Binding CIM and Modelica for Consistent Power System Dynamic Model Exchange and Simulation

Francisco José Gómez[1], Luigi Vanfretti[12], Svein H. Olsen[2]
[1]Electric Power Systems Department, KTH Royal Institute of Technology, Stockholm, Sweden
[2]Research and Development, Statnett SF, Oslo, Norway
fragom@kth.se, luigiv@kth.se, svein.harald.olsen@stattnet.no

***Abstract*— The Common Information Model (CIM) is considered the most prominent data model for power systems information exchange between Transmission System Operators (TSO), facilitating coordination of TSO for steady-state operation. However, information exchange should also consider power systems dynamic models required to perform dynamic simulations so to coordinate TSOs operations under emergency conditions. This work describes the design and implementation of a mapping between CIM and Modelica. The Modelica models provide a strict mathematical representation of power system dynamic models using a standardized modeling language. The proposed solution combines both modeling languages, thus providing a CIM-compliant unambiguous power system model exchange and simulation solution considering both steady-state and dynamic models.**

***Index Terms*— CIM, Modelica, Power System Dynamics**

I. Introduction & Motivation

A. The European Drive for Information Exchange

The European Commission (EC) adopted the The Third Energy Package [1] of legislative proposals on 19 of September of 2007 to enable a competitive and integrated energy market. As a result, Regulation (EC) 714/2009 [2] established the European Network of Transmission System Operators for Electricity (ENTSO-E), which bears the responsibility of ensuring secure and reliable operation of the European interconnected transmission system[1]. The regulation considered the need of coordination between TSOs, which should "use a common transmission model dealing efficiently with interdependent physical loop-flows and having regard to discrepancies between physical and commercial flows". This model should be used by ENTSO-E to support "common network operation tools to ensure coordination of network operation in normal and emergency conditions".

After this regulation, the EC issued Mandate M/490 to European standardization bodies CEN/CENELEC and ETSI, which resulted in a report [3] that recommends integration technology to facilitate the "establishment of a common information model that is to be used throughout many applications and systems". The report highlights that the Common Information Model (CIM) (IEC 61968, 61970 and 62325) and the IEC 61850 data model are the most prominent data models, as reported in [4]. The ENTSO-E has made large efforts to comply with the regulation and mandates, and as a result the Common Grid Model Exchange Standard (CGMES) was approved and conformity of the tools to CGMES have been carried out through Interoperability (IOP) tests [5].

B. Challenges of CIM for Unambiguous Model Exchange

The results from different IOP tests have resulted in evolutions of the CIM standard. It is a challenge to develop CIM to fulfill the functions of Regulation (EC) 714/2009, as it will be necessary to exchange models not only for steady state operation, but also for power system dynamic models so to facilitate coordination of network operation in emergency conditions. Exchange of dynamic information is currently considered in CIM through IEC 61970-302 [6] and IEC 61970-457 [7]. These standards propose the exchange of dynamic models by providing power system data related to the parameters of an associated block diagram. The main issue with this approach is that the exchange of such information cannot guarantee unambiguous model exchange between different applications (i.e. simulation tools) [8].

The approach has two main back draws. First, dynamic models for different components are not consistent through platforms due to simplifications, modeling philosophy and assumptions. Second, conventional block diagram modeling forces users to share only parameters of models with predetermined structure, the model's mathematical representation is therefore not shared explicitly. This leaves open to interpretation how the actual implementation of the models is carried out in each application. As a consequence, two different model implementations of the same *specification* and *parameters/block diagram* will be inconsistent and will produce inconsistent simulation results. This is unacceptable as coordination has to take place also during emergency conditions where dynamic power system simulation is required in order to perform dynamic security assessment.

[1] This work was supported in part by the EU-funded FP7 iTESLA project and by Statnett SF, the Norwegian Transmission System Operator. iTESLA (Innovative Tools for Electrical System Security within Large Areas), online: www.iTESLA-project.eu

978-1-4673-8040-9/15/$31.00 ©2015 IEEE

However, explicitly exchanging the equations of the model may aid in achieving consistency across different simulation platforms.

C. Paper Contributions and Organization

This paper takes a first step to addresses the challenge of CIM-compliant unambiguous model exchange and simulation when power system dynamics are considered. To do so, a combination of CIM and Modelica languages is proposed so that dynamic model information exchange between Transmission System Operators (TSO) can be carried out unambiguously. The paper proposes a systematic binding of classes from the iTesla Power System Modelica Library (iPSL) [8] with CIM, which at the same time will allow for automated power system equation-based simulation using Modelica tools.

The reminder of this paper is organized as follows. Section II describes benefits of modeling with both CIM and Modelica; in section III the representation of CIM schema and Modelica code is described; section IV describes the proposed mapping schema and its implementation. In Section V conclusions and future work is discussed.

II. Background: CIM Modeling and Modelica

A. UML & CIM in Power Systems

Building a model involves the interpretation of components of the physical world and their properties, and an understanding of the physical laws that bound their interaction. The Unified Modeling Language (UML) follows the Object-Oriented Programming (OOP) principles [9], defining a set of classes, attributes diagrams and rules to represent physical objects their properties and behavior into a data model.

The CIM Standard uses UML to represent the semantic information of a real power system. CIM follows the OOP principles, since it defines all the basic components and topology of the power network, with its steady-state behavior and a limited description on its dynamics. A data model built using CIM represent the power systems objects as objects that can be quickly converted to classes in an appropriate object-oriented programming language application [10].

In the industry, different TSOs use different software tools for analysis and simulation. CIM defines a common data model for the exchange of information between TSOs. However, CIM does not include a strict mathematical representation that can describe the dynamic behavior of each component and their interaction.

B. The Choice of Modelica for Power Systems Modeling

Modelica is an object-oriented equation-based programming and modeling language, which allows the representation of cyber-physical systems using a strict mathematical representation, for the purpose simulations using digital computers. This is possible thanks to the standardized Modelica language definition [11].

The Modelica compiler compiles the Modelica model into machine code and executes it together with a numerical solver of choice, capable of solving sets of differential and algebraic equations (DAE), discrete equations or their combination. Thus, a Modelica model is not coupled to any specific solver[2], and can be easily exchange between software environments or specific applications capable of compiling Modelica code. This characteristic with OOP compliance makes Modelica a suitable language for the implementation of dynamic models of power systems components, from CIM models.

Modelica tools like ModelicaML [12] exploits this characteristic, allowing the design of models and classes and the generation of Modelica code from a data model representation. Hence, from the CIM representation of a power system, Modelica code can be generated and the model variables can be properly initialized with the power flow solution stored in the corresponding CIM model.

C. Initialization of Modelica models

Before performing time-domain simulations, the initialization process takes place. In power system simulation tools this process is only concerned in assigning initial values to the continuous variables of the model, starting from a power flow solution. This means that the equilibrium of the model's differential equations is computed after the equilibrium of the algebraic variables has been found. In contrast, Modelica solves the initialization problem by finding the equilibrium of the entire set of equations (algebraic, continuous and discrete).

To initialize a Modelica model, start values (i.e. initial guess) for *all algebraic*, continuous and discrete variables need to be provided [13]. These values should either be written manually when designing the model or computed through an external routine. Otherwise, initialization problems may arise. Not assigning an initial value to algebraic states or using default values (0 or no value) may result in divergences of the initialization algorithm. Modelica tools like Dymola and OpenModelica manipulate symbolically the initialization problem and generate analytic Jacobians for nonlinear problems. This approach is more robust and reliable than specifying a 0, or no value, for initial conditions of the set of Differential Algebraic (DAE) equations. Different works address initialization issues in Modelica. [13],[14],[15] provide different algorithms to state or reduce the number of initial conditions of a model, but do not consider start (guess) values.

CIM tools, like CIMphony Composer [16], can perform power flow calculations and the solution into the corresponding CIM data model. This defines a mapping to create an automatic conversion of CIM models into Modelica code, which allows providing start values to the algebraic and dynamic states of a Modelica model from a power flow solution. From a CIM steady-state model with power flow solution, the start guess values can be provided directly into the parameters and equations of the Modelica models. Thus, the model is ready for performing time-domain simulations.

[2] The solvers available depend on each tool's implementation.

III. DESCRIPTION OF CIM DATA FORMAT AND MODELICA

A. CIM RDF Schema

The data base of components and connections represented by the CIM data model are stored in a single file, using eXtensible Markup Language (XML) serialization, based on the Resource Description Framework (RDF). With the RDF schema, objects and relations between them are stored in graph-like data structures (see Fig. **1** and Fig. **2**). Serialization, checking and making files for storing topology changes are easier to interpret by XML processing tools, like XML Spy [17] that are able to read graph-like structures.

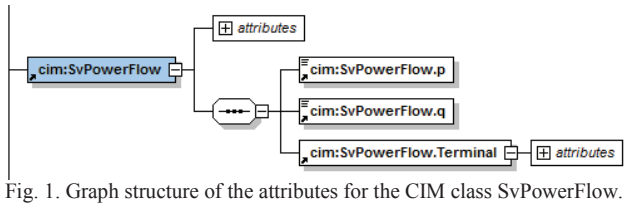

Fig. 1. Graph structure of the attributes for the CIM class SvPowerFlow.

Fig. 2. Graph structure of the attributes of the CIM class SynchronousMachine.

This kind of representation makes it easy to understand the attributes and the associations that CIM classes have between them. In case of Fig. 1, the CIM class **SvPowerFlow** has attributes *p* and *q* and an association with a CIM class Terminal. This class is used to store the power flow solution for *P* and *Q* and the connection terminal to which these values are associated with.

B. Modelica Code Representation

A model implemented in the Modelica language is comprised of three parts. First a specialization keyword, indicating which kind of class should be represented, and the name of the class, open the model declaration. Second, there is the variable section where all the attributes of the model are defined, with their proper visibility, variability, name, value and comments. The variability indicates the kind of variable; and visibility indicates the level of protection of the variable. Third there is the equation section where the mathematical model is represented. An implementation of a Pi-equivalent model for a Line for the iPSL is represented by next lines of code, with attributes and equations:

```
class Line "Pi-Line"
  PowerSystems.Connectors.PwPin p ;
  PowerSystems.Connectors.PwPin n ;
  parameter Real R "Resistance p.u.";
  parameter Real X "Reactance p.u.";
  parameter Real G "Shunt half conductance p.u.";
  parameter Real B "Shunt half susceptance p.u.";
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 Line;
```

The variability of a certain variable will explicitly indicate how to assign the start value of the variable. A variable with no variability indicates that its value is able to change during the simulation, so a start keyword should be indicated for assigning the initial value of the variable:

```
Real P (start=0.25) "active power injection";
```

For other kind of variability such as the **parameter** variability indicates that its value does not change during simulation, but can be changed between simulations. The starting value of a variable will be assigned as an equation:

```
parameter Real S = 100 "Base apparent power of the
component";
```

The remaining attributes from other classes will have *parameter* variability.

IV. TOWARDS A CIM 2 MODELICA MAPPING

The mapping between CIM/UML and Modelica code is based on ModelicaXML [18]. ModelicaXML defines a set of rules for exporting Modelica code to XML. The structure of the mapping is based on the definition of ModelicaXML. Considering the different Modelica class stereotypes, in this work a proposal for mapping the **model** stereotype, the **class** stereotype and the **connector** stereotype is described. Only the mapping of CIM data will be explained in detail in this work. The Modelica equations should be mapped from other sources. For power system phasor time-domain simulations the iTesla Power Systems Modelica Library is used to provide the dynamic models of different power system components in the Modelica language [8].

A. Mapping Power System Model

The **model** tag defines the top level model of a complete Modelica model. The root element of a CIM/XML file will be mapped as **model** stereotype. (see Fig. **3**). The model contains a set of component references and equations of type connect. A *component* tag is used for mapping the ID reference of other classes, so only the name of the class and the data type values will be loaded from the CIM data model. For a top level power system model, when only the connection between components is needed to create the model, the Modelica equation **connect** is used.

B. Mapping Component Class Model

A CIM class describing conducting or regulating equipment is mapped with a **component** tag (see Fig. **4**). The design of the mapping will consider the attributes from CIM classes with *parameter* variability when doing the translation into Modelica. A *component* tag contains the tag-attribute **name**, mapping the CIM name of a device, and the tag-attribute **stereotype**, mapping the kind of Modelica object. The component tag contains an **attribute** tag, which maps the attributes contained in CIM class. The attribute is composed by a **name**, for the CIM attribute name, a **datatype**, for the data type of the CIM attribute, *variability* and *visibility*. The tag **value** indicates the value from the CIM attribute.

C. Mapping Connector Model

Topology CIM classes such as **Terminal**, **SvPowerFlow** and **SvVoltage** should be treated differently from other CIM classes. Terminal class is analogous to a **connector** stereotype. The connector is a kind of component which does not allow equations in it. This kind of components has variables which specify flow quantities, analogous to Kirchhoff's Current Law. The variables which do not specify the flow prefix are analogous to Kirchhoff's Voltage Law. A **connector** tag (see Fig. **5**) has the same definition as the **component** tag, with the addition of the tag-attribute **flow**, to indicate if the variable is designed for the flow of data or not.

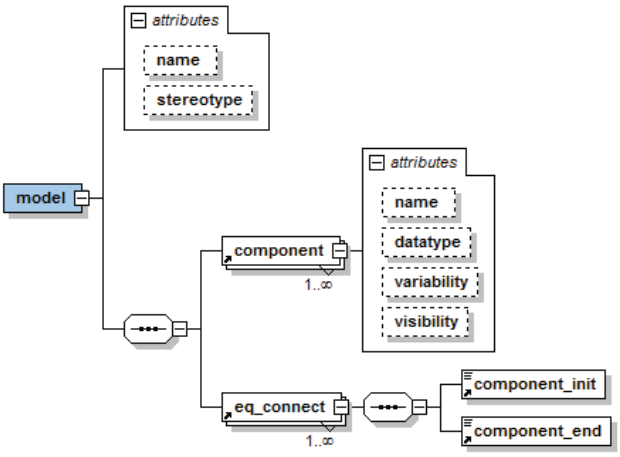

Fig. 3. Mapping structure for a top level power system model.

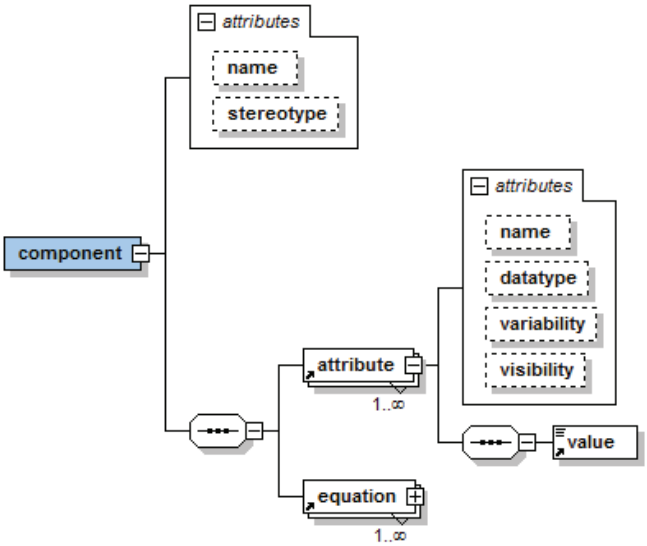

Fig. 4. Mapping structure for power system components.

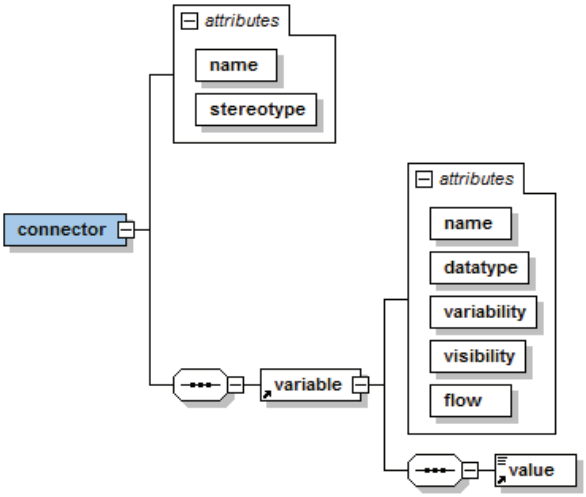

Fig. 5. Mapping structure for connectors.

D. Mapping implementation

The mapping application follows the Factory method design. These classes store the values and methods for generating the corresponding Modelica code. The ModMapping class is the responsible for loading the proper XML mapping schema. Creating a *loadMapping* function, the values from the CIM model are loaded into the XML map. With the *createModel* method, the XML map is used for creating the internal classes for generating Modelica modes, components and connectors.

The use of the Factory method allows the mapping to be scalable for further creation of code for other modeling languages. (see Fig. **6**). The following lines of code show an example of generation of Modelica code. The code the generation of a Modelica model for an ***ACLineSegment*** CIM class, with the *Terminal* and *SvPowerFlow* CIM values associated to it.

```
class ACLineSegment "ACLineSegment"
       parameter Real r=0.15 "Resistance";
       parameter Real x=0.05 "Reactance";
       parameter Real bch=0.1 "Susceptance";
       parameter Real gch=0.25 "Conductance";
       constant Real lenght=100.0 "Line lenght";
       flow Terminal pin(p=49.5,q=0.25) "Connector
Class";
       equation
end ACLineSegment;
```

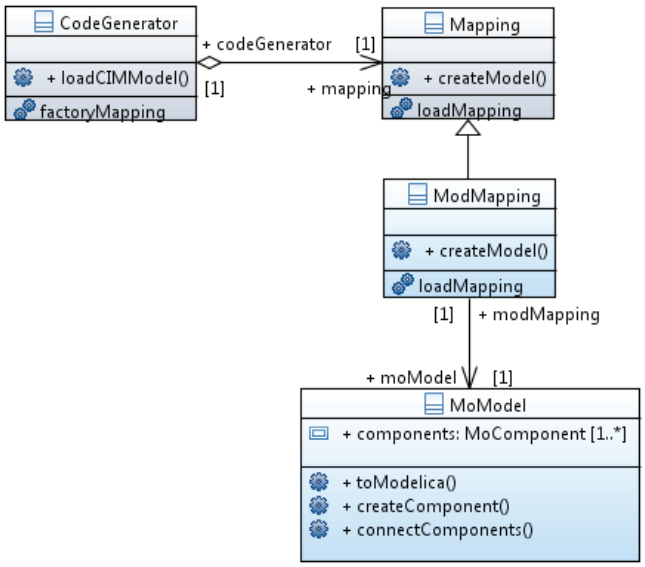

Fig. 6. Implementation of the Factory method. ModMapping class loads the XML mapping corresponding to the generation of Modelica models.

V. CONCLUSIONS AND FUTURE WORK

This work states the problem of initialization of Modelica power system models and describes the design of a mapping between CIM and Modelica languages for the information exchange and simulation of CIM-compliant power system dynamic models. The proposed mapping schema considers the generation of Modelica code using the names and values defined in the CIM data model.

The mapping offers a solution for assigning start values to continuous, discrete and algebraic state variables from a power flow solution stored in a CIM data model, and to generate the corresponding Modelica classes. Moreover, the implementation of the mapping will allow executing time-domain simulations of cyber-physical power system models, using Modelica compiler, directly from their CIM definition.

In order to perform the time-domain simulations, the proposed mapping needs to be completed. Further work will develop a solution for mapping the mathematical equations of power system components automatically. The proposed XML definition for the Modelica equations from the ModelicaXML schema [18] needs to be studied an adapted to this work in order to automatically generate the equations defined the iTesla Modelica Power System Library [19].

This work is part of the implementation of the Steady-State Solver Engine module, included in the open source modular simulation software architecture conceived for power system model validation [20].

REFERENCES

[1]. European Commission Third Energy Legislative Package. [On-line] Available:http://ec.europa.eu/energy/gas_electricity/legislation/third_legislative_package_en.htm

[2]. Regulation (EC) No 714/2009 of The European Parliament and of the Council of 13 July 2009. [On-line] Available: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2009:211:0015:0035:EN:PDF

[3]. CEN-CENELEC-ETSI Smart Grid Coordination Group. *Smart Grid Reference Architecture.* November, 2012. [On-line] Available: http://ec.europa.eu/energy/gas_electricity/smartgrids/doc/xpert_group1_reference_architecture.pdf

[4]. M. Uslar, M. Specht, S. Rohjans, J. Trefke, and J. M Gonzalez. *The Common Information Model CIM: IEC 61970, 61968 and 1078 62325*. Springer Heidelberg, 2012

[5]. ENTSO-E CIM Inter Operability Tests. [On-line] Available: https://www.entsoe.eu/major-projects/common-information-model-cim/interoperability-tests/Pages/default.aspx

[6]. IEC 61970-302: Common Information Model (CIM) for Dynamics Specification

[7]. IEC 61970-457: Common Information Model (CIM) for Dynamics Profile.

[8]. L. Vanfretti, W. Li, T. Bogodorova, and P. Panciatici, "Unambiguous power system dynamic modeling and simulation using modelica tools," *2013 IEEE Power and Energy Society General Meeting*, pp.1,5, 21-25 July 2013.

[9]. Unified Modeling Language Specification, OMG. (2001). [Online].Available: http://cgi.omg.org/docs/formal/01-09-67.pdf.

[10]. A.W. McMorran, G.W. Ault, C. Morgan, I.M. Elders, J.R. Mcdonald, "A Common Information Model (CIM) toolkit framework implemented in Java," *IEEE Transactions on Power Systems*, vol.21, no.1, pp.194,201, Feb. 2006.

[11]. P. Fritzson and P. Bunus, "Modelica, A General Object-Oriented Language for Continuous and Discrete-Event System Modeling," *In Proceedings of the 35th Annual Simulation Symposium*, 2002, pp. 14–18.

[12]. W. Schamai, P. Fritzson, C. Paredis, and A. Pop, "Towards Unified System Modeling and Simulation with ModelicaML: Modeling of Excecutable Behavior Using Graphical Notations," Proceedings of the 7th Modelica Conference, Como, Italy, Sep. 20-22, 2009,

[13]. S.E. Mattsson, H. Elmqvist, M. Otter and H.Olsson, "Initialization of Hybrid Differential-Algebraic Equations in Modelica 2.0", *Proceedings of the 2nd International Modelica Conference*, Oberpfaffenhofen, Germany, Mar 18-19, 2002.

[14]. B. Bachmann, P. Aronsson and P. Fritzson, "Robust Initialization of Differential Algebraic Equations" *In Proc. of Modelica '06*, Viena, Austria, Sep. 4-5, 2006.

[15]. L. Ochel, B. Bachmann, F. Casella, "Symbolic Initialization of Over-determined Higher-index Models", *Proceedings of the 10th International Modelica Conference*, Lund, Sweden, Mar 10-12, 2014.

[16]. Open Grid Systems. *Cimphony.* [On-line] Available: https://www.opengrid.com/content/cimphony/

[17]. Altova. XmlSpy, http://www.xmlspy.com

[18]. A. Pop and P. Fritzon, "ModelicaXML: A Modelica XML Representation with Applications", Proceedings of the 3rd International Modelica Conference, Linköping, Sweden, Nov 3-4, 2003.

[19]. T. Bogodorova, M. Sabate, G. Leon, L. Vanfretti, M. Halat, J.B. Heyberger and P. Panciatici, "A Modelica power system library for phasor-time domain simulation," *2013 4th IEEE/PES Innovative Smart Grid Technologies Europe*, pp.1,5, 6-9 Oct. 2013.

[20]. F. Gómez, L. Vanfretti, Svein H. Olsen, "A Modelica-BAsed Execution and Simulation Engine for Automated Power system Model Validation", Innovative Smart Grid Technologies (ISGT) Europe, Istanbul, Oct. 12-15, 2014