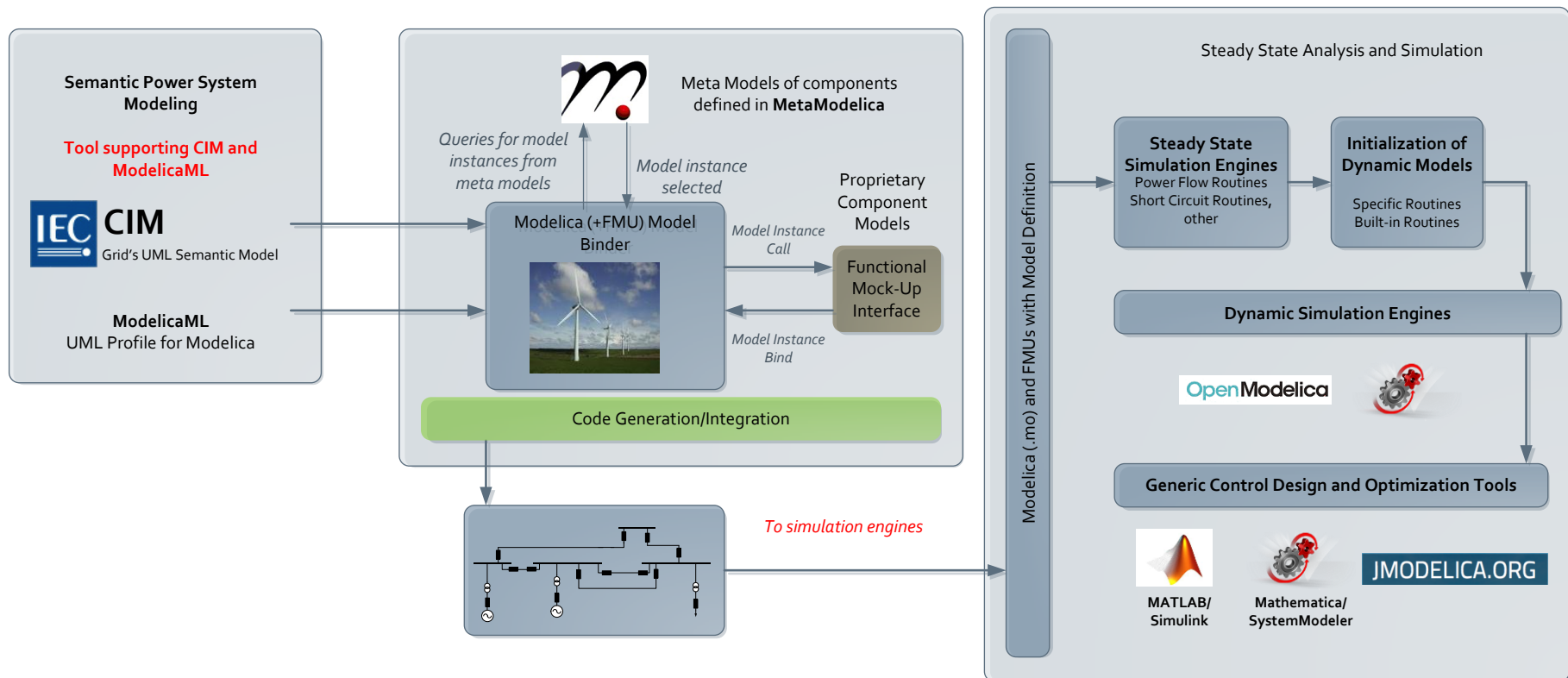
- Main issue: guarantee security of supply when facing
 - increased amount of “variable” RES
 - limited transmission capacity coupled to growth and dependency on demand
 - cyber-security concerns/threats
- These and many other challenges will require:
 - New/different set of tools [Feynman]
 - New/different ways of thinking and dealing with new/old problems [teach an old dog new tricks]
- Two key areas where Modelica and Modelica Tools can help [immediately]:
 - Unified CIM-compliant unified modeling of power grids (in multiple time scales)
 - Smart Grids: cyber-physical modeling of power systems and computer systems

CIM is the (future) *Grid Model*

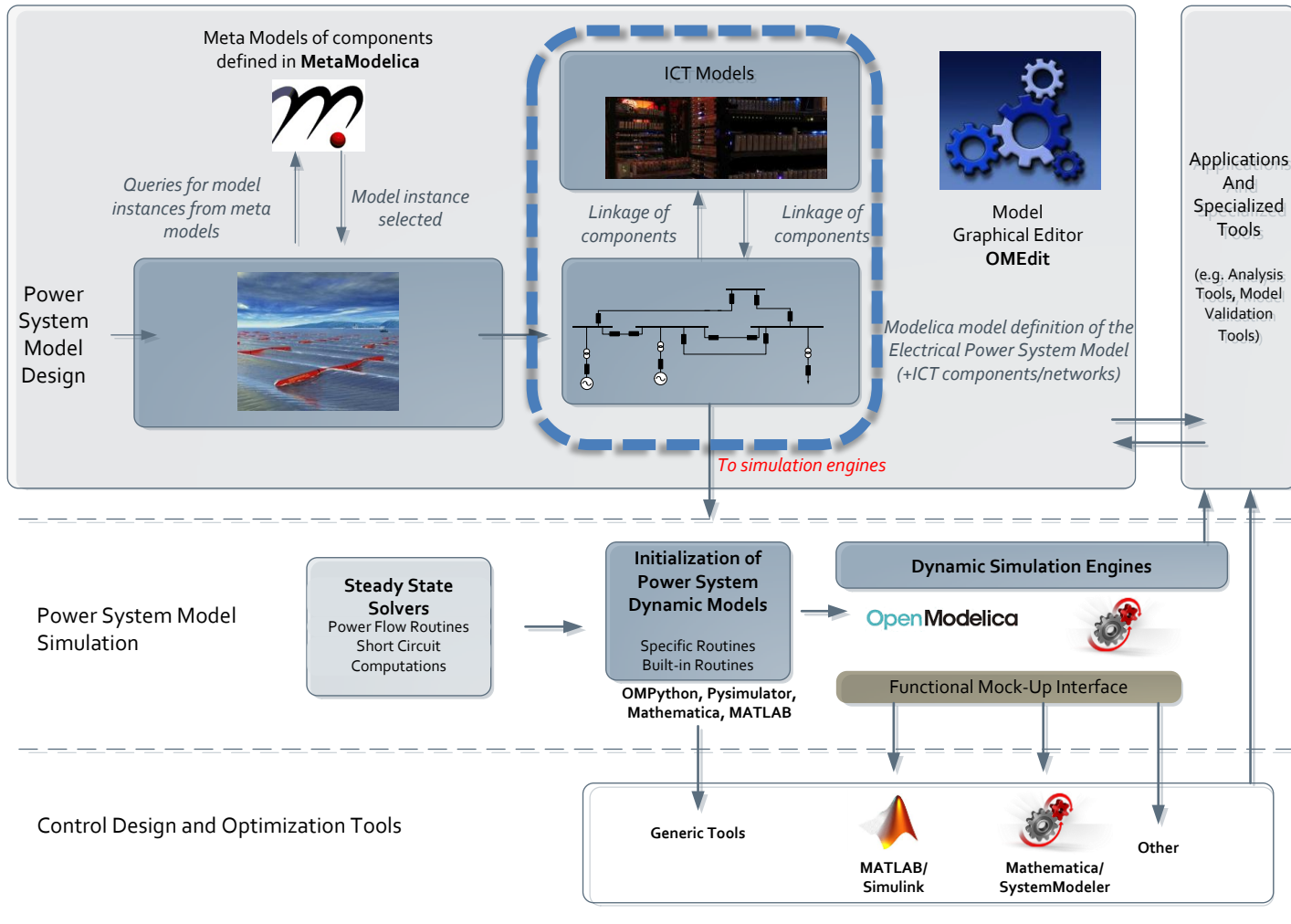


Perspectives for Unified Grid Modeling

Multiple Time-Scale Unified Power System Modeling



Perspectives on Modelica *for Smart Grids*



Thank you!

Questions?

Looking for people to join these efforts!

- PhD Position on “CIM-Compliant Unified Modeling, Simulation and Estimation of Smart Transmission Grids”
- Please see the webpage: <http://www.kth.se/en/om/work-at-kth/vacancies>, around Feb. 11th.
- Email me: luigiv@kth.se