Towards Automated Power System Model Transformation for Multi-TSO Phasor Time Domain Simulations using Modelica

Luigi Vanfretti,
M. Ahsan Adib Murad,
Francisco Gómez López
KTH Royal Inst. Of Tech.
Stockholm, Sweden
luigiv@kth.se, maamurad@kth.se,
fragom@kth.se

Gladys León,
Silvia Machado
Aplicaciones en Informática Avanzada
S. Cugat del Vallés, Spain
leonge@aia.es, machados@aia.es

Jean-Baptiste Heyberger,
Sebastien Petitrenaud
Résau de Transport d'Électricité
Paris, France
jean-baptiste.heyberger@rte-france.com,
sebastien.petitrenaud@rte-france.com

***Abstract*—Transmission system operators in Europe describe their dynamic power system models using different simulation tools, and due to the de-facto modeling philosophy used, these descriptions are ambiguous between tools. In addition the current CIM standard for dynamic model exchange does not guarantee consistency when exchanging models. This poses a challenge to perform pan-European dynamic security assessment. This paper presents a method for transforming power system model descriptions typically used by TSOs into a consistent and unambiguous equation based modeling language. As a result, this method allows performing simulations in multiple tools supporting the standardized Modelica language. The transformation method is validated by steady state and dynamic simulations and comparing simulation outputs between a reference tool (PSS/E or Eurostag) and a Modelica tool. It is shown that the Modelica language can be used as a common language to provide unambiguous model descriptions consistent with those tools typically used by TSOs, without loss of information and maintaining simulation fidelity.**

***Index Terms*-- Modelica, Open source software, power system simulation, dynamic security assessment, CIM.**

I. INTRODUCTION

One of the major goals of the EU Framework Project iTESLA [1] is to perform pan-European dynamic security assessment, which requires executing time domain simulations considering a large number of contingencies, in order to assess the security of the grid including its dynamic performance. However, each European TSO has its own dynamic model usually expressed in the proprietary format of the software tools used to run their own dynamic studies. As a consequence, the utilization of a common and standardized language for modelling power system dynamics is necessary [2].

Beyond the exchange of parameter values, to provide consistent simulation results in different software tools, power system dynamic simulation requires unambiguous mathematical models across different platforms. This is very difficult to attain given the different model realizations and limitations of existing proprietary data exchange formats. In addition, the fact that CIM does not provide means to exchange the models' equations explicitly limits the means to guarantee simulation consistency in different tools. This makes the use of a common modelling language very attractive to address the issues listed above. The iTESLA project has chosen the Modelica [3] language for this purpose because it offers a standardized modelling language suitable for equation-based modelling of complex cyber-physical systems. Other advantages include its maturity, the size of the community that supports it, its open nature, its equation-based approach that allows modelling of complex devices and the availability of variety of simulation tools, i.e. both open source and proprietary, which support the modeling language.

However, to adapt to different TSO's need, it is necessary to automatically generate Modelica models starting from proprietary dynamic data exchange formats and other information specific to each TSO. To address this need, several software modules have been developed in iTESLA to automatically transform power system models from different proprietary software tools; and a Modelica library containing mathematical models that have been implemented and validated against domain-specific simulations tools (PSS/E and Eurostag). This paper details the model transformation process, the different software modules developed to automate and integrate it in the iTESLA platform, and its validation.

The remainder of this paper is organized as follows. Section II gives component model example. In Section III, the automatic model transformation method is explained. In Section IV, two power system test models are presented and their transformation is validated through comparisons with PSS/E and Eurostag. Finally, in Section V, conclusions are drawn and future work is outlined. Links to the Open Source Software implementation of the proposed method are provided in the "Further Reading" section at the end of the paper.

II. POWER SYSTEM MODELING USING MODELICA

This work expands previous efforts [4] in building basic electrical component models into the iTESLA Power Systems Library (iPSL) [2] to provide consistent simulation results in different software tools that support the Modelica language. The library has been extended with new Three-Winding Synchronous Generator models used in Eurostag and also new models of other types of generators, governors, stabilizers and excitation system models, taking as reference their implementation in PSS/E, as well as models from other tools. The development of equivalent PSS/E models in Modelica required to understand the specifications from the PSS/E reference models and implementation details presented in [5]. This proved to be a challenging and a tedious work because there are no explicit mathematical equations in most of the model descriptions.

Different Modelica Integrated Development Environments (IDE) allow to build power systems models using the Modelica language [2]. Using Modelica, a power system model can be implemented using explicit mathematical equations or connecting models represented in block diagrams, which can be either user defined or from the Modelica Standard Library [6]. One sample model available in iPSL, which correspond to PSS/E, the excitation systems, IEEET1 shown in Fig. 1 and Modelica implementation is shown is Fig. 2. Observe in Fig. 2 the use of the `connect` statement, which binds each of the blocks shown in Fig. 1. While instantiation and connection of models occurs automatically by using the diagram view of a Modelica IDE, initial equations for each model have to be manually defined and are required during the initialization process. This include, as shown in Fig. 2, the initialization of differential equations, e.g. `Efd0 = EFD0`, as well as the computation of values to initialize algebraic functions, such as saturation, e.g. `SE_Efd0 = SE(EFD0, SE1, SE2, E1, E2).`

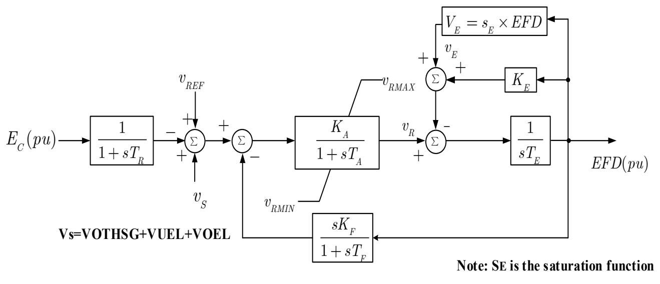


Fig. 1: Block diagram of the IEEET1 model [7].

The components available in the iPSL have been created following the same naming convention found in the reference software, i.e. PSS/E or other. Furthermore, the automatic conversion requires that the components available in the iPSL are modeled with the same naming convention, in order to generate the equivalent Modelica model.

```
model IEEET1
parameter Real Ec0 "power flow node voltage";
parameter Real TR = 1 "Voltage input time const.,
s";
parameter Real KA = 40 "AVR gain, p.u";
…
initial equation
VT0 = Ec0;
Efd0 = EFD0;
SE_Efd0 = SE(EFD0, SE1, SE2, E1, E2);
(VRMAX0, KE0) = ini0(VRMAX, KE, E2, SE2, Efd0,
SE_Efd0);
VR0 = Efd0 * (KE0 + SE_Efd0);
…
Equation
…
connect(imIntegrator.n1, EFD);
connect(se1.VE_IN, imIntegrator.n1);
connect(V_Erro.u1, Vref.n1);
connect(EC, V_Erro.u2);
connect(VOTHSG, Vs.p1);
connect(VOEL, Vs.p2);
connect(VUEL, Vs.p3);
…
end IEEET1;
```

Fig. 2: Modelica implementation of the IEEET1 (partial code excerpt).

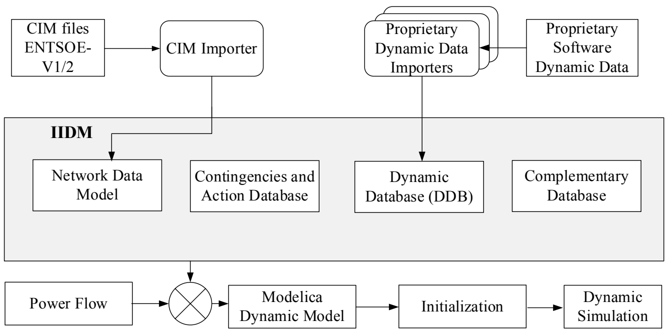


Fig. 3: Workflow for generating a Modelica model from CIM and proprietary data formats.

III. MODEL TRANSFORMATION AND INITIALIZATION

This section presents the automatic model transformation of networks described by dynamic data from different proprietary software (PSS/E and Eurostag) and CIM snapshots into equivalent Modelica models for two different test models. Three conversion tools developed for the whole transformation process are: CIM to iTESLA Internal Data Model (IIDM) converter, proprietary dynamic data importers and IIDM to Modelica converter. The general workflow used within the iTESLA toolbox is shown in Fig. 3.

A. CIM to IIDM converter

The CIM importer in Fig. 3 was implemented in Java and its function is to convert CIM files into the iTESLA internal data model. Network static data are made available in the IIDM using this converter. The converter supports the first version of the CIM ENTSOE profile. The second version will be implemented in the future.

B. *Proprietary dynamic data importers*

Dynamic data importers were developed in order to insert Eurostag and PSS/E dynamic data into the iTESLA platform. These importers automatically import a .dd file (.dta chunk for Eurostag), a .dyr file (for PSS/E) and a dictionary (CSV file) containing the mapping between Eurostag or PSS/E equipment identifiers and the IIDM equipment identifiers into the dynamic database (DDB).

C. *IIDM to Modelica Transformation*

The IIDM to Modelica transformation was realized in a software tool that takes the model description stored in the IIDM and outputs the model description in the Modelica language. To transform a power system network to Modelica, the tool retrieves the systems' data from different sources (see Fig. 5). The network data and dynamic data are taken from the network data model and DDB respectively within the IIDM. The network data describes the topology of the system to perform power flow computations.

Next the automatic model transformation method allows the conversion from IIDM to a Modelica model. All models generated with this tool (here on referred to as IIDM2Modelica) are defined by initializing the models from iPSL and connecting them using the `connect` keyword, as shown in Section IV (see Fig. 5 and Fig. 8). This implies that to transform a power system model to the Modelica language, it is necessary to have all Modelica component models from the iPSL as part of the internal dynamic database (DDB) of the iTESLA platform. So, prior performing the transformation, the iPSL should be available in the DDB. The main steps involved in IIDM2Modelica are the following:

- Obtain a power flow solution using the Holomorphic Embedding Load Flow (HELM) [8] to obtain initial values.
- Identify the network components stored in the IIDM and their parameters.
- Identify the dynamic components stored in the DDB and their parameters.
- Identify the connections between different components of the network to be converted.
- Write the model using the Modelica language; the system is ready for dynamic simulation.

When generating the Modelica model the graphical layout of the system will be provided. This can be supported in the future by using the `annotation` keyword of the Modelica language, however the original graphical layout definition should be known and mapped into all of the component's annotation fields.

D. *Initialization*

The general form for representing a power system can be written as:

$$\dot{\bar{x}} = f(\bar{x}, \bar{y}_d, \bar{y}_n, \bar{\eta}, \bar{u}, t) \qquad (1)$$
$$0 = g(\bar{x}, \bar{y}_d, \bar{y}_n, \bar{\eta}, \bar{u}, t) \qquad (2)$$
$$0 = h(\bar{y}_n, \bar{\eta}, \bar{u}) \qquad (3)$$

where, $\bar{x}, \bar{y}_d, \bar{y}_n, \bar{\eta}, \bar{u}$ and t are the vector of state variables, algebraic variables of dynamic components (AVR reference voltage etc.), algebraic variables of network (Voltage amplitude and phases of network buses), parameters, discrete variables and time respectively. The functions *f(),g()* and *h()* represents differential equations, algebraic equations and algebraic equations of network for power flow solution respectively. Power system domain specific tools find the solution for $\bar{y}_n$ by solving Eq. (3) then resulting values are then used to solve for $\bar{x}$ and $\bar{y}_d$ by setting Eq. (1) zero, for each individual component or a coupled subset of them. The initialization procedure for custom built Modelica models is described in [4] and for models generated using IIDM2Modelica discussed next.

1) *Initializing Models in iPSL*

For Modelica power system models that contain components from any other reference tool other than Eurostag (i.e. PSS/E, PSAT, etc.), the initialization of dynamic models is the same as the one used by any other Modelica library [9]. Thus, the procedure finds a solution for the resulting set of non-linear algebraic equations (1), (2), (3) when setting (1) equal to zero. This requires that each of the components used in the network model define which variables have to be initialized, within the "`initial equation`" section of the Modelica model, as shown in Fig. 2 for the IEEET1 Excitation control system. Although this approach is supported by the Modelica language standard, and Modelica tools have efficient methods to solve the initial value problem [10] [11], this is still a difficult non-linear problem for solvers and optimizers [12]. Therefore, the common practice adopted to initialize iPSL models is to define as `parameters` the values corresponding to a power flow solution and other static equipment variables, like exciter saturation parameters, maximum and minimum values of the limiters for helping the solver in use to calculate the initial conditions of a simulation.

2) *Initializing Models from Eurostag in iPSL*

Eurostag equivalent models in iPSL are comprised by two parts: (a) model containing all model equations, and (b) initialization models that contain the initial equations necessary to find the initial state of each corresponding dynamic model. Because of this, Eurostag equivalent models in iPSL do not contain "`initial equation`" statements as all other models do. Instead, these models make use of variable attributes in order to control the initialization of each of the component's variables. This is carried out by using the `start` attribute in conjunction with the `fixed` attribute

All of these `start` values are computed through the execution of the auxiliary file that defines the specific initialization equations and procedures. Thus, after obtaining a power flow solution for the network model, the automatic conversion tool stores the solution for algebraic, continuous and discrete variables into the DDB, i.e. the solution of Eq. (3). Then to solve for Eq. (1) and (2), each dynamic component is initialized separately, i.e. by solving a subset of Eq. (1) and (2) for example, f_i and g_i for all *i=1...n* (*n* is the number of dynamic components). In order to carry out this step the component represented by f_i and g_i for all *i=1...n* must have a corresponding initialization model. For example in Fig. 4, M1S_1_init.mo and Reg1_init.mo to RegN_init.mo are the initialization network model represented by different subsets of Eq. (1) and Eq. (2). After that, the initialization network model is simulated to calculate the steady state values, and the obtained values are inserted in the corresponding components. This procedure has been achieved by using the OpenModelica API for JAVA [13]. The initial values are provided to the full network model, which is now ready to perform dynamic simulations.

The reasons to adopt this approach were (a) to maintain consistency with the original Eurostag source code, which contains an initialization model separate from a dynamic model, and (b) due to a lack of knowledge during the initial stages of the iTESLA project of the technical reasons why Modelica offers the `initial equation` construct and how to properly utilize it [14]. Note that the risk of using attributes [15], even when utilizing the `start` in conjunction with the `fixed` attribute, is that Modelica tools may choose to add equations to solve as constraints during the initialization process, and thus, the solution that the Modelica tool will yield and provide initial values are different than those specified in the `start` attribute.

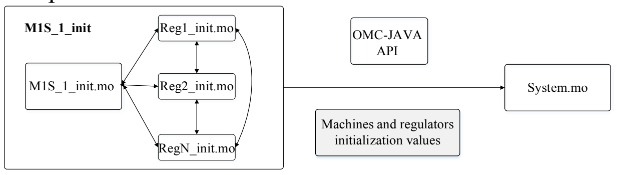


Fig. 4: Model initialization procedure.

E. *Dynamic data conversion*

TSOs work with proprietary software, which have their own proprietary data format. In order to adapt the TSOs models into the iTESLA project, a revision and update of the existing models in the iPSL was necessary to guarantee that the automatic conversion performed as required. For the Nordic 32 test system (shown in Section IV), this required to add parameters into existing controls and other minor modifications. Moreover, the library has been extended with new components from PSS/E reference models. These models [5] have been used for the implementation of the Nordic 44 test system. In the process of developing new Modelica models, the IIDM has to provide the correct names and parameters for the components of any proprietary data format and units, so the conversion can handle easily the translation into the internal database and to instantiate the appropriate models in Modelica.

IV. Validation

Two models, the Nordic 32 test system with reference Eurostag, and the Nordic 44 test system with reference PSS/E, are transformed into Modelica using the approach in Section III. The resulting Modelica models are simulated using Dymola [16] to obtain their dynamic response when subjected to different perturbations. The simulation outputs from the Modelica tool are validated against each reference software package.

A. *Quantitative Assessment*

The validation against each reference software package was carried out both graphically and numerically. The numerical assessment is carried out using the Root Mean Square Error (RMSE) [6] and Mean Square error (MSE). The RMS and MS value of the errors are calculated using the following equations:

$$Z_{RMSE} = \sqrt{\frac{1}{n}[(x_1 - y_1)^2 + \cdots + (x_n - y_n)^2]} \quad (4)$$

$$Z_{MSE} = \frac{1}{n}[(x_1 - y_1)^2 + \cdots + (x_n - y_n)^2] \quad (5)$$

where, $x_1, \ldots, x_n$ are the discrete measurement point at time $t_1, \ldots, t_n$ for software package (a) and $y_1, \ldots, y_n$ are the discrete measurement points at time $t_1, \ldots, t_n$ for software package (b). Z_{RMSE} and Z_{MSE} is the RMSE and MSE value of the of Z variable. The chosen assessment metric is that Z_{RMSE} and Z_{MSE} will yield values $\leq 1\times 10^{-3}$ and 1×10^{-6} respectively, in order to accept the result as valid.

B. *Case1: Nordic 32 test system*

The KTH Nordic 32 test system is a conceptualization of the Nordic Grid. It is composed by 52 buses, 20 synchronous machines, 28 fixed-ratio transformers, 22 voltage-dependent loads, 11 reactor banks and different excitation systems. The model description used herein originated from [17]. It was implemented in Eurostag and used as reference. The automatic generated Modelica code obtained using the proposed transformation approach is shown in Fig. 7.

```
model Nordic32
parameter Real SNREF = 100.0;
PowerSystems.Connectors.ImPin omegaRef;
// BUSES
// LINES
// FIXED TRANSFORMERS
// LOADS
// CAPACITORS
// GENERATORS
// REGULATORS
// EVENT
PowerSystems.Electrical.Events.PwFault pwFault
(R = 0.1, X = 0.1, t1 = 20, t2 = 150);
equation
omegaRef  "sum of omega from all generators";
connect(pwGeneratorM2S.omegaRef, omegaRef);
…
// Connecting REGULATORS and MACHINES
connect(htgpsat3.pin_CM,pwGeneratorM2S.pin_CM);
…
// Connecting LINES
connect(bus.p, pwLine.p);
…
// COUPLING DEVICES
// Connecting LOADS
connect(bus.p, pwLoadPQ.p);
…
// Connecting Capacitors
connect(bus.p, pwCapacitorBank.p));
// Connecting GENERATORS
connect(bus.p, pwGeneratorM2S.sortie);
…
// Connecting FIXED TRANSFORMERS
connect(bus.p, pwTransformer.p);
…
//Connecting FAULT
connect(bus.p, pwFault.p);
…
end Nordic32;
```

Fig. 5: Modelica model of the KTH Nordic 32 system generated using the proposed transformation method (partial code excerpt).

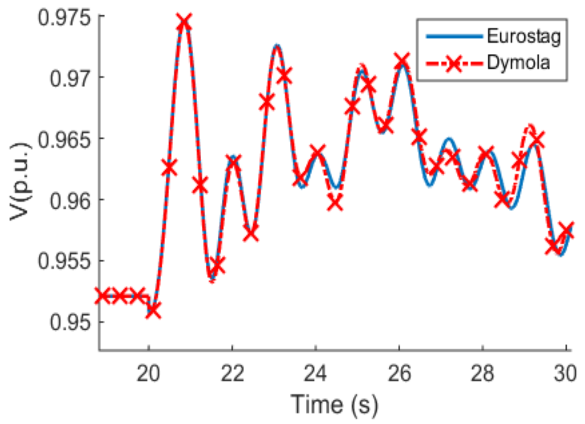


Fig. 6: Dynamic response from both Eurostag and Dymola simulators.

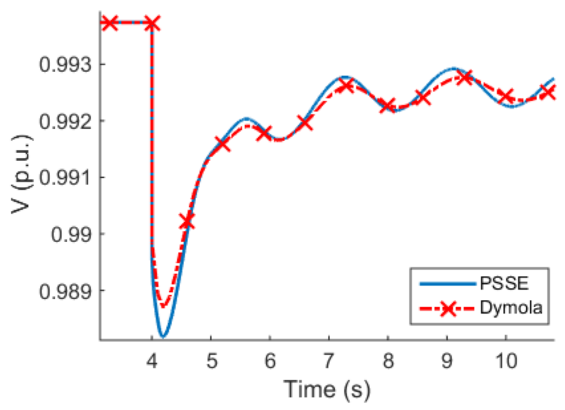


Fig. 7: Dynamic response from PSS/E and Dymola simulators.

Next, the simulation outputs of the Modelica model are validated against the Eurostag reference model. In this case, both the Eurostag model and the Modelica model are simulated to obtain their response due to a fault on bus N1014, at time 20s (see Fig. 6). The comparison is given in Table 1. Minor discrepancies seen in Fig. 6, and the acceptable RMSE and MSE values that are beyond the given numerical solver tolerance assure that the proposed transformation method is valid for models whose original reference is Eurostag.

C. Case 2: Nordic 44-Bus test system

The second model used for validation is the Nordic 44-Bus test system. This is an equivalent of the Nordic Grid developed within the iTESLA project. This system is comprised of 44 buses and 61 generators with different models for excitation systems, turbine governors and stabilizers. The system is converted from its original reference in PSS/E. The simulation result with a bus fault is shown in Fig. 7 and the numerical results for assessment are given in Table 1. Contrasting the traces in the Fig. 7 and the RMSE and MSE values in Table 1, indicates that the transformation process is valid for models whose original reference is PSS/E.

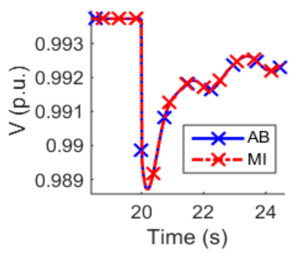
D. Case 3: Nordic 44 custom-built vs automatically-built

The aim of this validation case is to demonstrate that the converter tool can generate accurate models from external sources, in the same way an engineer can model a network by checking directly the reference model and do manual tests. For this purpose, the Nordic 44 model was manually implemented (MI). This means that the connections and model initialization has been done by hand, checking values and connections directly from the PSS/E model reference (i.e. through direct human intervention). The power flow values have been taken from the power flow solution provided by PSS/E.

In this case, the validation of the automatic-built (AB) model and the hand-built model is performed by obtaining the response for (a) a permanent fault and (b) a fault at time 20 s with duration of 0.2 s. The simulation results are shown in Fig. 9 and the quantitative assessment is given in Table 1. Both figures and quantitative values in the table indicate indiscernible numerical differences. Hence, it can be concluded that the transformation process for models whose reference is PSS/E is consistent with traditional/artisanal engineering practice.

```
model Nordic44
parameter Real SNREF = 100.0;
// BUSES
// TAP CHANGER TRANSFORMERS
// LINES
// LOADS
// CAPACITORS
// GENERATORS
// REGULATORS
// EVENT:FAULT
PowerSystems.Electrical.Events.PwFault_fault(X = 0.5, R = 0.5, t1 =
20, t2 = 100);
equation
// Connecting REGULATORS and MACHINES
connect(stab2a.PELEC, gENROU.PELEC);
…
// Connecting REGULATORS and REGULATORS
connect(stab2a.VOTHSG, ieeet2.VOTHSG);
…
// Connecting REGULATORS and CONSTANTS
connect(ieeet2.VOEL, const.y);
…
// Connecting LINES
connect(_bus.p, pwLine_2.p);
…
// COUPLING DEVICES
// Connecting LOADS
connect(bus.p, pwLoadVoltageDependence.p);
…
// Connecting Capacitors
Connect(bus.p, pwCapacitorBank.p);
…
// Connecting GENERATORS
connect(bus.p, gENROU.p);
…
// Connecting DETAILED TRANSFORMERS
connect(bus.p, pwPhaseTransformer.p);
//Connecting FAULT
connect(bus.p, _fault.p);
end Nordic44;
```

Fig. 8: Automatically generated Modelica model of the Nordic 44-Bus system partial code excerpt.



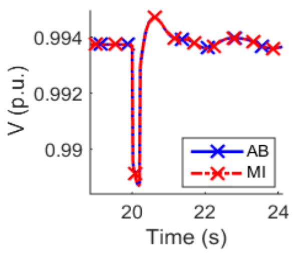


Fig. 9: Dynamic response of Nordica 44-Bus system. [in red: manually implemented model, in blue: automatically generated model], with a permanent fault at 20s (left) and with a fault from 20 s to 20.2 s (right).

Table 1: Quantitative Assessment for Validation.

Test System	Variable	RMSE	MSE
Nordic 32	*V2032*	9.2378e-04	8.53382e-07
Nordic 44	*V3020*	9.0215e-05	8.13877e-09
Nordic 44 (Test 1)	*V3020*	3.0426e-06	9.25789e-12
Nordic 44 (Test 2)	*V3020*	5.2357e-06	2.74131e-11

V. CONCLUSIONS AND FUTURE WORK

During the course of this work, several lessons about modelling and simulation with Modelica have been learnt. Previous work [4] showed preliminary results of modelling different Modelica models and test networks. The present paper expands this by proposing a method to automatically transform them from their native dynamic description (Eurostag and PSS/E) and CIM steady state snapshot. The main differences found in the larger networks considered here with respect to previous work are: a) different initialization schemes; b) new types of generator models and regulation schemes voltage and frequency controls and c) these systems were generated automatically. The use of the iTESLA Power System Library (iPSL), makes the automatic conversion from a proprietary data format into the iTESLA Internal Data Model possible (with some effort) for TSOs who want to utilize the iTESLA platform. The main effort would be to populate iPSL with models used by a TSO which are not available there, to develop the corresponding proprietary dynamic data importer if the TSO tool is different from PSS/E or Eurostag.

Another option, not explored in this work, would be to develop a self-contained transformation tool that could use as input a given ‘data format’ used by a TSO (e.g. CIM) and output a Modelica-compliant model. The effort to develop such tools is difficult to estimate, as there are many software design and implementation issues that need to be considered. One of them is that it would require to populate the iPSL with any models that are not available for the conversion process.

FURTHER READING: OPEN SOURCE SOFTWARE DISTRIBUTION

The software implementation of the proposed method is available as Open Source Software in the following Github repository: https://github.com/itesla/ipst , where the specific portion for the automated model transformation process can be found at https://github.com/itesla/ipst/tree/master/modelica-export

The iPSL Modelica library can be obtained in the following repository: https://github.com/itesla/ipsl, while improvements on the library made are being made by the research team of the first author are in a fork of the library called OpenIPSL that can be found in the following repository: https://github.com/SmarTS-Lab/OpenIPSL

ACKNOWLEDGMENTS

This work was supported in part by the EU-funded FP7 iTESLA project under grant agreement n283012 The authors are in debt to many iPSL and OpenIPSL model contributors of KTH SmarTS Lab that have participated in iTESLA.

REFERENCES

[1] iTesla: Innovative Tools for Electrical System Security within Large Areas. [Online] http://www.itesla-project.eu/
[2] T. Bogodorova, M. Sabate, G. Leon, L. Vanfretti, M. Halat, J.B. Heyberger, P. Panciatici, "A modelica power system library for phasor time-domain simulation," in *Innovative Smart Grid Technologies Europe (ISGT EUROPE), 2013 4th IEEE/PES* , vol., no., pp.1-5, 6-9 Oct. 2013.
[3] Modelica® and the Modelica Association. http://www.modelica.org/
[4] G. Leon, M. Halat, M. Sabate, J.B. Heyberger, F.J. Gomez, L. Vanfretti, "Aspects of power system modeling, initialization and simulation using the Modelica language," in *PowerTech, 2015 IEEE Eindhoven*, pp.1-6, June 29 2015-July 2 2015.
[5] M. Zhang, M. Baudette, J. Lavenius, S. L_vlund, L. Vanfretti, "Modelica Implementation and Software-to-Software Validation of Power System Component Models Commonly used by Nordic TSOs for Dynamic Simulations", *in: 56th Conf. Simul. Model*. (SIMS 56).
[6] M.A.A. Murad, F.J. Gomez, L. Vanfretti, "Equation-based modeling of FACTS using Modelica," in *PowerTech, 2015 IEEE Eindhoven*, pp.1-6, June 29 2015-July 2 2015.
[7] PSS®E 33.5, "Model Library", Technical report, Siemens PTI Ltd, October 2013.
[8] A. Trias, "The Holomorphic Embedding Load Flow method," in *Power and Energy Society General Meeting, 2012 IEEE* , pp.1-8, 22-26 July 2012.
[9] P. Fritzon, "Principles of Object-Oriented Modeling and Simulation with Modelcia 3.3: A Cyber-Physical Approach, Wiley-IEEE Press", 2nd Edition, April 2015.
[10] C. Pantelides, "The consistent initialization of differential-algebraic systems." SIAM J. Sci. Stat. Comput. 9:2, 213–231, 1988.
[11] S.E Mattsson, G. Söderlind, "Index reduction in differential–algebraic equations using dummy derivatives." SIAM J. Sci. Comput. 14(3), 677–692 (1993).
[12] B. Bachman, W. Braun, L. Ochel, V. Ruge, "Symbolical and Numerical Approaches for Solving Nonlinear Systems, Annual OpenModelica Workshop", February 2015.
[13] M. Sjölund, P. Fritzson. "An open modelica java external function interface supporting metaprogramming." *7th International Modelica Conference; Como; Italy; 20-22 September 2009*. Linköping University Electronic Press, 2009.
[14] M. Tiller, "Modelica by Example" [On-line] http://book.xogeny.com/behavior/equations/initialization/
[15] M. Tiller, "Introduction to Physical Modeling with Modelica", The Springer International Series in Engineering and Computer Science, 2001,
[16] Dymola: Tool for modeling and simulation of integrated and complex systems. A commercial product from Dassault Systemes. http://www.3ds.com
[17] Y. Chompoobutrgool, W. Li, L. Vanfretti, "Development and implementation of a Nordic grid model for Power System small-signal and transient stability studies in a free and open source software," Power and Energy Society General Meeting, 2012 IEEE , pp.1-8, 22-26 July 2012.