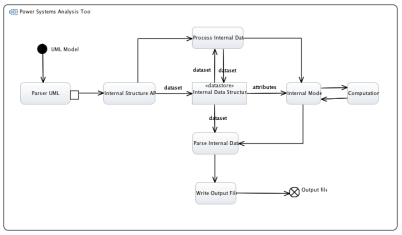
Meta-Modeling and Model Transformations for Power System Simulation using Model Driven Technologies

Francisco José Gómez & Luigi Vanfretti
KTH Royal Institute of Technology
fragom@kth.se , luigiv@kth.se

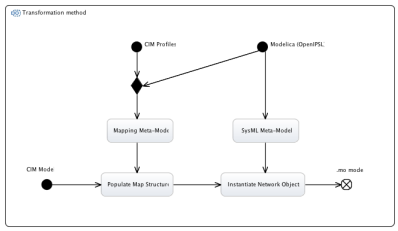
Motivation

- Regulation (EC) 714/2009 underlines the **need of coordination between transmission system operators (TSOs) -** CIM should aid in:
  - "model used to support common network operation tools to ensure coordination of network operation in normal and emergency conditions"
- Text-to-Text (T2T) transformations, "parsers" or "format converters/filters".

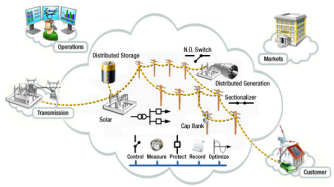


Description

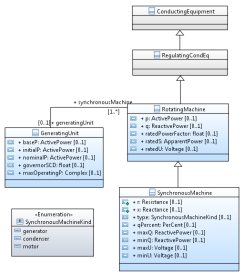
- Mode-Driven Software Development (MDSE) consist on the development paradigm based on models, for the development process and other model-based tasks of a complete software engineering process
- The development process within the MDSE requires models and model transformations.
  These transformations require the development of mappings between a source model and a target model.



Common Information Model

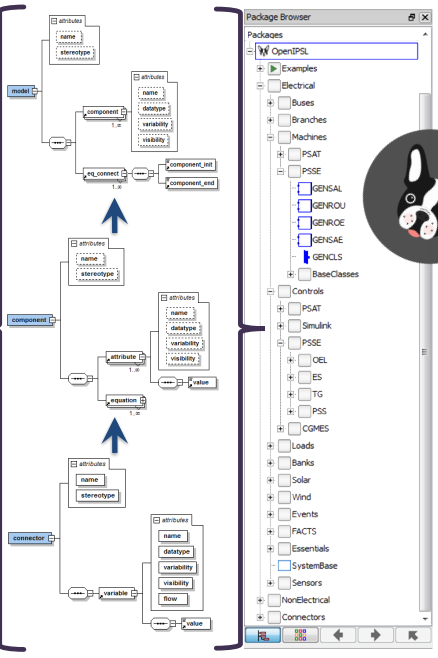


- The CIM Standard uses UML to represent the semantic information of a real power system.
- It defines all the basic components and topology of the power network, with its steady-state behavior



- **Dynamic models in CIM provide limited information on how the model is implemented.**

CIM-2-Modelica Mapping



Modelica Language

- Modelica is an object-oriented equation-based programming and modeling language, which allows the representation of cyber-physical systems using a strict mathematical representation.

```
protected
  parameter Real VRMAX0(fixed = false);
  parameter Real VRMIN0(fixed = false);
  parameter Real KE0(fixed = false);
  parameter Real VT0(fixed = false);
  parameter Real VREF(fixed = false);
  parameter Real Efd0(fixed = false);
  parameter Real SE_Efd0(fixed = false);
  parameter Real VR0(fixed = false);
  parameter Real Switch(fixed = false);
  parameter Real T_R(fixed = false);
initial equation
  VT0 = Ec0;
  Efd0 = EFD0;
  SE_Efd0 = SE(EFD0, SE1, SE2, E1, E2);
  (VRMAX0, KE0) = ini0(VRMAX, KE, E2, SE2, Efd0, SE_Efd0);
  VRMIN0 = -VRMAX0;
  VR0 = Efd0 * (KE0 + SE_Efd0);
  VREF = VR0 / KA + VT0 + Vs.a0;
  Switch = TR;
  T_R = if not Switch == 0 then TR else 1;
```

- Modelica is totally decoupled from the mathematical solver that is used to provide a numerical solution of the model equations

3.3. Modelica way to link with the code of the model

In Modelica the excitation model type SEXS is described by the equations related to the model. This equations are defined in a function[1] which has the input parameters that are the parameters of the model needed for the equations. For instance the excitation system type SEXS will be called with the following string:

Electrical.Controls.PSSE.ES.SEXS.SEXS sEXS(T_AT_B=0.3,T_B=10,K=200,T_E=0.05,E_MIN=0, E_MAX=4,EFD_MAX=4, EDF_MIN=0,KC=1,T_C=0)

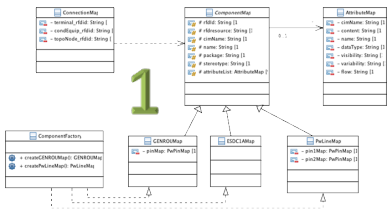
[1] Please note that equations of the models along with all of the parameters and variables are not functions in modelica but they are called models, or, in the traditional sense of object oriented programming, classes. So what is actually seen in the "calling" is instantiation of the exciter object from its class and specification of its parameters.

This string has the following main elements:

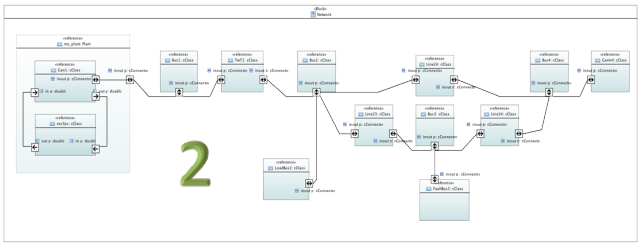
- The path (namespace) to the model i.e. *Electrical.Controls.PSSE.ES.SEXS.SEXS*
- The name of the model (the instantiated object) in the Modelica library, i.e. *sEXS*
- The parameters used by the Modelica code of the model, i.e. *(T_AT_B=0.3,T_B=10,K=200,T_E=0.05,E_MIN=0,E_MAX=4,EFD_MAX=4, EDF_MIN=0,KC=1,T_C=0, E0=1)*

ENTSO-E CGMES v2.5. and Annex F indicating the inclusion of Modelica in the standard

Workflow for Dynamic Simulation



UML class diagram, implementation design of a meta-model structure for the mapping rules. Populates CIM values from CIM model



SysML representation of a Modelica class. A block correspond an instance of an OpenIPSL components instances, with CIM values of the CIM model.

```
model IEEE_9BUS_CIM "something here"
OpenIPSL.Electrical.Branches.PwLine Ln46(R=0.017000000923871994,X=0.092000000017881393,G=0,B=0.07900000363588333) "something here" annotation();
..
OpenIPSL.Electrical.Buses.Bus BUS8 "something here" annotation();
..
OpenIPSL.Electrical.Loads.PSSE.Load Ld8(angle_0=-91.63185882568836,V_0=1.0116652250289917,P_0=1,Q_0=0) "something here" annotation();
..
OpenIPSL.Electrical.Machines.PSSE.GENROU.GENROU Gn2(S_b=100,M_b=320,V_b=18,V_0=1.0301070213131787,angle_0=-
82.51999664306644,P_0=177.21009826660156,Q_0=12.977800369262695,R_a=0,X1=0.10000000149011612,H=3.3299999237060547,D=0.6700000166893005,S10=1.0099999904632568,S12=1.0199999809265137,Tpd0=6,Tppd0=7.53500026226044,Tppd0=0.05000000074505806,Tppq0=0.05000000074505806,Xd=1.7200000286102295,Xpd=0.23000000417232513,Xppd=0.20999999344348907,Xq=1.659999966621399,Xpq=0.3700000047683716,Xppq=0.20999999344348907) "something here" annotation();
..
OpenIPSL.Electrical.Branches.PSSE.TwoWindingTransformer
T3(R=0,X=0.0625,G=0,B=0,ANG1=0,S_n=100,CW=0,CZ=0,t2=1,VNOM2=13.800000190734863,VB2=1.0229270458221436,t1=1,VNOM1=230,VB1=237.44049072265625) "something here" annotation();
..
// manually implemented
OpenIPSL.Electrical.Events.PwFault f_BUS8(R = 0.1, X = 0.009999999776482582, t1 = 0.9, t2 = 1.1) "something here" annotation();
equation
connect(Ln46.p, BUS4.p);
connect(Ld8.p, BUS8.p);
connect(Gn2.p, BUS2.p);
connect(T3.n, BUS3.p);
..
connect(Gn1.PMECH0, Gn1.PMECH);
connect(Gn1.EFD0, Gn1.EFD);
connect(Gn2.PMECH0, Gn2.PMECH);
connect(Gn2.EFD0, Gn2.EFD);
connect(Gn3.PMECH0, Gn3.PMECH);
connect(Gn3.EFD0, Gn3.EFD);
// manually implemented
connect(f_BUS8.p, BUS8.p);
end CIM_IEEE_9Bus;
```