CIM-Compliant Model-to-Model Transformation

For Modelica Models Generation and Power Systems Dynamic Simulations

Francisco J. Gómez[1], Prof. Luigi Vanfretti[1]

Svein H. Olsen[2]

fragom@kth.se, luigiv@kth.se
Electric Power Systems Dept.
KTH
Stockholm, Sweden

svein.harald.olsen@statnett.no
Research and Development Division
Statnett SF
Oslo, Norway

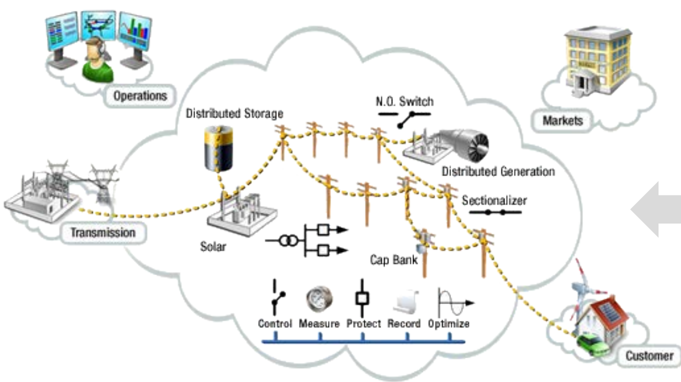
MODPROD, 8TH February 2017, Linköping

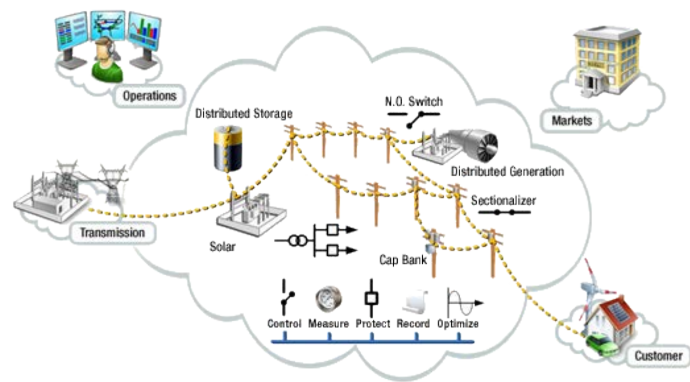
Outline

- Introduction & Motivation
- Background on MDSE & Model Transformations
  - M2T, M2M
  - UML & SysML
- Workflow M2M – Research Focus on MDSE
  - Reverse Engineering
  - Binding Semantics: Mapping Meta-Model
  - Power Network SysML Meta-Model
- Conclusions

Exchange of information

- Network planning, power systems operations, demands high degree of coordination and consistency in data exchanges,
- Significantly streamlined through a common data exchange standard
- The exchange of dynamic models provides power system data related to the parameters of an associated block diagram

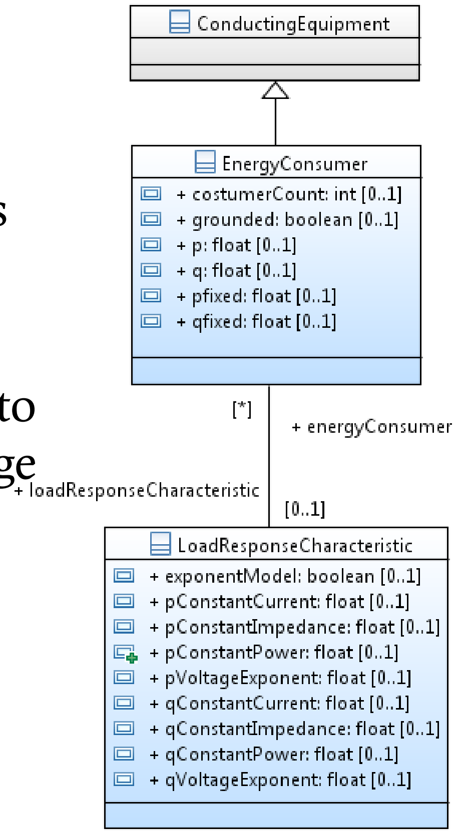




Harmonization of the different information modeling and physical modeling computer languages attractive to support power system model exchange and dynamic applications

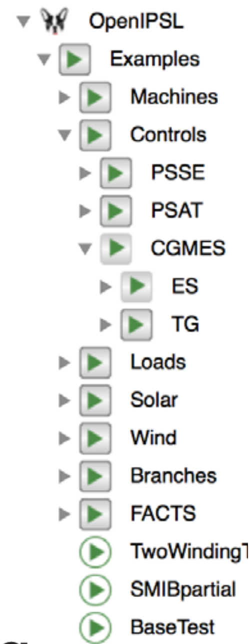
Common Information Model

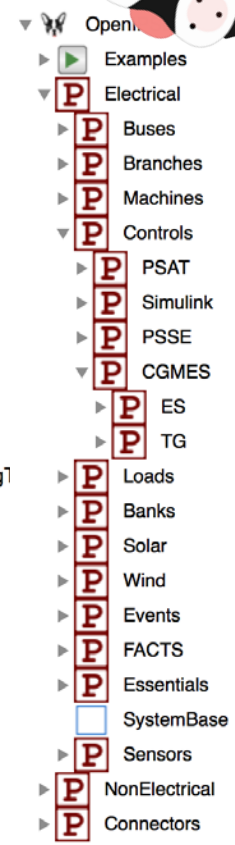
- IEC CIM Standard based on UML to represent semantic information of a real power system.
- OOP principles, defines all the basic components and topology of the power network
- ENTSO-E adopts different IEC CIM standards to conform the Common Grid Model Exchange Standard (CGMES)



The *OpenIPSL* Power System Library

- OpenIPSL Modelica library for modeling power grid components.
  - Model components
  - Tested and validated against reference software tools
  - Sample test networks (IEEE models, and others)





- The library makes available standardized and *de facto* standard power systems models usually available in power system tools only accessible through proprietary (and expensive) licenses

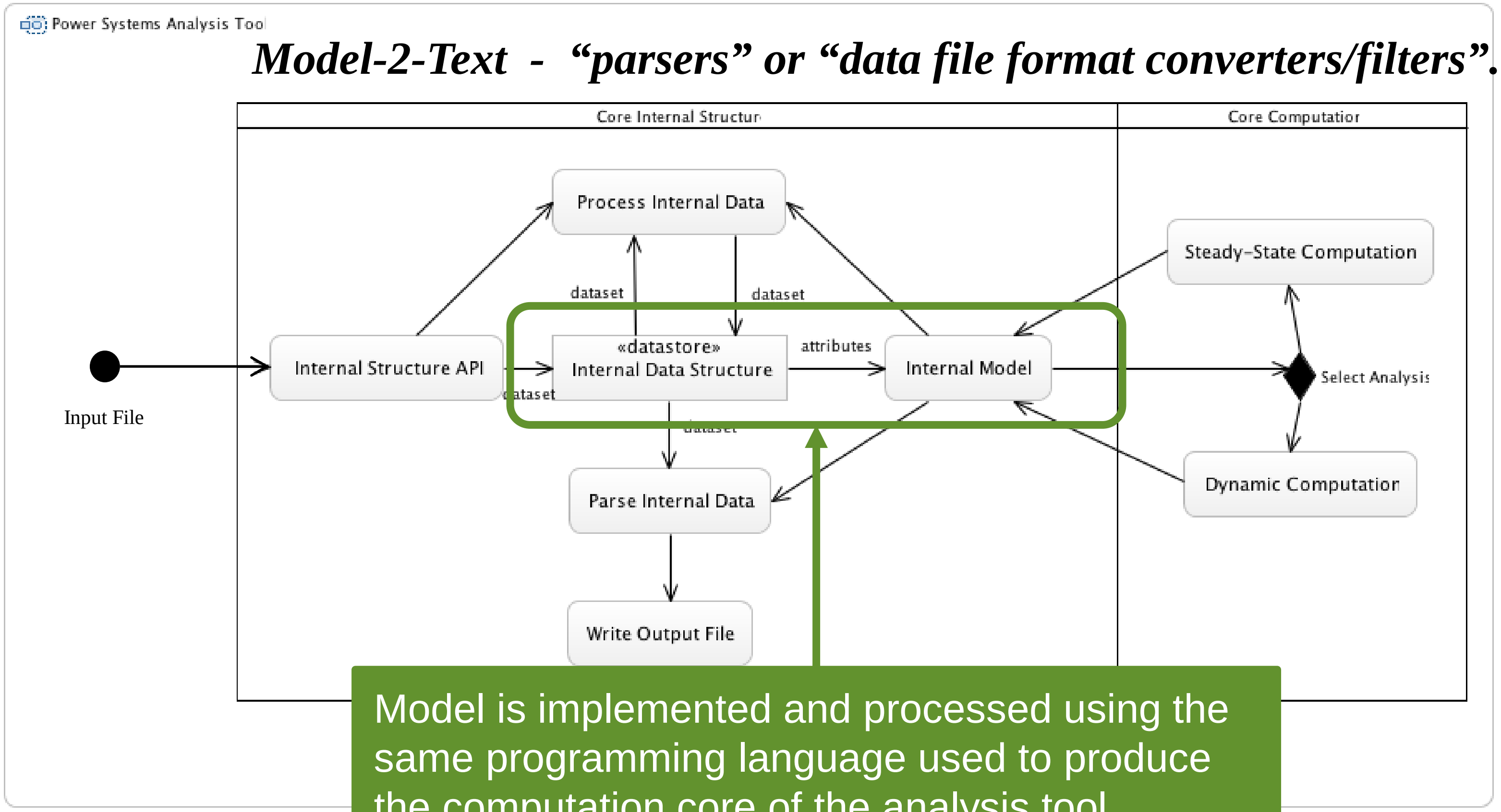
Outline

- Background on MDSE & Model Transformations
  - M2T, M2M
  - UML & SysML

M. Brambilla, J. Cabot, and M. Wimmer, “Model-driven Software Engineering (MDSE) in Practice”, Morgan & Claypool, USA, 2012, pp. 9-11, ISBN: 978-1-608-45882-0

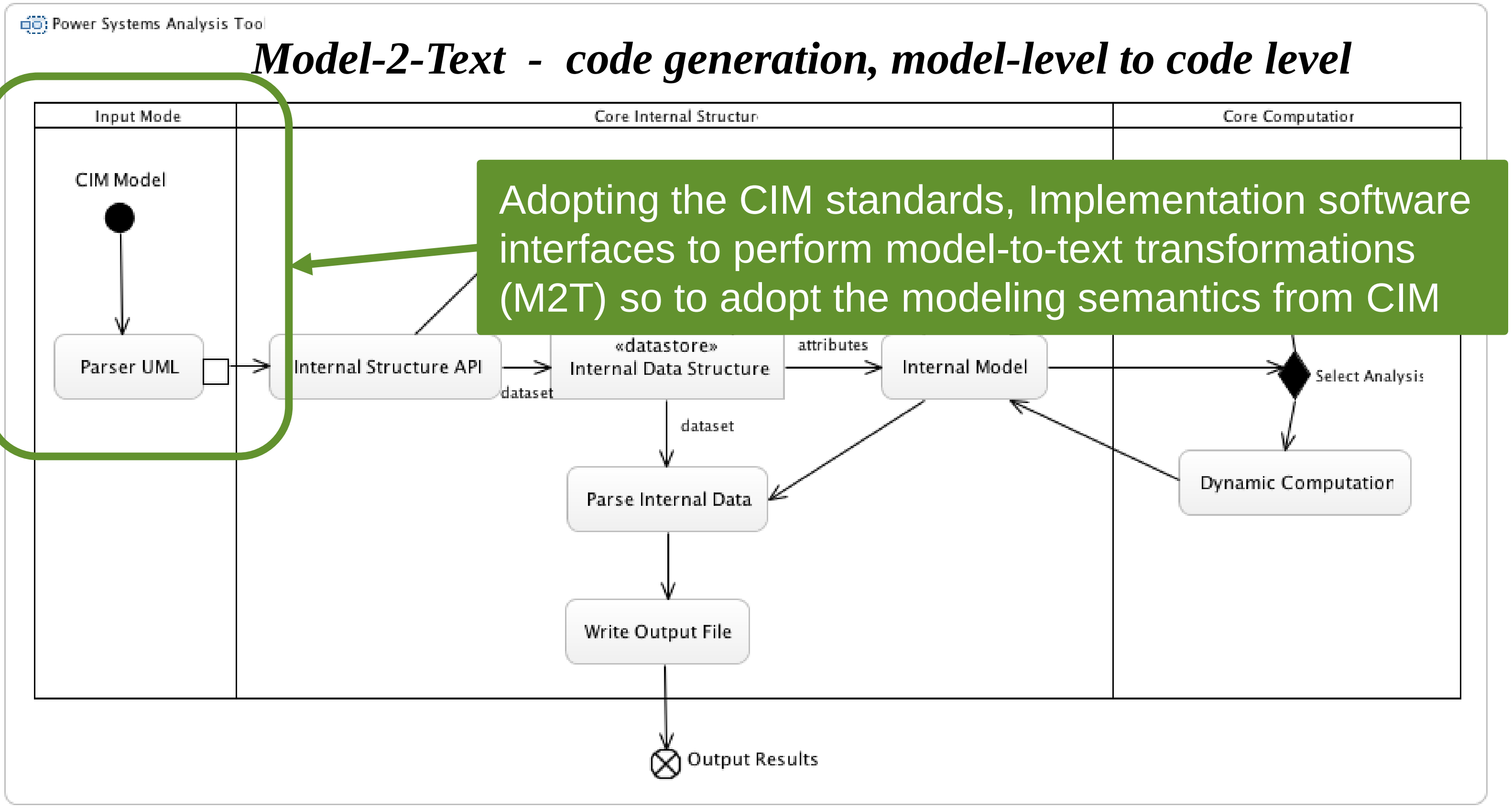
Model transformation – M2T

Model-2-Text - "parsers" or "data file format converters/filters".



Model is implemented and processed using the same programming language used to produce the computation core of the analysis tool

Model transformation – M2T

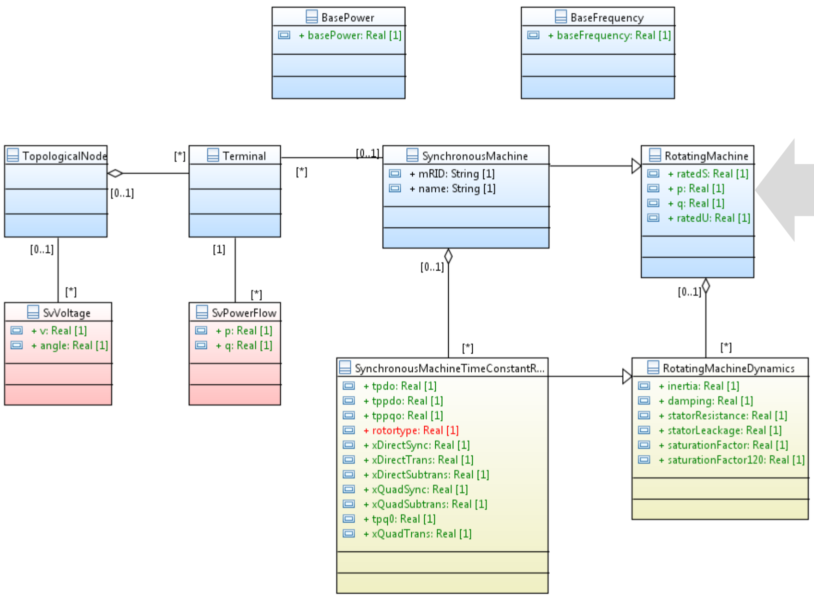


Model transformation – M2M

Model-2-Model transformation processes source models to generate target models,

Endogenous (in-place) – transformation defined within the same modeling language

Exogenous (out-place) – transformation between different modeling languages



Mappings

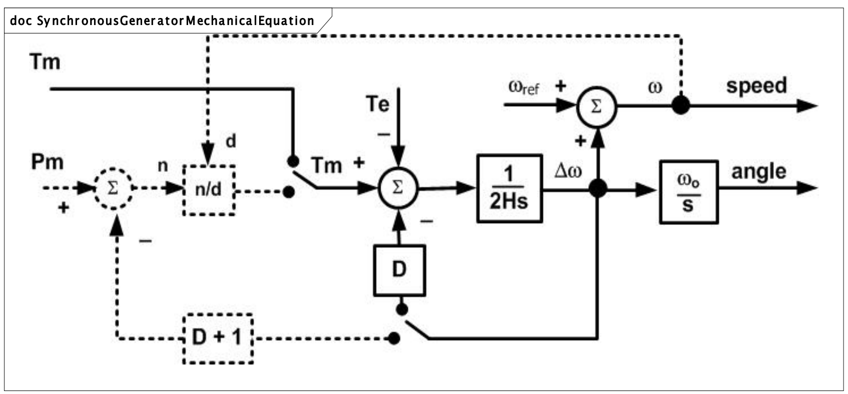
```
model gensal
   …
   parameter Real wbase = 2 * pi * 50 "system base speed";
   parameter Complex Epqp = fpp + a * It;
   parameter Real delta0 = arg(Epqp);
   parameter Real Pm0 = p0 + (id0 * id0 + iq0 * iq0) * Ra;
   Real delta "rotor angle";
   Real w "machine speed deviation, p.u.";
   …
initial equation
  delta = delta0;
  w = 0;

equation
  …
  der(w) = ((Pm0 - D * w) / (w + 1) - Te) / (2 * H);
  der(delta) = wbase * w;
end gensal;
```

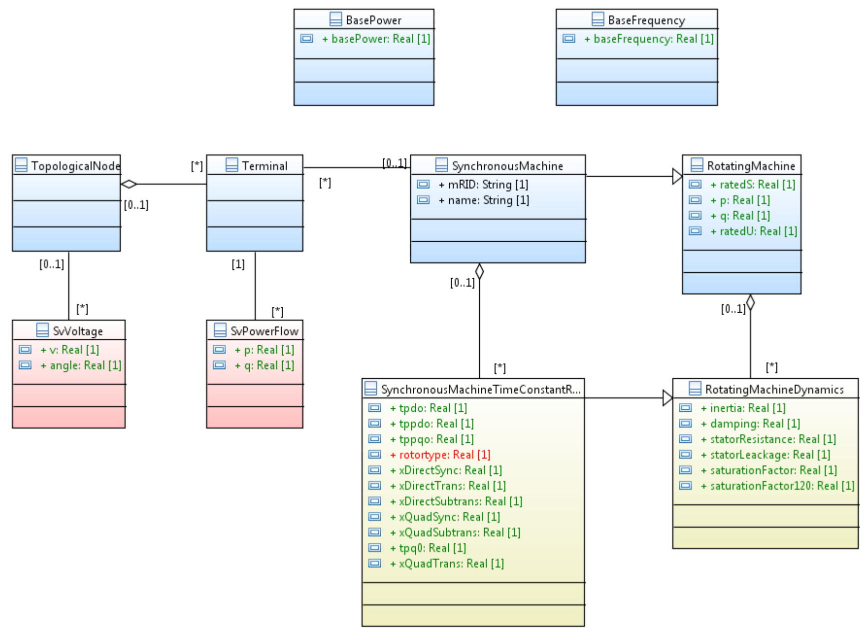
UML & CIM

- UML: set of model elements representing an analysis of the properties and behavior of a system.

- IEC 61970-301 CIM Base UML package containing static information
- IEC 61970-457 CIM for Dynamics Profile, UML package containing dynamic information

- **Limited description of the dynamics information**

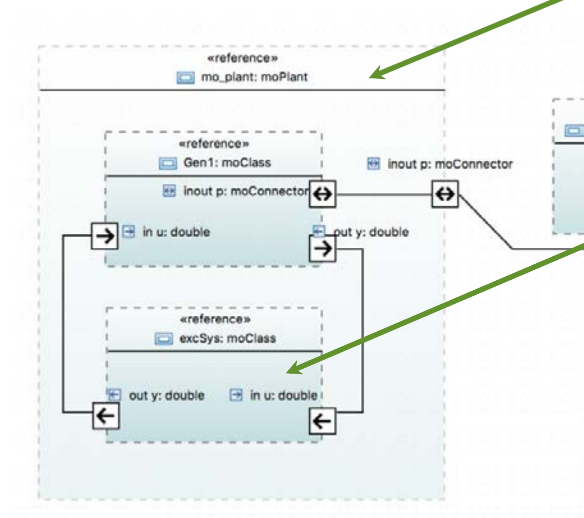


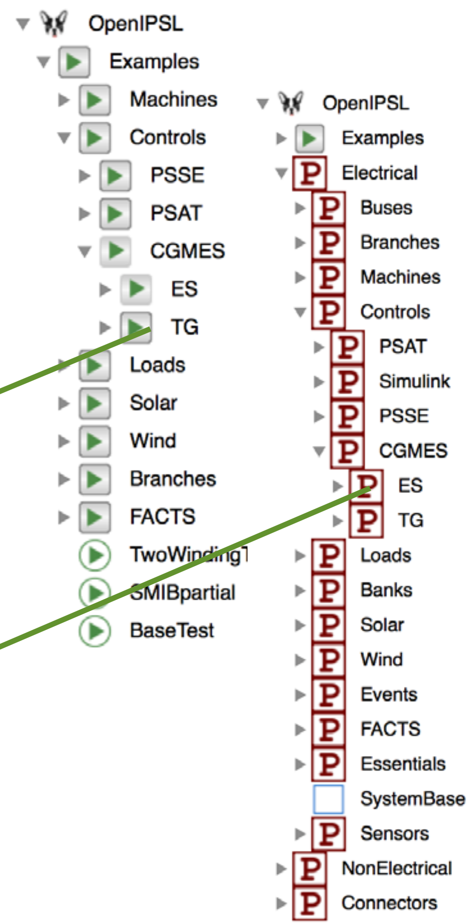


SysML & Modelica

- SysML: Add design principles such as requirements modeling and reuse of UML class diagrams as block diagrams supporting parametric modeling interoperability

- Document and design concept model that we want to implement
- Modelica language offers mechanism to implement SysML models





Outline

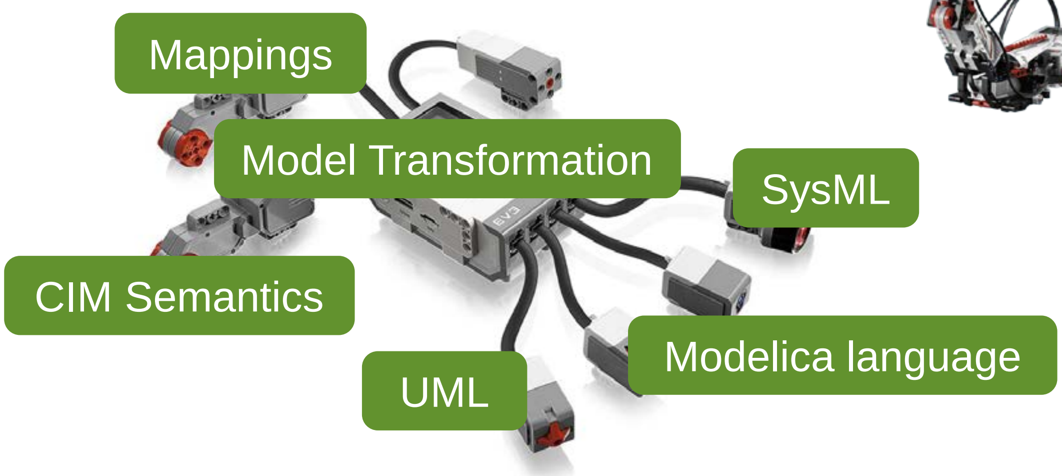
- Workflow M2M – Research Focus on MDSE
  - Reverse Engineering
  - Binding Semantics: Mapping Meta-Model
  - Power Network SysML Meta-Model

F. J. Gómez, L. Vanfretti and S. H. Olsen, "Binding CIM and Modelica for consistent power system dynamic model exchange and simulation", 2015 IEEE Power & Energy Society General Meeting, Denver, CO, 2015, pp. 1-5. doi: 10.1109/PESGM.2015.7286434

Reverse engineering

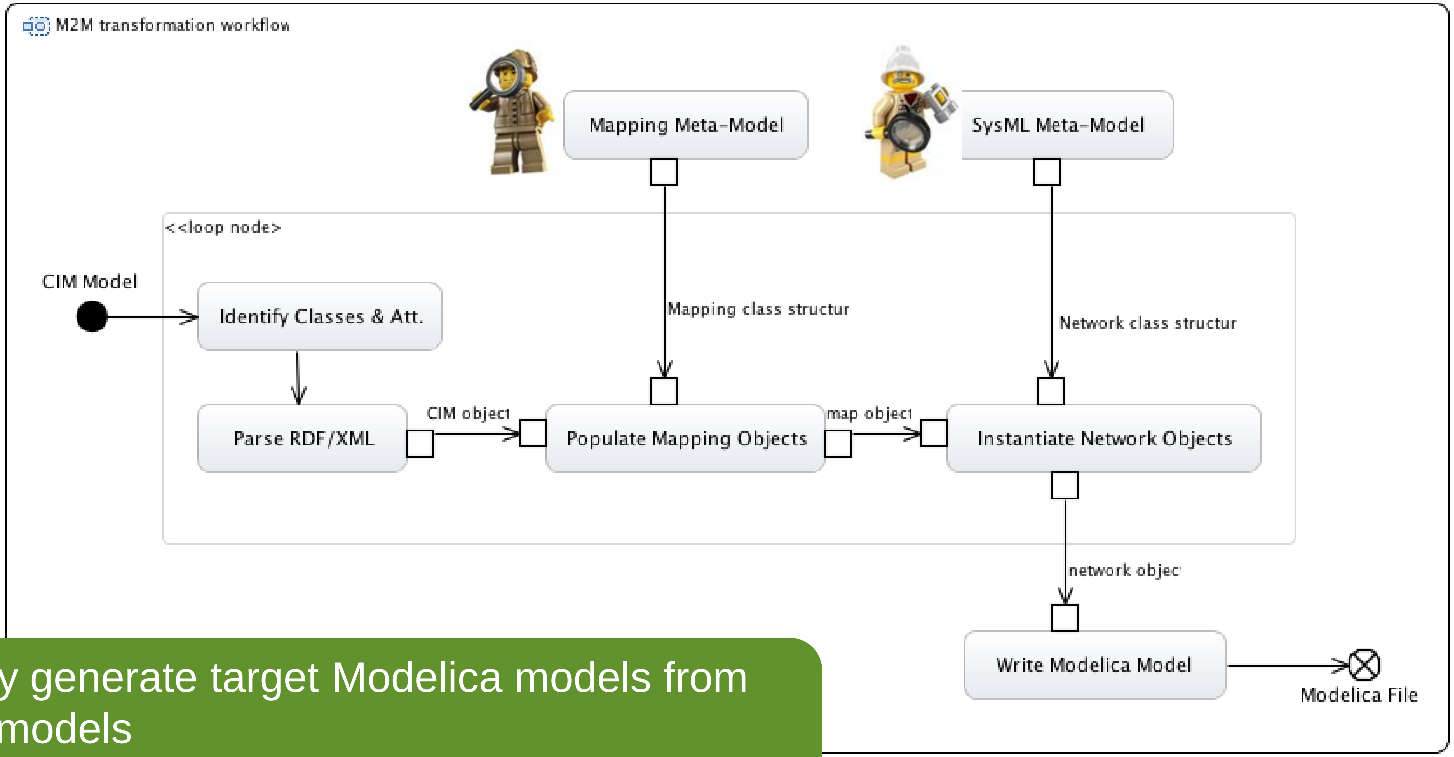
- CIM 2 Modelica Factory
    - XML for component mapping,
    - JAVA code implementation

Apply Model-Driven Software Engineering principles: Let´s formalized this solution!



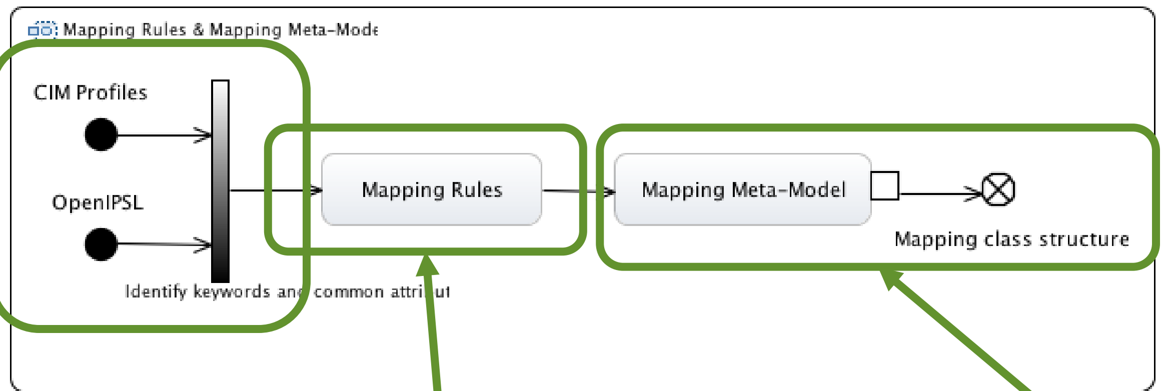
Model-2-Model Transformation

Design of a M2M workflow to generate power system Modelica models generation from CIM



Automatically generate target Modelica models from source CIM models
General workflow, represent key actions to implement

Binding Semantics: Mapping meta-model

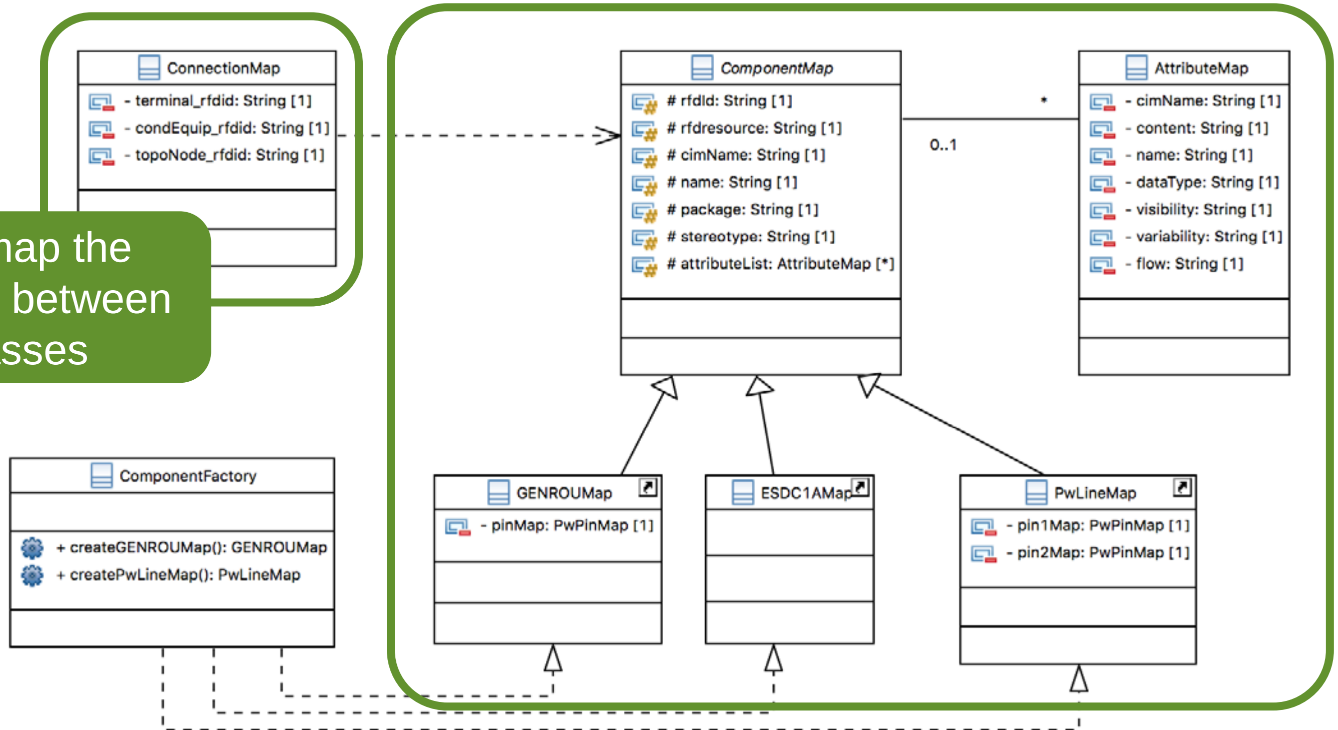


Identification of relevant attributes from CIM and OpenIPSL

Each OpenIPSL component has mapping rules

Class Structure / Meta-Model to populate CIM values
JAXB (XML/JAVA parser)

Binding Semantics: Mapping Meta-Model



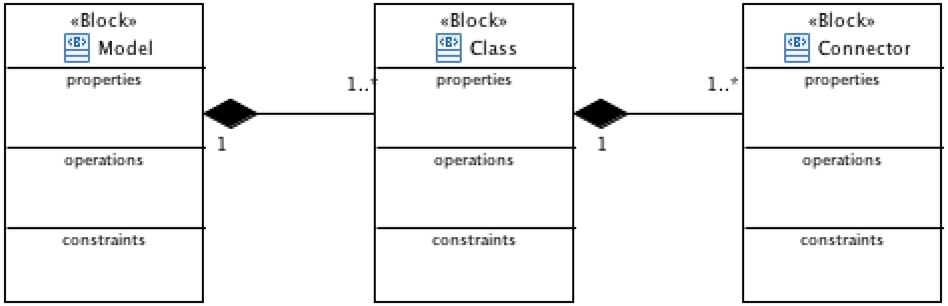
Class to map the connections between CIM classes

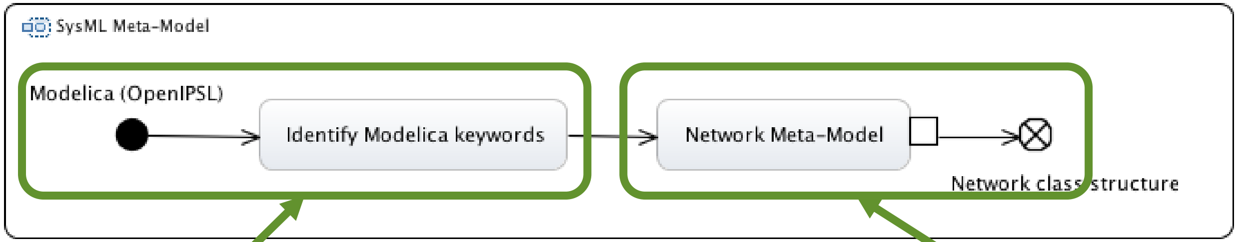
Component's mapping: Specific class for each component and general classes for common attributes

Power Network SysML Meta-Model

Modelica language stereotypes for models and components.

- **Model** – for a high-level power system model
- **Class** – for component-level power system model
- **Connector**

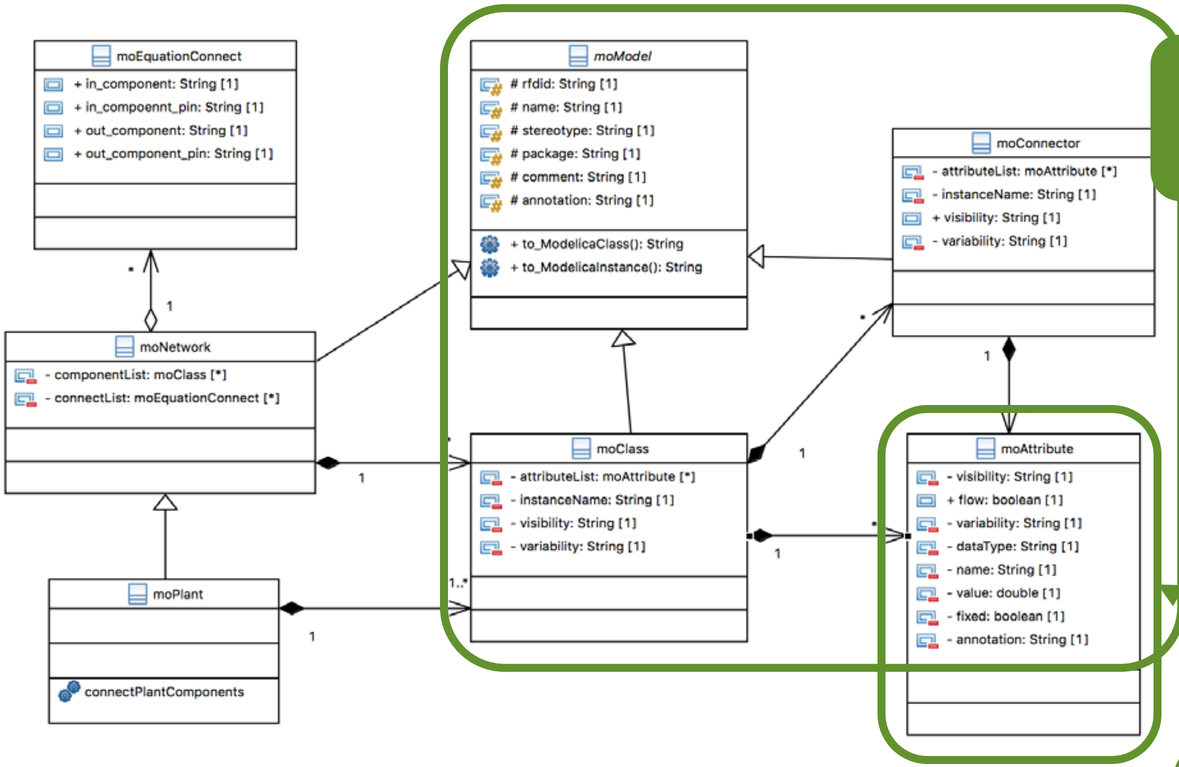




Identification of relevant keywords to use for parameter, variable and object declaration

Meta-model, to instantiate component objects

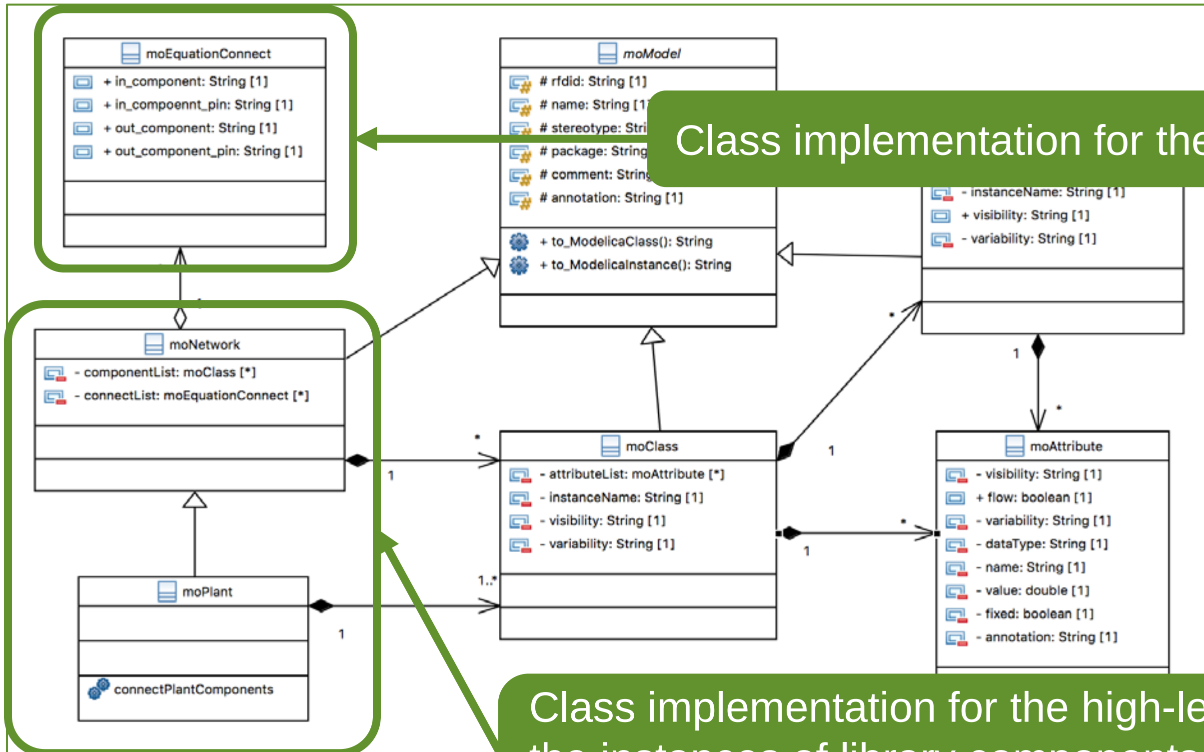
Power Network SysML Meta-Model



One class for each stereotype with the relevant keywords for parameter declaration

Define the declaration of the component's variables

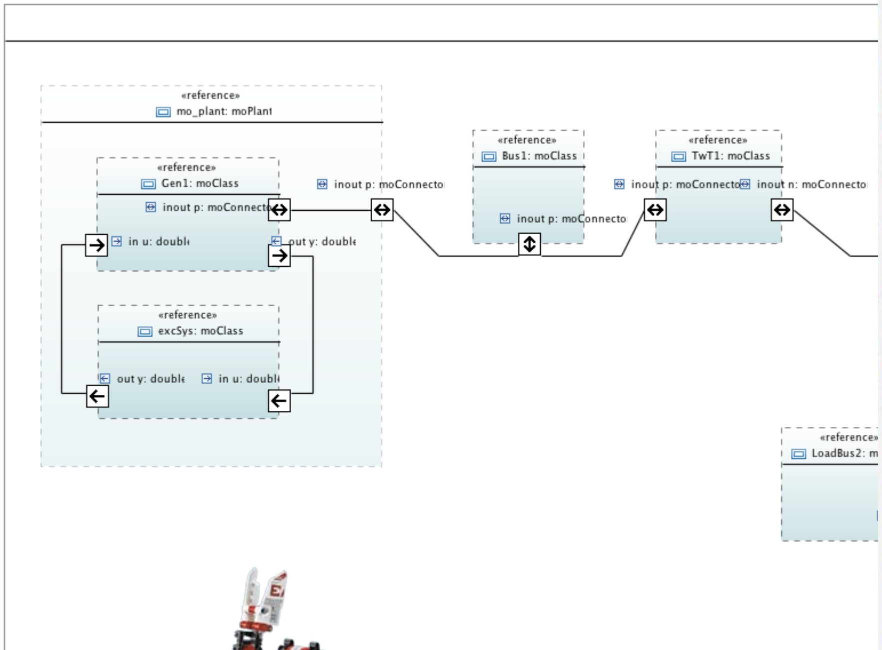
Power Network SysML Meta-Model



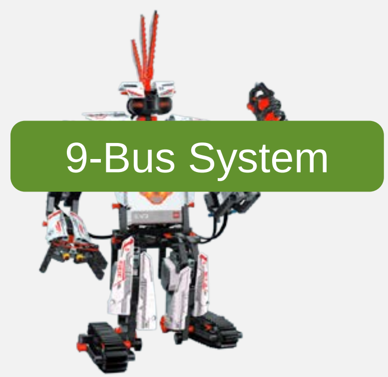
Class implementation for the connections

Class implementation for the high-level models. Store the instances of library components that compose the power system models

Model-2-Model T




```
model IEEE_9BUS_CIM "automatically generated comment"
OpenIPSL.Electrical.Branches.PwLine Ln46(R=0.017,X=0.092,G=0,B=0.079) "something here" annotation();
OpenIPSL.Electrical.Buses.Bus BUS4 "automatically generated comment" annotation();
OpenIPSL.Electrical.Buses.Bus BUS2 "automatically generated comment" annotation();
OpenIPSL.Electrical.Buses.Bus BUS7 "automatically generated comment" annotation();
OpenIPSL.Electrical.Buses.Bus BUS6 "automatically generated comment" annotation();
OpenIPSL.Electrical.Loads.PSSE.Load Ld8(angle_0=-91.63,V_0=1.011,P_0=1,Q_0=0) "automatically generated comment"
annotation();
OpenIPSL.Electrical.Buses.Bus BUS8 "automatically generated comment" annotation();
OpenIPSL.Electrical.Machines.PSSE.GENROU.GENROU Gn2(S_b=100,M_b=320,V_b=18,V_0=1.03,angle_0=-82.51, P_0=177.21,
Q_0=12.97,R_a=0,Xl=0.10,H=3.32,D=0.67,S10=1.01,S12=1.02,Tpd0=6,Tpq0=0.535,Tppd0=0.05,Tppq0=0.05,Xd=1.72,Xpd=0.23,
Xppd=0.20,Xq=1.65,Xpq=0.37,Xppq=0.20) "automatically generated comment" annotation();
OpenIPSL.Electrical.Branches.PwLine Ln57(R=0.032,X=0.16,G=0,B=0.15) "automatically generated comment" annotation();
OpenIPSL.Electrical.Buses.Bus BUS5 "automatically generated comment" annotation();
OpenIPSL.Electrical.Loads.PSSE.Load Ld6(angle_0=-96.99,V_0=1.01,P_0=1,Q_0=0) "automatically generated comment"
annotation();
OpenIPSL.Electrical.Branches.PwLine Ln45(R=0.009,X=0.085,G=0,B=0.087) "automatically generated comment" annotation();
OpenIPSL.Electrical.Loads.PSSE.Load Ld5(angle_0=-97.19,V_0=0.99,P_0=1,Q_0=0) "automatically generated comment"
annotation();
OpenIPSL.Electrical.Branches.PwLine Ln89(R=0.011,X=0.1,G=0,B=0.1045) "automatically generated comment" annotation();
OpenIPSL.Electrical.Buses.Bus BUS9 "automatically generated comment" annotation();
OpenIPSL.Electrical.Branches.PSSE.TwoWindingTransformer
T3(R=0,X=0.0625,G=0,B=0,ANG1=0,S_n=0,CW=0,CZ=0,t2=1,VNOM2=13.8,VB2=1.023,t1=1,VNOM1=230,VB1=237.44) "automatically
generated comment" annotation();
OpenIPSL.Electrical.Branches.PSSE.TwoWindingTransformer
T1(R=0,X=0.0586,G=0,B=0,ANG1=0,S_n=0,CW=0,CZ=0,t2=1,VNOM2=230,VB2=235.93,t1=1,VNOM1=16.5,VB1=1.038) "automatically
generated comment" annotation();
OpenIPSL.Electrical.Branches.PSSE.TwoWindingTransformer
T2(R=0,X=0.0575,G=0,B=0,ANG1=0,S_n=0,CW=0,CZ=0,t1=1,VNOM1=18,VB1=1.03,t2=1,VNOM2=230,VB2=235.927) "automatically
generated comment" annotation();
OpenIPSL.Electrical.Buses.Bus BUS3 "automatically generated comment" annotation();
OpenIPSL.Electrical.Machines.PSSE.GENSAL.GENSAL Gn1(S_b=100,M_b=275,V_b=16.5,V_0=1.038, angle_0=-94.34, P_0=58.78,
Q_0=27.51,R_a=0,Xl=0.06,H=9.55,D=1.6,S10=1.01,S12=1.02,Tpd0=8.96,Tppd0=0.05,Tppq0=0.05,Xd=0.36,Xpd=0.15,Xppd=0.1,
Xq=0.24,Xppq=0.1) "automatically generated comment" annotation();
OpenIPSL.Electrical.Buses.Bus BUS1 "automatically generated comment" annotation();
OpenIPSL.Electrical.Branches.PwLine Ln78(R=0.0085,X=0.072,G=0,B=0.0745) "automatically generated comment"
annotation();
OpenIPSL.Electrical.Machines.PSSE.GENROU.GENROU Gn3(S_b=100,M_b=300,V_b=13,V_0=1.03,angle_0=-87.72, P_0=89.56,
Q_0=-12.13, R_a=0,Xl=0.15, H=2.35, D=0.47, S10=1.01, S12=1.02, Tpd0=5.89, Tpq0=0.6, Tppd0=0.05, Tppq0=0.05, Xd=1.68,
Xpd=0.23,Xppd=0.21,Xq=1.61,Xpq=0.32,Xppq=0.21) "automatically generated comment" annotation();
OpenIPSL.Electrical.Branches.PwLine Ln69(R=0.039,X=0.17,G=0,B=0.179) "automatically generated comment" annotation();
// the fault component is added manually, to show dynamic reponse of the system
OpenIPSL.Electrical.Events.PwFault f_BUS8(R = 0.1, X = 0.01, t1 = 0.9, t2 = 1.1) "automatically generated comment"
annotation();
equation
connect(Ln46.p, BUS4.p);
connect(T2.n, BUS7.p);
connect(T2.p, BUS2.p);
connect(Ld8.p, BUS8.p);
connect(Gn2.p, BUS2.p);
connect(Ln57.p, BUS5.p);
connect(Ld6.p, BUS6.p);
connect(Ln45.n, BUS5.p);
connect(Ld5.p, BUS5.p);
connect(Ln89.n, BUS9.p);
connect(T3.n, BUS3.p);
connect(T1.n, BUS4.p);
connect(Ln89.p, BUS8.p);
connect(Ln45.p, BUS4.p);
connect(Ln57.n, BUS7.p);
connect(T1.p, BUS1.p);
connect(Gn1.p, BUS1.p);
connect(Ln78.n, BUS8.p);
connect(Ln46.n, BUS6.p);
connect(Gn3.p, BUS3.p);
connect(Ln69.p, BUS6.p);
connect(Ln69.n, BUS9.p);
connect(Ln78.p, BUS7.p);
connect(T3.p, BUS9.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);
// the connection of the fault component with the Bus component is added manually
connect(f_BUS8.p, BUS8.p);
end CIM_IEEE_9Bus;
```



Outline

- Conclusions

Conclusions

- Workflow that defines a modeling process that takes advantage of CIM semantics, UML/SysML and Modelica languages.

- Defines a method to complement CIM Dynamics profile with equation-based component model definitions for physical modeling behavior,
  - Using the Modelica language for supporting the CIM, for dynamic simulation analysis.

- Defines of a scalable, modular and reusable mapping between different modeling semantics for M2M transformation.

Thank you!

MODPROD, 8^{TH} February 2017, Linköping