CIM-Compliant Power System Dynamic Model-to-Model Transformation and Modelica Simulation

Francisco J. Gómez, Luigi Vanfretti, and Svein Harald Olsen

***Abstract*—European regulations on information exchange have put new requirements on analysis tools, the main one being the adoption of the International Electrotechnical Commission common information model (CIM) that may help interoperability across applications. This paper proposes the use of model-driven software engineering methods to meet these new requirements. Specifically, this paper shows how to apply model-to-model (M2M) transformations. The M2M method presented herein allows us to work directly with the information and mathematical description and computer implementation of dynamic models, independent from specific analysis tools. The M2M method proposed requires the development of a mapping between CIM/unified modeling language and the Modelica language, which allows us to derive Modelica models of physical power systems for dynamic simulations.**

***Index Terms*—Common information model (CIM), information exchange, information modeling, Modelica, open instance power system library (OpenIPSL), power systems dynamics, systems modeling language (SysML), unified modeling language (UML).**

Nomenclature

CGMES	Common grid model exchange standard.
CIM	Common information model.
DSO	Distribution system operators.
ENTSO-E	European Network of Transmission System Operators in Electricity.
ERGEG	European Regulators' Group for Electricity and Gas.
EPRI	Electrical Power Research Institute.
IEC	International Electrotechnical Commission.
iTESLA	Innovative tools for electrical system security within large areas.
MDD	Model-driven development.
MDSE	Model-driven software engineering.
M2M	Model to model.
M2T	Model to text.
OMG	Object management group.
OpenIPSL	Open instance power system library.
RDF	Resource document format.
SysML	Systems modeling language.
TSO	Transmission system operators.
T2T	Text to text.
UML	Unified modeling language.
W3C	World wide web consortium.
XML	eXtensible modeling language.

Manuscript received November 8, 2017; accepted December 8, 2017. Date of publication December 20, 2017; date of current version September 4, 2018. Paper no. TII-17-2620. *(Corresponding author: Francisco J. Gómez.)*

F. J. Gómez is with the Electric Power and Energy Systems Department, Royal Institute of Technology, Stockholm 10044, Sweden (e-mail: fragom@kth.se).

L. Vanfretti is with the Electrical, Computer and Systems Engineering Department, Renssellaer Polytechnic Institute, Troy, NY 12180 USA (e-mail: luigi.vanfretti@gmail.com).

S. H. Olsen is with Statnett SF, Oslo 0423, Norway (e-mail: svein.harald.olsen@statnett.no).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2017.2785439

I. Introduction

A. Motivation

EFFICIENT information exchange between transmission system operators (TSO), distribution systems operators (DSO), and generation companies is required for network planning and power systems operations, e.g., real-time monitoring and steady-state analysis. These functions demand a high degree of coordination and consistency in data exchanges, which can be significantly streamlined through a common data exchange standard. The European Network of Transmission System Operators for Electricity (ENTSO-E) has adopted regulation EC 714/2009 for cross-border exchanges in the electricity network to ensure coordination of "data exchange and settlement rules, network security and reliability rules, interoperability (IOP) rules, and transparency rules" [1].

To comply with these rules, software tools are one of the primary means to perform analysis and coordination tasks. The International Electrotechnical Commission (IEC) Technical Committee 57 (TC57) Working Group 13 and the Electrical Power Research Institute have been working on the development of the common information model (CIM) to provide standard modeling semantics for power systems information exchange [2].

While the IEC CIM provides a solid basis for information exchange, it does not any guarantee on the model simulated response. The IEC CIM for dynamic defines a relevant standard

This work is licensed under a Creative Commons Attribution 3.0 License. For more information, see http://creativecommons.org/licenses/by/3.0/

model with parameters for exchange. However, the physical dynamic behavior is only represented in a block diagram rather than in mathematical form (equations). In other words, the lack of a strict mathematical description defining each of the power system models is not exchanged (only parameters and pictorial diagrams). Thus, IOP can only be unassailable though the comparison of several implementation results.

B. Previous Works

The authors' recent work has shown how the use of the Modelica language can preserve consistency of the behavior exposed by power system dynamic models when simulated in different tools [3]. In addition, Gómez *et al.* in [4] a model execution engine leveraging Modelica tools, showing that commercial Modelica compilers, such as Dymola, and open-source Modelica compilers, such as OpenModelica, can be used for power systems model simulation and analysis.

Open standard modeling languages for simulation, such as Modelica [5], can provide the mathematical description of physical systems, which is attractive to exchange user-defined models. ModelicaML [6] helps us to link system modeling language (SysML)-based models to derive Modelica code. This allows us to derive equation-based models of entire systems from the mathematical description of each component and their interconnections (system structure).

From the above, it becomes evident that means to harmonize and interface the different information modeling and physical modeling computer languages can be very attractive to support power system model exchange and dynamic applications. A detailed account of these languages, model-driven software engineering (MDSE) principles, and other background is described in Section II.

C. Contributions

To address the issues outlined above, this paper provides three main contributions.

1) It proposes modeling process that CIM/unified modeling language (UML) and SysML to generated Modelica simulation models.
2) It defines a method to complement CIM dynamics profile (DY) with equation-based component model definitions for physical modeling behavior, by using the Modelica language, for dynamic simulation analysis.
3) It defines of a scalable, modular, and reusable mapping between different modeling semantics for model-to-model (M2M) transformation (UML/CIM, SysML/Modelica).

To show the detailed description of these contributions, this paper is organized as follows. In Section II, the authors describe the state of the art on modeling languages for information exchange and make a detailed description of the proposed model transformation methodology. Section III describes in detail the mapping between the two modeling languages. Section IV describes the results of the application of the proposed methodology, and in Section V, the authors present the conclusions of this paper.

TABLE I
MODEL TRANSFORMATION TECHNIQUES

T2T	Transformation of origin data or input program code into documentation or program code, which can be executed by a machine.
M2T	Transformation of an input model, implemented in a modeling language, such UML, into program code or documentation. This transformation can be also interpreted as code generation.
M2M	Transformation of source model, defined with a modeling language, into a target model, defined in a different or the same modeling language, without compiling the target model into program code.

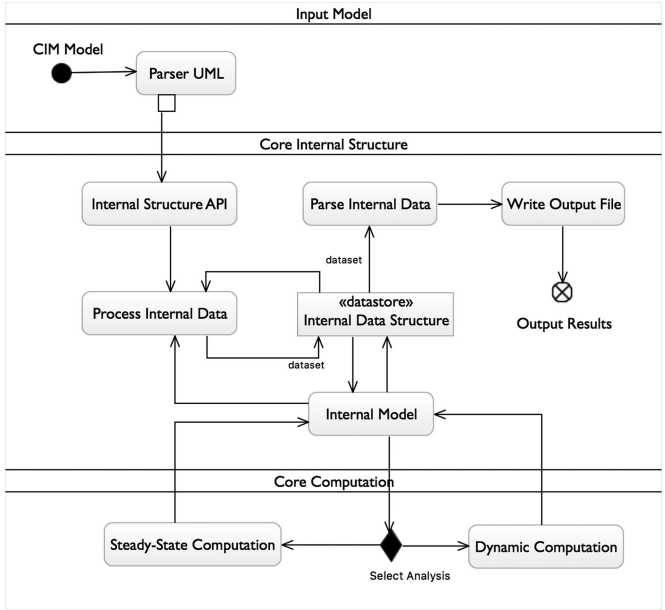

Fig. 1. UML representation of the workflow of power systems analysis tools, which have the capability to adopt the M2T approach. The input models of these tools are CIM/UML models.

II. BACKGROUND

A. Model-to-Model Transformations

MDSE is a subset of model-driven development (MDD) that consists of a development paradigm based on models, which relies on the use of object management group (OMG) standards.

This study uses OMG standards, such as UML and SysML, for describing a MDSE methodology for model transformation and dynamic simulation. The software 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 [7]. Table I shows a description of those transformation techniques.

Most power system analysis tools only implement T2T transformations, known in the power system domain as "parsers" or "data file format converters/filters." Recently, with the goal of adopting the CIM standards, some power system analysis tools implement software interfaces, for M2T transformation, to read CIM files into the tools' internal proprietary tool data format and representation, as illustrated in Fig. 1 (see [20] for detailed example of such approach).

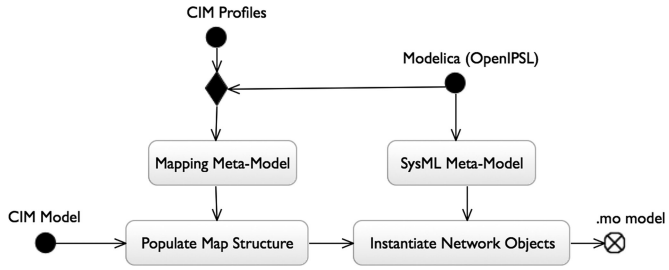

Fig. 2. UML representation of the high-level workflow for M2M transformation between CIM and a target modeling language, such as Modelica.

In this paper, the M2M transformation is built on [8], with a focus on the UML and SysML semantics representations from CIM and the Modelica language. The workflow in Fig. 2 represents a transformation process between CIM and Modelica. As shown in Fig. 2 mapping rules are used for reading a source model in CIM, and the creation of a SysML Modelica-based class structure (as a metamodel). The metamodel is used to create the target model in Modelica that will be used for dynamic simulations.

B. UML and SysML

The UML is one of the OMG ISO standard specifications for system modeling [9]. UML plays an important role for MDD providing a standardized modeling language to create models for software development. The UML consists in a set of model elements representing an analysis of the properties and behavior of the system.

Those elements are classified in the following categories: *Classifiers*, describing a set of objects; *events*, describing a set of occurrences; and *behaviors*, describing a set of possible executions; all are represented by the UML semantics that give a standard interpretation of the system being modeled.

UML semantics are classified in two categories: Structural, defining the meaning of the model's elements of an engineering domain at specific point in time, i.e., within a class diagram representation; and behavioral, defining model elements that change over time, i.e., within a sequence diagram representation.

The OMG defines an extension of UML for modeling of system engineering applications that considers hardware, software, information, and processes. The SysML [10] provides additional design principles, such as requirements modeling and reuse of UML class diagrams as block diagrams supporting parametric modeling IOP, etc.

In this study, two representations of UML and SysML semantics are used for power system modeling.

C. Standards in ENTSO-E; CIM and Common Grid Model Exchange Standard (CGMES)

The coordination of TSO in planning and operation procedures, such as system security, capacity calculation, and outage planning, was not sufficient to avoid a large system disturbance on 2006, affecting major parts of Europe [11]. After these events, the European Regulators' Group for Electricity and Gas defined a set of recommendations stating the need for compliance and consistency and highlighting the importance of the interdependencies and information exchange of the national implementations of grid models, which refers to both static and dynamic information upon which these models are implemented.

Within this context, the ENTSO-E has made large efforts in adoption of the CIM to improve the means for information exchange between TSO. ENTSO-E has adopted different IEC CIM profiles to comply with the regulation and mandates [12], and, thus, the CGMES [13] was approved. Conformity of modeling, simulation, and analysis tools to CGMES is carried out through IOP tests [14]. The CGMES reflects current TSO requirements for accurate modeling of the ENTSO-E area for power flow, short-circuit computations, and dynamic simulations reflected within the following standards.

1) IEC 61970-552 CIM eXtensible markup language (XML) model exchange format define language style and implementation rules of the data syntax of CIM.
2) IEC 61970-301 CIM base defines a static information UML package containing UML semantics for the physical characteristics of the power network and electrical and nonelectrical characteristics of equipment static models.
3) IEC 61970-302 CIM for dynamics specification and IEC 61970-457 CIM for DY, define a dynamic information UML package containing the UML semantics for the dynamic characteristics of equipment models.

D. CIM for Power System Dynamic Analyses

The CIM covers the needs for power network analysis studies and is based on UML. The CIM semantic representation defines all the basic components and topology of the power network, with its steady-state behavior and a *limited description on its dynamics*. CIM uses UML's structural semantics, defining sets of classes, attributes, and binding rules among classes, to represent physical objects and their properties. The relation of classes is represented within different UML class diagrams that classify the network's information into different packages. The set of these packages constitute the whole CIM canonical model.

The CIM canonical model is represented under the IEC 61970 standard covering base power system models (IEC 619170-301) and base dynamic models (IEC 67190-302) [15]. The CGMES covers different subsets of this standard.

1) The equipment profile (EQ) (based on IEC 61970-452) that defines classes and attributes with basic information of equipment models.
2) The topology profile (TP) (based on IEC 61970-456) describes how a model is connected. It contains information of terminal classes and their relation with ConductingEquipment and TopologicalNode classes.
3) The state variable (SV) profile (based on IEC 61970-456) is the result from a topology processor and power flow calculations.
4) The DY (based on IEC 61970-302 and IEC 61970-457) defines the classes and attributes for the behavior of the equipment.[1]

[1]These attributes will be used in this study to assign parameters to each of the mathematical model equations that are defined in Modelica components.

The implementation of these profiles is performed by using semantic web language resource description framework (RDF) defined by W3C [16]. The serialization syntax is RDF/XML, which are extensions of the XML. In this study, the RDF/XML implementation is exploited to select the most relevant model information.

E. Modelica for Power Systems

The modeling of power system equipment and networks does not end with the definition of attributes. The mathematical representation of these models must be developed to ensure unambiguous modeling, consistent exchange of the information to represent dynamic behavior, and to perform consistent simulations in different tools. The Modelica language satisfies the requirements [3] and, in addition, it can be described with SysML semantics [6].

The CIM for DY includes the definition of block diagrams of correspondent network component and their attributes. While this is very useful for humans to exchange knowledge, a computer-readable description would require a unique mathematical equation specification to guarantee consistency [3].

Modelica is an object-oriented equation-based programming and modeling language, which allows the representation of cyber-physical systems using a strict and openly standardized mathematical representation for dynamic modeling and simulation. The authors' previous works in the FP7 iTESLA project has used Modelica for power system modeling [4], [8], [17]–[20], and Modelica-compliant tools have been used for simulations and compared with proprietary simulation tools [17]–[19]. One result of this project is the Modelica library containing dynamic for power systems component models [3], the OpenIPSL.

Modelica is suitable for unambiguous modeling and simulation [17]. A model in Modelica is totally decoupled from the mathematical solver that is used to provide a numerical solution of the model equations [3]. This characteristic guarantees an unambiguous way of modeling and simulation. Modelica models need to be provided with adequate starting guess values to perform simulations. OpenIPSL uses as starting guess the power flow solution values and Modelica tools solvers for the initial conditions of the entire dynamic model (for all algebraic and dynamic equations) to perform dynamic simulations. The challenges related to model initialization are discussed in [20] and [21].

III. CIM-to-Modelica Model Transformation

To have a complete representation of a physical model, the use of the OpenIPSL Modelica components can complement the CIM/CGMES models. This study proposes *the design of a M2M workflow* to generate power system Modelica models from CIM, as shown as in Fig. 2. The design includes the development of mapping rules between CIM/UML and Modelica syntaxes, and the design of a SysML-based metamodel for generating the resulting Modelica network model. Each of these mapping rules have been prototyped individually using XML, a portion of the code for one design is shown in Fig. 3 (others are omitted due to space constraints).

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE model SYSTEM "openipsl_component_map.dtd">
<pwPinMap cim_name="class_name" rfd_id="" rfd_resource=""
          package="openipsl_package_path" name="openipsl_component_name"
          stereotype="connector">
<attributeMap cim_name="attribute_cim_name"
          name="openipsl_parameter_name"
          datatype="Real" variability="parameter" visibility="public"
          flow="0"> value </attributeMap>
<attributeMap cim_name="attribute_cim_name"
          name=" openipsl_parameter_name"
          datatype="Real" variability="none" visibility="public"
          flow="1"> value </attributeMap>
</pwPinMap>
```

Fig. 3. Sample of the XML code with the mapping rules between CIM and the OpenIPSL Modelica library.

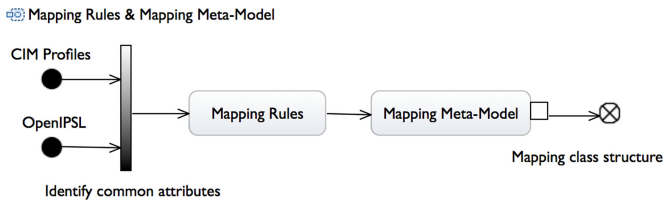

Fig. 4. Workflow for creating the mapping rules and the class structure implementing those rules.

A. Binding Semantics: Mapping Metamodel

The CIM semantics and Modelica language are implemented using different syntaxes, thus it is important to identify the key features for binding both modeling languages. The workflow for the creation of a metamodel class structure based on the mapping rules is shown in Fig. 4.

The mapping matches the CIM classes and attributes names with the component names and their parameter names from the OpenIPSL library models. The action *Mapping Rules* define the creation of matching criteria of CIM classes and OpenIPSL components using XML for each target component model.

The mapping is enhanced by including object's references from the CIM model, to be used by the M2M transformation process for populating these IDs, to be used to identify relations between classes, as follows.

1) The *rdf:ID* attribute contains a unique ID, which is used to identify the object from the CIM model.
2) The *rdf:resource* is a pointer to another object within the CIM model. Its value is other rfd:IDs and that will be used to identify the other objects related to the current object.

The *Mapping Metamodel* action uses the mapping rules to create a class structure that encapsulates the mapping rules.

These mapping rules are manually implemented for each new power system component to be modeled with CIM and the OpenIPSL (see Fig. 3). The *Mapping Metamodel* action in Fig. 4 creates a new class from the XML definition of the new mapping rules using a JAVA parser for XML [22]. The result of this action is a *Mapping Class Structure*, shown in Fig. 5, which is used to populate the target model values from a CIM model, as follows.

1) The classes *ComponentMap* and *AttributeMap* contain the key attributes to identify the CIM classes and CIM attributes to read from the CIM model, and their correspondent key attributes in the Modelica component.

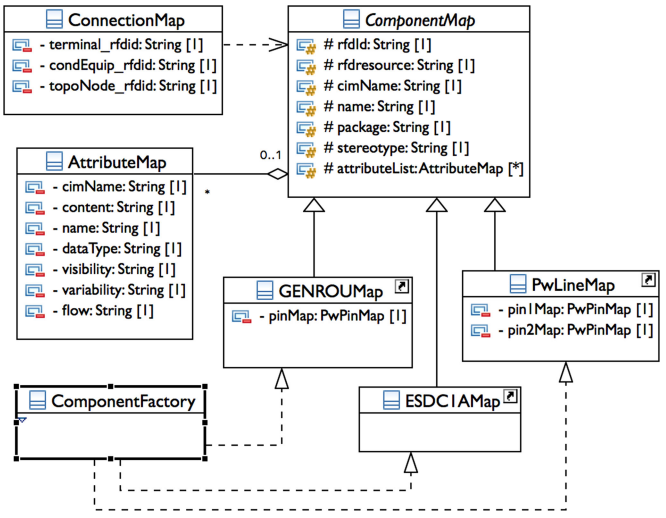

Fig. 5. UML class diagram shows the design of a metamodel structure of the mapping containing either both CIM classes attributes and Modelica language keywords and OpenIPSL names.

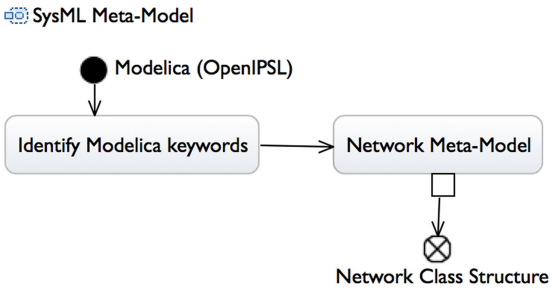

Fig. 6. Workflow for creating a class structure based on Modelica syntax and SysML semantics.

2) The values from the CIM model are populated within these two classes.
3) An additional class, *ConnectionMap* class, contains the references of the CIM classes *Terminal, ConductingEquipment*, and *TopologicalNode*.
4) *ConnectionMap* is used to populate the references of these classes and to create the connections between components.

B. Building a Power Network's SysML Metamodel

A second workflow shows the process of identification of the target language syntaxes to be used to create the instances of the power system components models and power networks (see Fig. 6). This workflow considers basic characteristics from the Modelica language such as declaration of model parameters and declaration of components' connection.

The *Network Metamodel* action in Fig. 6, based on syntax definitions of the Modelica language, uses the Modelica class stereotypes *Model*, *Class,* and *Connector* to differentiate which component should be created and define its place into the model hierarchy for a network model (see Fig. 7). The *network class structure,* considers a model hierarchy based on a high-level model e.g., the *network*, medium-level models, e.g., a *component*, and low-level models, e.g., a *connector*, and its implementation is depicted in Fig. 8.

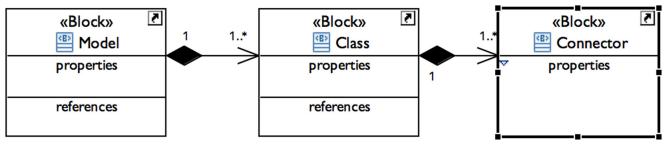

Fig. 7. This block diagram represents the three Modelica class stereotypes defined as blocks.

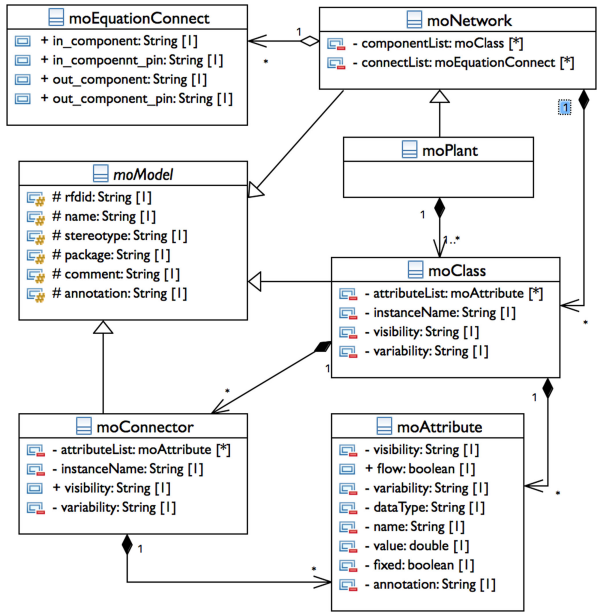

Fig. 8. UML class diagram shows the design of the proposed internal structure for creating the Modelica code. It defines Modelica names and keywords that will be used for generating the final network model.

The *Network Class Structure* complements the *Mapping Class Structure* and is used in the proposed M2M transformation to create the network model with the instances of the OpenIPSL Modelica library components and their connections, as follows:=.

1) Classes *moModel*, *moClass*, and *moConnector* correspond the Modelica stereotypes *model, class*, and *connector*.[2]

The metamodel structure includes two additional classes for defining high-level models within a network model.

1) The class *moPlant* encapsulates the components that are included in the model of a plant, e.g., machines and regulator components.
2) The *moNetwork* plant encapsulates both plant objects and component objects and defines the whole network structure.
3) *moClass* and *moConnector* will be used with values from the mapping to create the instances of the OpenIPSL objects. The class *moAttribute* stores the values of the parameters of each component.

[2]Because the words model, class, and connector might be reserved keywords in different programming languages, we add the prefix mo in the name of the classes.

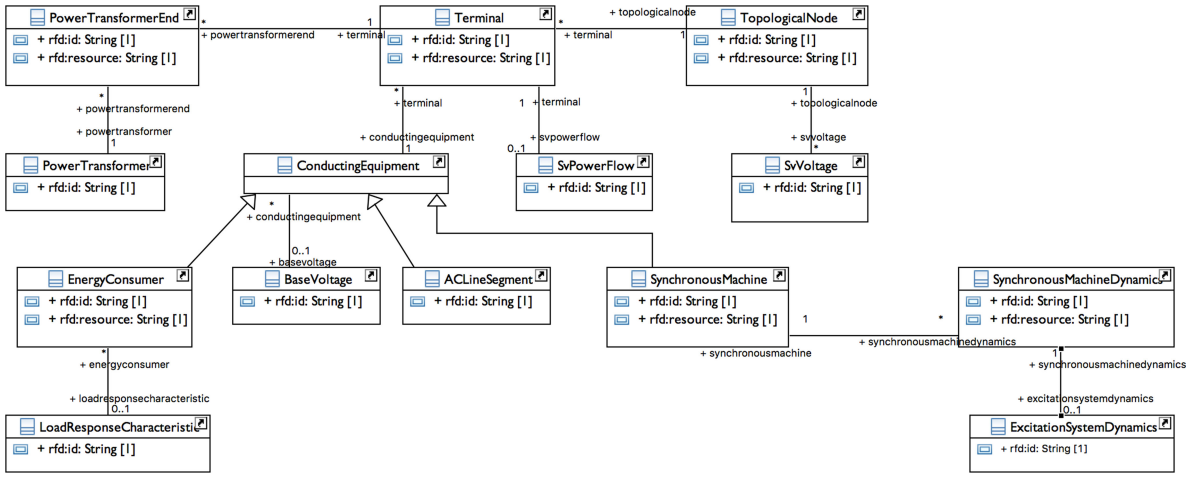

Fig. 9. Hierarchy of CIM classes indicates the classes from the CIM model read by the M2M transformation workflow. Line without arrow indicates class association, with the multiplicity at each end of the association, and full arrows indicate generalization.

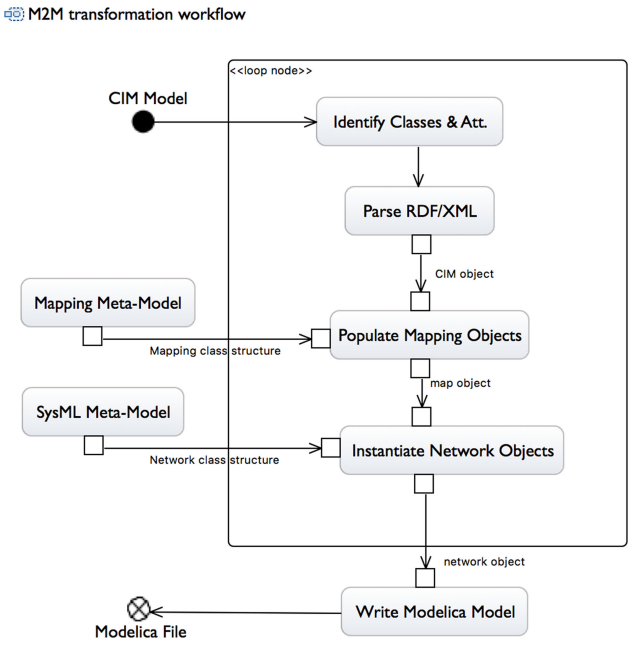

Fig. 10. Activity diagram depicts the workflow and actions for the proposed M2M transformation process.

IV. Results: Model-To-Model Transformation and Simulation

The main workflow of the proposed M2M transformation uses the structures defined within the *Mapping Metamodel* and *SysML Metamodel* workflows from Sections III-A and III-B to automatically generate the target Modelica models from the source CIM model.

A. Workflow Execution

The detailed actions within the main workflow are depicted in Fig. 10. It starts with an input CIM model that contains classes and attributes with equipment and topology information of the network (see the class hierarchy on Fig. 9). The next action of the workflow, *Parse RDF/XML*, identifies the CIM classes and attributes, and parses the CIM/RDF implementation into memory. Then, the *Populate Mapping Objects* action populates these values into the mapping structure, executing the following steps.

1) The values from the CIM model are read starting from a *Terminal* class and then from the classes related to that class.
2) With the rfd:ids and resource:ids attributes from CIM, the classes *PowerTransformerEnd, ConductingEquipment, TopologicalNode*, and *SvPowerFlow* are identified.
3) In case of a *ConductingEquipment* class is found, *EnergyConsumer, ACLineSegment, SynchronousMachine* are identified.
4) For *PowerTransformer*, first the *PowerTransformerEnd* class is identified and then its related *Terminal* class.

The values from the *Mapping Metamodel* object are used to create the instances of the corresponding OpenIPSL components. The action *Instantiate Network Objects* create the network structure adding the component objects and connections to the model, with the following steps.

1) The CIM *Terminal* class is transformed into an OpenIPSL *Pin* component, using the *moConnector* class in Fig. 8.
2) The specific CIM *ConductingEquipment* class is transformed into the corresponding *OpenIPSL component*, such as machine component or line component, using the *moClass* class in Fig. 8.
3) Following the instance definition of the components, *the* values of the CIM *SvPowerFlow* and *SvVoltage* classes are used to set the component instance's parameters initial guess values.
4) The CIM *TopologicalNode* class is transformed into a *OpenIPSL Bus* component, with the use of the *moClass* in Fig. 8.
5) The connection between components are created using the *moEquationConnect* class in Fig. 8. It contains the component name and its pin and the name and pin of the associated bus.

These workflow is executed within a loop, until the whole CIM model has been processed.

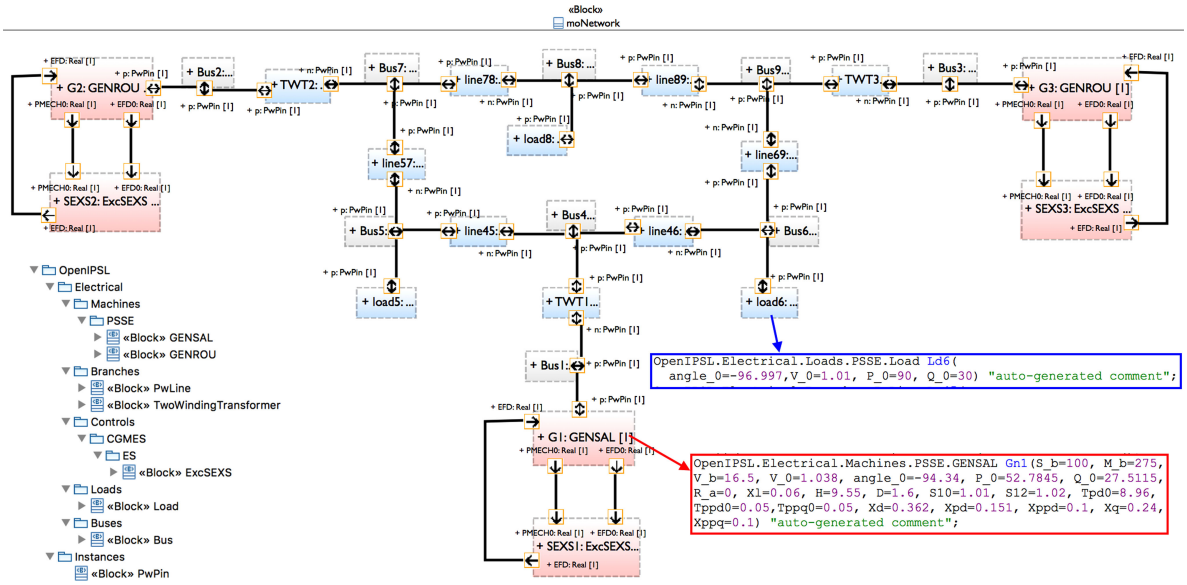

Fig. 11. SysML diagram of the 9-bus system. In the bottom left corner, there is the hierarchy of OpenIPSL power system components that form the network. In the bottom right corner, there is the Modelica instance implementation of load and machine components.

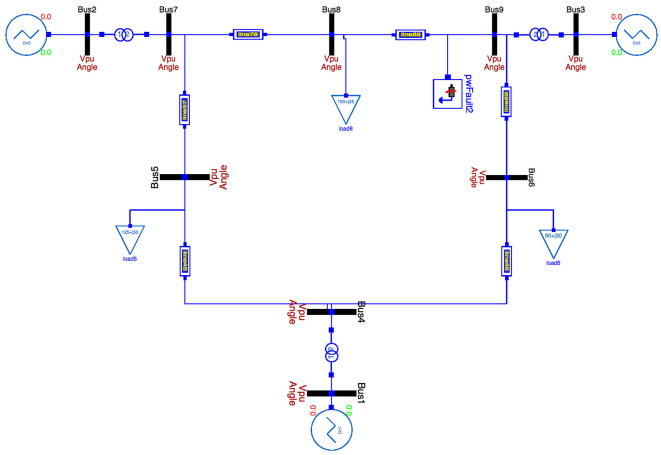

Fig. 12. Modelica diagram of the 9-bus system. Graphical view of the autogenerated model.

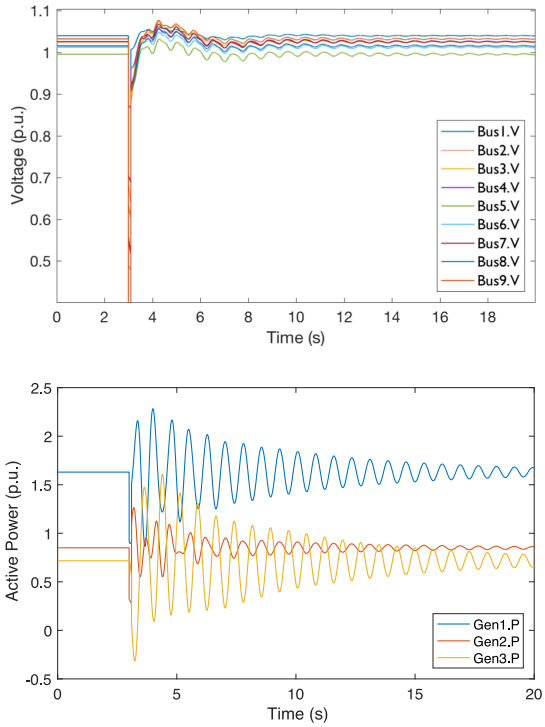

Fig. 13. Simulation results of the simulation generated model, IEEE 9-bus model, first: Voltages (p.u.) at each bus; second: Active power (p.u.) at each generator.

B. Simulation Model and M2M Results

To test the proposed M2M transformation process, the IEEE 9-Bus model [23] is used as input CIM model. The system is composed by three generators operating at 275 MVA and 16.5 kV in generator 1; 320 MVA and 18 kV at generator 2; 300 MVA and 13 kV at generator 3, with a system base power of 100 MVA. This input model is implemented in four different XML/RDF files, which contain information from the EQ, TP, SV, and DY.

Using the class structures described in Section III, the M2M transformation process workflow (see Fig. 10) converts the CIM information model of the IEEE 9-Bus system into the corresponding Modelica mathematical model. The target network model is shown in Fig. 11, which shows the instances of OpenIPSL components and the topology of the network.

For simulation purposes, a fault has been added to the model to show better the dynamic behavior of the model. The fault is applied at Bus 8 in the system (see Fig. 12),[3] and the buses voltages (V) and generators active power (P) are plotted in Fig. 13.

The reason to split the mapping into two different structures follows a modular implementation of the proposed methodology. The *Mapping Class Structure* of Section III-A can be reused

[3]The fault component model is manually instantiated in the parameter section of the model and manually connected to the power network, in the equation section.

to convert the CIM into another modeling language, while the *Network Class Structure* in Section III-B can be reused to generate Modelica code from a different source modeling language or from other simulation tools. The SysML representation of the network (see Fig. 11) shows model design independent from the modeling language or the power system simulation and modeling tool. The target model design can also be represented as a block diagram in Modelica (see Fig. 12).

V. Conclusion and Discussion

The complete process of binding the CIM syntax and Modelica syntax presented in this paper gives a mechanism to use mathematical modeling language as a complement for the CIM dynamics information exchange profile, and its CGMES extension. The use of Modelica for the modeling of power systems dynamic behavior allows us to comply with the application of EU rules for security, IOP, and transparency established by European regulations [8]. This study gives a proof of concept of how to follow Annex F in the new CGMES 2.5 standard guidelines [14], corresponding to the IEC 61970-600, to use Modelica models as a means to exchange user-defined models defined in a Modelica library, and serves as an example of the feasibility of the approach recommended by this Annex F.

The results from the workflow process show that the proposed method is capable of transform a CIM/RDF classes and attributes of the network model into the OpenIPSL Modelica components and attributes, which contains the corresponding mathematical behavior of the network components. The resulting model can be used by different Modelica compilers, which compile the Modelica language into executable programing code. This implementation guarantees the scalability, reusability, and modularity of the proposed methodology.

With this solution, dynamic analysis, such as time-domain simulations, can be simulated by utilizing Modelica compilers that have proved to be applicable for this kind of analysis in different fields [24] and are showing promise for continental-level power systems simulations [25].

The results of this workflow show how to build a power systems network model with the use of CIM information and OpenIPSL power system components. To show the dynamic behavior of the network with the presence of a fault, a fault model has been implemented manually into the output Modelica model. To automatically add events, such as line trips, other CIM profiles [26] need to be supported in the proposed workflow.

References

[1] C. Ivanov, T. Saxton, J. Waight, M. Monti, and G. Robinson, "Prescription for interoperability: Power system challenges and requirements for interoperable solutions," *IEEE Power Energy Mag.*, vol. 14, no. 1, pp. 30–39, Jan./Feb. 2016, doi: 10.1109/MPE.2015.2485798.

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

[3] L. Vanfretti, T. Rabuzin, M. Baudette, and M. Murad, "iTesla power systems library (iPSL): A Modelica library for phasor time-domain simulations." *SoftwareX*. (2016, May 18). [Online]. Available: http://dx.doi.org/10.1016/j.softx.2016.05.001

[4] F. Gómez, L. Vanfretti, and S. H. Olsen, "A Modelica-based execution and simulation engine for automated power systems model validation," in *Proc. Innov. Smart Grid Technol. Eur.*, Istanbul, Turkey, Oct. 12–15, 2014, doi: 10.1109/ISGTEurope.2014.7028828

[5] P. Fritzson, *Introduction to Modeling and Simulation of Technical and Physical Systems With Modelica*. New York, NY, USA: Wiley-IEEE Press, 2011.

[6] W. Schamai, P. Fritzson, C. Paredis, and A. Pop, "Towards unified system modeling and simulation with ModelicaML: Modeling of excecutable behavior using graphical notations," in *Proc. 7th Modelica Conf.*, Como, Italy, pp. 612–622, Sep. 20–22, 2009.

[7] M. Brambilla, J. Cabot, and M. Wimmer, *Model-Driven Software Engineering (MDSE) in Practice*. San Rafael, CA, USA: Morgan & Claypool, 2012, pp. 9–11, ISBN: 978-1-608-45882-0.

[8] F. J. Gómez, L. Vanfretti, and S. H. Olsen, "Binding CIM and Modelica for consistent power system dynamic model exchange and simulation," in *Proc. 2015 IEEE Power Energy Soc. Gen. Meet.*, Denver, CO, USA, 2015, doi: 10.1109/PESGM.2015.7286434.

[9] OMG, Needham, MA, USA. *Unified Modeling Language Specification*. (2015) [Online]. Available: http://www.omg.org/spec/UML/2.5/

[10] OMG, Needham, MA, USA. *Systems Modeling Language Specification*. (2015) [Online]. Available: http://www.omg.org/spec/SysML/1.4/

[11] *Europe's National Regulators of Electricity and Gas at EU and International Level*. [Online]. Available: http://www.ceer.eu/portal/page/portal/EER_HOME/EER_PUBLICATIONS/CEER_PAPERS/Electricity/2007/E06-BAG-01-06_Blackout-FinalReport_2007-02-06.pdf

[12] *CEN-CENELEC-ETSI Smart Grid Coordination Group*. *Smart* Grid Reference Architecture. (2012, Nov.) [Online] Available: http://ec.europa.eu/energy/gas_electricity/smartgrids/doc/xpert_group1_reference_architecture.pdf

[13] *ENTSO-E Common Grid Model Exchange Standard (CGMES)*. [Online]. Available: https://www.entsoe.eu/major-projects/common-information-model-cim/cim-for-grid-models-exchange/standards/Pages/default.aspx

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

[15] L. O. Osterlund *et al.*, "Under the hood: An overview of the common information model data exchanges," *IEEE Power Energy Mag.*, vol. 14, no. 1, pp. 68–82, Jan./Feb. 2016, doi: 10.1109/MPE.2015.2485859.

[16] *Resource Description Framework for Semantic Web*. [Online]. Available: https://www.w3.org/RDF/

[17] L. Vanfretti, W. Li, T. Bogodorova, and P. Panciatici, "Unambiguous power system dynamic modeling and simulation using Modelica tools," in *Proc. 2013 IEEE Power Energy Soc. Gen. Meet.*, Jul. 21–25, 2013, doi: 10.1109/PESMG.2013.6672476.

[18] T. Bogodorova *et al.*, "A Modelica power system library for phasor-time domain simulation," in *Proc. 4th IEEE/PES Innov. Smart Grid Technol. Eur.*, Oct. 2013, doi: 10.1109/ISGTEurope.2013.6695422.

[19] G. León, M. Halat, M. Sabaté, J. B. Heyberger, F. J. Gómez, and L. Vanfretti, "Aspects of power system modeling, initialization and simulation using the Modelica language," in *Proc. 2015 IEEE PowerTech Conf.*, Eindhoven, The Netherlands, 2015, doi: 10.1109/PTC.2015.7232504.

[20] L. Vanfretti *et al.*, "Towards automated power system model transformation for multi-TSO phasor time domain simulations using modelica," in *Proc. IEEE PES Innov. Smart Grid Technol. Conf. Eur.*, Ljubljana, Slovenia, 2016, doi: 10.1109/ISGTEurope.2016.7856341.

[21] L. Ochel, B. Bachmann, and F. Casella, "Symbolic initialization of over-determined higher-index models," in *Proc. 10th Int. Modelica Conf.*, Lund, Sweden, 2014, pp. 1179–1187.

[22] *Java Architecture for XML Building (JAXB)*, 2003. [Online]. Available: http://www.oracle.com/technetwork/articles/javase/index-140168.html

[23] The Illinois Center for a Smarter Electric Grid (ICSEG), Urbana, IL, USA, "WSCC 9-Bus System," *Power Cases*, [Accessed Apr. 11, 2015]. [Online]. Available: http://publish.illinois.edu/smartergrid/wscc-9-bus-system/

[24] F. Casella, "Simulation of large-scale models in Modelica: State of the art and future perspectives," in *Proc. 11th Int. Modelica Conf.*, Versailles, France, Sep. 21–23, 2015, pp. 459–468.

[25] F. Casella, A. Bartolini, S. Pasquini, and L. Bonuglia, "Object-oriented modelling and simulation of large-scale electrical power systems using modelica: A first feasibility study," in *Proc. 42nd Annu. IEEE Conf. Ind. Electron. Soc.*, Firenze, Italy, Oct. 24–27, 2016, pp. 6298–6304.

[26] *System Integrity Protection Schemes (SIPS)*, 2016. [Online]. Available: http://cimug.ucaiug.org/Groups/ENTSO-E_IOP/2016/Shared%20Documents/Reference%20Documents/20160714_CGMES_IOP_SIPS.pdf