Interactive Model Transformations from the Common Information Model (CIM) to Modelica

Glen K. Halley
City Utilities
Springfield, MO 65802
Glen.Halley@CityUtilities.net

Luigi Vanfretti
Rensselaer Polytechnic Institute,
Troy, NY 12180
vanfrl@rpi.edu

Marcelo de Castro
Mitsubishi Electric Power Products
Warrendale, PA 15086
marcelo.decastro@meppi.com

***Abstract*—Model transformation is necessary to save time in using different power system analysis tools. The transformation process should ensure that model parameters are correct when using a tool that is different than the original source, and systematically moving and mathematically changing the source parameters. This paper describes an interactive model transformation process using eXtensible Stylesheet Language Transformations (XSLT) within the EditiX software. The process starts from models in the Common Information Model (CIM). The XML files were generated by the PSS®E ODMS software according to the Common Grid Model Exchange Specification (CGMES) Library. The transformation results in a text file, linking the OpenIPSL library, with a MO extension for use with Modelica-standard-compliant dynamic simulation tools.**

Index Terms *— Common Information Model (CIM), XSLT, information exchange, information modeling, Modelica, OpenIPSL, power system dynamics.*

I. Introduction

The Common Grid Model Exchange Specification (CGMES) [1] contains descriptions of Extensible Markup Language (XML) files used to facilitate the exchange of operational and grid planning data among transmission system operators. Extensible Stylesheet Language Transformations (XSLT) [2] within EditiX's interactive development environment (IDE) provides means to interactively translate CIM/CGMES XML files to a Modelica file linked with the OpenIPSL library [3]. Observe that XSLT is meant specifically for translating XML documents. This model transformation allows to perform dynamic simulations using Modelica-standard-compliant software such as OpenModelica, Wolfram SystemModeler or Dymola. This approach makes the process of transforming the models from CIM to Modelica understandable, flexible, and maintainable.

CGMES extends a Common Information Model (CIM) which provides components and system information in a structure that matches the intent of the European Network of Transmission System Operators for Electricity (ENTSO-E) protocols [4]. The protocols describe the tagging file structure, hierarchy, and relationships that are necessary to understand the data. In this case CGMES/XML contains modeling parameters for power system components. Because the XMLdoes not contain possible (or default) parameters used as part of the meta-data, it is necessary to understand parameter definitions from the standards.

A. Motivation

Power systems are rapidly evolving; however, the availability of efficient collaborative engineering design and analysis tools lags the technological deployment in the field. Most of the existing tools used in the United States for power system analysis can't share information efficiently and transparently, and more importantly, without ambiguity [5]. In many cases this means that mostly only one tool is used for typical analysis needs, i.e., to perform power flow and dynamic simulations. When a new analysis need appears and it is not supported in the tool, e.g., thermal analysis, it is most likely not performed due to the substantial time, effort and experience required to use the model in another environment. In contrast, the data model provided by CIM [6] and the additional definitions provided by CGMES[1] allow for the exchange of data that is created once and used by many tools. This capability has made it the preferred translation method used in European utilities for data exchange and is now supported by software such as PSS®E ODMS and DigSILENT PowerFactory.

OpenIPSL [3], written using the open-access and standardized Modelica language, is a power system dynamic model library with many of the components available in PSS®E (e.g., GENROU, for generators, HYGOV for hydro turbine-governors, and so on). When used in conjunction with a Modelica-standard-compliant software tool, it is possible to perform dynamic studies leveraging both time-domain simulation and linearization. It becomes attractive to translate models from CIM/CGMES to Modelica in such a way that the OpenIPSL library models are re-used. In this context, CGMES/CIM files include all the necessary information to transfer the power flow and dynamic model information from the PSS®E ODMS to OpenIPSL. To automate this process, the transformation normally requires parsing the original files, storing the parsed data, and reassembling the data into a new Modelica file, i.e., a ".mo" file.

This paper proposes to use the EditiX software and XSLT as a means to perform such translation. To this end, the CIM/CGMES XML files can be translated using templates for each component, thereby operating only on the required data without parsing and storage. Observe that this is an important aspect, as it implies that it is not necessary to create an intermediate data model and associated database, as in previous efforts [7]. The process begins with the required parameters, in this case those used by OpenIPSL models, and looks for the equivalent parameters within the XML files. Then, unit conversions, per unit modifications, or other mathematical functions are used to make sure the parameters match requirements. Template translation using EditiX and XSLT consists of sequential steps, each of which relates to the final output. With prerequisite understanding of Modelica and OpenIPSL, the EditiX/XSLT toolset simplifies the workflow and creates maintainable translation rules. It does not require complete understanding of all the CIM/CGMES XML files since we will introduce analysis to determine the proper translation rule to apply.

Even though attractive, this approach would still face some challenges that are independent of the software technologies used. First, observe that in the utility context in the United States there is no motivation for vendors to comply with CIM standards, and even less to adopt CGMES, and there isn't a long list of successful projects of interoperability. Most importantly, in the United States, there isn't any sort of government mandate for interoperability. In contrast, in Europe, it is a mandate for vendors to provide and maintain CIM exports. Finally, regardless of the regulatory context, a vendor's software may not have complete parameters for a given function in another piece of software.

Despite the lack of regulation or vendor roadblocks for model transformation to be successful, the payoff for using other tools (e.g., Modelica tools with OpenIPSL) offers unique benefits because it provides:

- A different perspective that builds confidence in the first tool's problem statement and exposes other branches for investigation. Based on Modelica, OpenIPSL is easy to understand and use, as it follows textbook-like equations [3].
- Analysis extension by incorporating a different domain [8], for example, generating constraints from limitations on fuel such as natural gas during winter events.
- Originating tool testing for comparable results for the same parameters [3].

B. *Previous Work*

The ENTSO-E provided business standards through the CGMES Libraries. CGMES is part of the overall International Electrotechnical Commission (IEC) 61970 CIM standards which define formats for, and definitions of power system parameters used for network planning and power system operations. These standards provide modeling information for individual components and their associated parameters, but don't describe overall verification after translation and use by target tools. In this work, successful translation is verified by comparing results between the origination tool and the resulting Modelica model simulated under the same conditions using a Modelica tool. This process would be outside the XML file structure used for dynamic analysis.

The best parameter descriptions are listed by component and table at the following web site: https://cgmes.github.io/.

II. XML, XSLT, FUNCTIONS, AND EDITIX TOOLS

The XML was exported from PSS®E ODMS[9]. Translating from XML to Modelica is a process of determining what parameters the components in the Modelica library need, where the parameters reside in the XML files, and writing the translation templates. XSLT naturally accommodates standards-based XML. For example, there is no expectation of order for parameters, but XSLT can easily sort by multiple parameters if necessary.

Processes that are not encapsulated in XSLT are placed in helper functions that are executed within EditiX. This set of functions is limited to a few common needs characterized by accommodating multiple XML files, ancillary tables, and to format the Modelica output file.

A. *XML Analysis*

The XML is structured into a tree of nodes. While an XML tree can be many levels deep, the CGMES limits it to a root, components or ancillary table, and parameters. This limitation allows one to treat the tree as a relational database where a database table is the components (or ancillary) table, and the parameters are the fields within the table.

The table is formed by looking at all the nodes with the same tag, for example the *ACLineSegment* tag contains line parameters. XSLT provides XPath (XML Path Language), a syntax to navigate nodes and match patterns to do this. In the *ieee14_EQ.xml* file[9], the nodes are grouped together in alphabetical order. This is not expected, but it is helpful to understand the way it behaves as a relational database table. It typically requires two namespaces entered in the *Namespace* tab, as shown in Fig. 1.

Namespace prefixes are listed in the root node and used in the entire XML document. XSLT is an XML document and the root node is name *xsl:stylesheet*. '*xsl*' is the namespace prefix. Then, using an XPath expression to select the nodes with the same tag '*/rdf:RDF/cim:ACLineSegment*' and executing this expression would result in a list of all nodes with the pattern of root node *rdf:RDF* and component/table node for power lines. XPath is used extensively within the XSLT select statements.

Like relational databases, CGMES XML also has pre-defined relationships between the tables using keys. This allows data normalization. Each relationship has a key that connects the component and table. Tables contain parameters that are used by more than one component such as bus names.

Namespace is three characters followed by a colon within a tag. It is used to discern who is responsible for the tag. For example, '*xsl:*' are XSLT commands, '*cim:*' is part of the CGMES data, and for custom functions '*gkh:*' was used. A list of *namespaces* is included in each file in the *xsl:stylesheet* tag with the attribute *xmlns* followed by the three-character code. It is worth it to note that the *pti-namespace* is also present, which is for PSS®E-specific information and is not used.

XML has a tree graph structure. For the purposes of this paper, CGMES and RDF truncates the tree to three levels: root, component/table, and parameters. The domain is limited based on the active template. Templates act on the component or table level, therefore, when working on one component/table functions can't "see" the other components. This limitation is overcome by using functions which operate at the root level. Not only can they "see" other components, but they can "see" other components in other files.

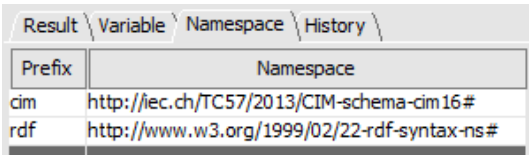

Fig. 1. Required namespaces.

CGMES uses normalization that is traditionally a relational database mechanism. One component can reference another component/table using a foreign key that links to a primary key, respectively. Both use a 128-bit GUID with an underscore (_) in front. The foreign key adds an additional pound (#) -sign to the front. The GUIDs are meant to be unique. Therefore, without much knowledge one can search with a GUID from the main component (less the #) and find related information, albeit the search may be required for all the files individually.

B. XSLT Structure

The templates or translation rules have a main umbrella file, herein refered to as *CIMtoMO4.xsl*, that contains the overall list of component files (imports), list of utility functions, and all the XML document files (documents). The first and only template in the main file points at the root using the '*<xsl:template match="/rdf:RDF">*

</xsl:template>' node. Nodes have opening and closing tags. The whole EQ XML file is its domain and applies to everything between the tags. The first sub-node is a *<xsl:text>package </xsl:text>*. This places the text "*package*" in the results file.

"*Package*" corresponds to the first word of the first line in the Modelica file. Templates combined with text are used to build the translation. The next line is '*<xsl:apply-template select="md:FullModel"/>*'. In this case the opening and closing tags are combined into a single statement. Apply-template reduces the domain from everything under the root to just the *FullModel* node (it has a '*md*' namespace) and it looks for the corresponding template node '*<xsl:template match= "md:FullModel">*' to execute the template.

When the main file is opened in EditiX, all imported files are shown as tabs at the top of the document window as shown in Fig. 2. This helps to quickly find each different component. If imports are added to a sub-file, they will also show up in a tab. If a new component is needed, then a new file is created and a template and import statement are added to the main file. The component file requires the XSLT namespaces, output method (text), and template.

C. Common Workflow

The general workflow for the simplest translation uses variables, XPath, and helper functions. We'll start by looking at a translation for the power line in the file '*lines.xsl*'. After the stylesheet tag, which includes the namespaces, the template begins with a variable called *baseImpedance*, as shown in Listing 1.

This one statement uses *name=* to name the variable, *select=* to assign the function *baseImpedance*, uses XPath to navigate to the Conducting Equipment Base Voltage parameter, and uses the built-in function substring to parse the parameter.

```
<xsl:variable name="baseImpedance" select= "gkh:
baseImpedance ( cim:ConductingEquipment.BaseVoltage/
substring(@rdf:resource,2),0,0)"/>
```

Listing 2. *baseImpedence* variable

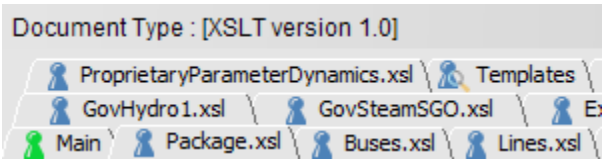

Fig. 2. Translation rules in separate files shown as tabs.

A more complex example is that of transforming a transmission line model. Consider the next XSLT rule shown in Listing 2.

In the XSLT rule, the *text* and *value-of* commands are after the variable. *Value-of* copies the information between the tags for the parameters. Only formatted output functions are necessary for simple translation. All the parameters are in the core node. The transmission line for OpenIPSL needs R (resistance), X (reactance), G (shunt conductance), and B (shunt susceptance) parameters for the *PwLine* Modelica component. The available XML parameters will come from the XML node *ACLineSegment* (line) including parameters *r*, *x*, *gch*, and *bch*. The rule requires converting all values to per unit using the base impedance. It also requires dividing both the G and B values in half, see the *ieee14_EQ.xml* file [9].

Functions are used within the select statements. In addition to the built-in *substring* function, the built-in functions *format- number* and *concat* are also needed. The custom functions *compliantName* and *defaultNumbers* were also required to perform the translation. Finally, per unit calculations were created in the *select* statements with the *div* and * operators in conjunction with the *$baseImpedance* variable.

D. Helper Functions

There are many built-in functions of XSLT, none of which include namespace identification. For example, *format- number(...)* will format a number. The helper functions created for this work use the '*gkh*'-namespace. For

```
<xsl:template match="cim:ACLineSegment">

    <xsl:variable name="baseImpedance" select=
"gkh:baseImpedance(cim:ConductingEquipment.BaseVolt
age/substring(@rdf:resource,2),0,0)"/>

    <xsl:text>OpenIPSL.Electrical.Branches.PwLine</xsl:te
xt><xsl:value-of select=
"gkh:compliantName(concat('line',cim:IdentifiedObject.n
ame,substring(@rdf:ID,6,4)))"/>
<xsl:text>(R =</xsl:text><xsl:value-of select="format-
number( cim:ACLineSegment.r div
$baseImpedance,'0.000000000#')"/><!          >
<xsl:text>, X =</xsl:text><xsl:value-of select="format-
number(cim:ACLineSegment.x div
$baseImpedance,'0.0000000000#')"/>
<xsl:text>,G = </xsl:text><xsl:value-of
select="gkh:defaultNumbers(cim:ACLineSegment.gch
*$baseImpedance div 2,0.00)"/>
<xsl:text>, B =</xsl:text><xsl:value-of select="format-
number(cim:ACLineSegment.bch * $baseImpedance div
2,'0.0000000000#')"/>
<xsl:text>);</xsl:text>
…
</xsl:template>
```

Listing 1. Transmission line transformation rule, *ACLineSegment*

example, *gkh:defaultNumbers* is defined in the *CIMtoMO4.xsl* file and is used to select a default number on the right if there is an invalid number on the left as *gkh:defaultNumbers(NaN,0.00)* will output 0.00. In addition to the two functions already mentioned, the line components use *gkh:compliantName(...)* to create a unique name necessary for OpenIPSL components. It combines the word '*line*', *cim:IdentifiedObject.name* and four digits of the line identifier GUID using a built-in function substring (i.e., 'line' + '1_2_1' + 25f8).

E. Example of Running an XSLT-based Transformation

Using EditiX, we use the File->Open Project and select the downloaded *XSLTConferencePaper* directory in [10]. The XSLT can run against the XML using the following procedure:

1. Opening the *scenario.xfl* scenario file assigns:
   a. *CIMtoMO4.xsl* XSLT as the core translation file.
   b. Input file relative path/name (EQ-file) which also has three other files in the same directory (DY, SV, and TP).
   c. Input parameters to the XSLT including frequency, results path, and the faulted bus's name, resistance, and reactance, start time, and end time.
2. Use the Visual Editor tab in EditiX at the bottom for help in setting up the scenario paths show in Fig. 3.
3. Two scenarios have been created in the file. Run the IEEE14 scenario using the red underlined button in Fig. 5.

III. ASSOCIATIONS

Parameters could be within components or common tables. If they are within the tables, there will be an association. Associations can be determined by looking at any GUID code used to link components' and tables' global information. Use https://cgmes.github.io/#core-equipment-ac-line-segment to see the possible component/table associations.

CGMES XML classifies the XML nodes of a power system as shown in Fig. 4. Terminals are linked to connectivity nodes (buses) and conducting equipment such as lines, transformers, shunts, machines, and loads. The OpenIPSL connections are created by going through all terminals

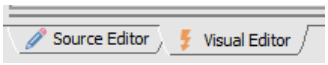

Fig. 5 Visual editor tab.

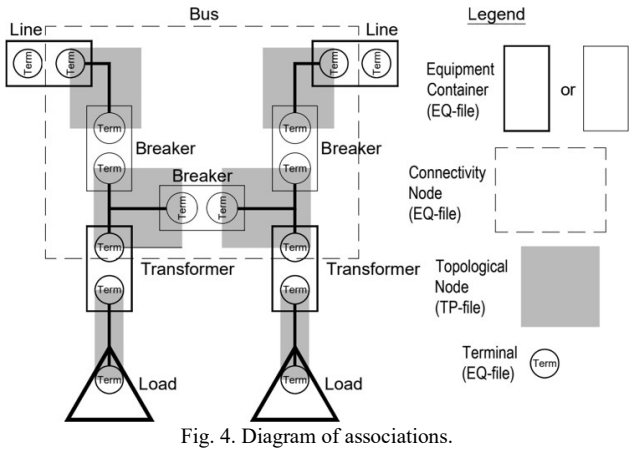

Fig. 4. Diagram of associations.

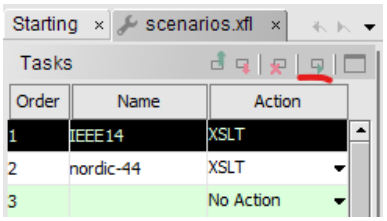

Fig. 3 Generated scenarios and execution of translation.

except for *BusNameMarkers*. The connection is the bus and equipment for each terminal.

You can see two GUIDs within each line component. The top GUID is a unique identifier of the line with an attribute of *rdf:ID*. It is the same as the connectivity node in *cim:Terminal*. Any class with a dependency on the line will use the GUID with a pound-sign (#) in front of it. In order to determine the per unit values, we will use the

<cim:ConductingEquipment.BaseVoltage>

attribute rdf:resource GUID to find the value.

The desired Modelica output for a load starts with *PSSE.Load*. Upon examination of the examples in OpenIPSL: characteristic, initial active power, initial reactive power, initial voltage magnitude, and initial voltage angle are required. The characteristic is determined based on the model selection. All other parameters are values copied out of the XML into the OpenIPSL Modelica models.

Unlike transmission line translation, the required parameters for loads are not all within the core component (EQ-file). The component uses relationships between attributes to add power/reactive power and voltage/angle parameters. Such parameters are stored in the SV-file (*cim:SvPowerFlow*) which uses a *Terminal* (*rdf:resource*) attribute for relationships.

It is necessary to determine associations between the *Terminal* attribute and the load's (*cim:ConformLoad* and *cim:NonConformLoad*) available attributes: equipment code, Load Group, Load Response, Equipment Container. Each component has an equipment attribute, which is a GUID. Equipment attribute of load is used as the function parameter of *gkh:SvLoadTree* which is used to get the Modelica parameters.

SvLoadTree calls another function *gkh:EQterminal* to convert the equipment attribute to a Terminal attribute. The function return is the entire tree for the particular *cim:SvPowerFlow* which includes both power and reactive power. Voltage/angle are added with the *gkh:SvVolt* function.

In this case PSS®E's load model is a ZIP (total load is assumed to be composed of parts defined as constant impedance, current, or power) model. We must use the characteristic setting that will guarantee power and allow reactive power to drop as a function of voltage squared. This means the OpenIPSL characteristic attribute must be set as 2. Load Response relationship is used to create the settings for this model type.

The model *OpenIPSL.Electrical.Buses.Bus* requires the *gkh:baseVoltage* function in addition to *gkh:SvVolt* in order to get the Modelica parameters into per unit. Otherwise, it is very similar to the voltage/angle code for loads. Meanwhile, the model

OpenIPSL.Electrical.Banks.PSSE.Shunt requires the *gkh:baseImpedance* function to create per unit impedance parameters for Modelica.

IV. Creating Functions

Templates and functions are the key mechanisms used for translating CGMES XML. Templates match an XML node. In this case the node is a component or table. Under the node are the parameters (that are also nodes), and any keys used for linking to other components/tables. Functions are used to access components/tables using keys.

XPath is part of XSLT and typically shown in a select attribute. It uses path expressions to navigate XML documents. It can select parts of the tree. XSLT variables were created at the top of the main translation file linking to the different CIM files. They are *$DY* (dynamic), *$TP* (topology),*$SV* (state variables), and $rdf (equipment) files. The files will be followed by either the root node (*rdf:RDF*) or the component/table. For example, *$rdf/cim:Terminal* starts with all the Terminal nodes in the equipment file.

The function *gkh:tapchanger* returns a complete node with its parameters, which is a return type of as="element()". It receives one string parameter *as="xs:string"* and names it *name="transEndCode"*. The command copy-of is the element return value. The select statement uses the document variable $rdf for the main document and the Xpath

"cim:RatioTapChanger/cim:RatioTapChanger.TransformerEnd"

with the bracketed filter. A substring function cuts off the pound sign from the foreign key and matches it with the parameter variable. Finally, the XPath ".." moves the context back up to the *RatioTapChanger* node. See the example in Listing 3.

V. Conclusion

It takes hundreds of equipment parameters to complete a power flow or dynamic analysis. The size of the analysis is based on the pool of Real Time Operators (RTOs) in the United States. For example, the Southwest Power Pool (SPP) includes large parts of North Dakota, South Dakota, Nebraska, Iowa, Kansas, Missouri, Arkansas, Oklahoma, and Texas. While PSS®E is the standard modeling software for static and dynamic analysis for some RTOs, OpenIPSL running within a Modelica-compliant tool offers attractive benefits, different features and analysis possibilities. It also provides opportunities for looking at multi-domain models with thermal or physical parameters in addition to electrical parameters.

This paper presented the CIMtoMO XSLT, to provide means to translate PSS®E files to Modelica using the EditiX software. When using EditiX, this process is interactive, transparent, revealing, and extensible. Understanding the tools that make it possible to transform models in the CGMES XML requires one to perform XML analysis, to understand the XSLT structure and common workflow within EditiX, XSLT functions, and to determine how to perform the translation within EditiX. Obtaining parameters from tables through associations is required for parameters that are common to multiple components. In addition, it is possible to exploit templates, functions, and associations to automate the process, as described in this paper.

```
<xsl:function name="gkh:tapPosition">
<xsl:param as="xs:string?" name="tapChangerCode"/>
<xsl:value-of select="$SV/
rdf:RDF/cim:SvTapStep/cim:SvTapStep.TapChanger[sub
string(@rdf:resource,2)=$tapChangerCode]/../cim:SvTa
pStep.position"/>
</xsl:function>
```

Listing 3. tapPosition function.

Acknowledgements

The authors would like to thank Svein H. Olsen of Statnett SF who generated the models from PSS®E ODMS. L. Vanfretti and M. de Castro were supported in part by Dominion Energy and by NYSERDA under agreement number 137940.

References

[1] "IEC 61970-600-1 Ed. 1.0 en:2021 - Energy management system application program interface (EMS-API) - Part 600-1: Common Grid Model Exchange Standard (CGMES) - Structure and rules." Accessed: Oct. 16, 2023. [Online]. Available: https://webstore.ansi.org/standards/iec/iec61970600eden2021

[2] "XSLT, 2nd Edition [Book]." Accessed: Oct. 16, 2023. [Online]. Available: https://www.oreilly.com/library/view/xslt-2nd-edition/9780596527211/

[3] M. de Castro *et al.*, "Version [OpenIPSL 2.0.0] - [iTesla Power Systems Library (iPSL): A Modelica library for phasor time-domain simulations]," *SoftwareX*, vol. 21, p. 101277, Feb. 2023, doi: 10.1016/j.softx.2022.101277.

[4] "IEC 61970-552 Ed. 2.0 b:2016 - Energy management system application program interface (EMS-API) - Part 552: CIMXML Model exchange format." Accessed: Oct. 16, 2023. [Online]. Available: https://webstore.ansi.org/standards/iec/iec61970552ed2016

[5] L. Vanfretti, W. Li, T. Bogodorova, and P. Panciatici, "Unambiguous power system dynamic modeling and simulation using modelica tools," in *2013 IEEE Power Energy Society General Meeting*, Jul. 2013, pp. 1–5. doi: 10.1109/PESMG.2013.6672476.

[6] "A Brief History: The Common Information Model | IEEE Power & Energy Society eNews Update." Accessed: Oct. 16, 2023. [Online]. Available: https://site.ieee.org/pes-enews/2015/12/10/a-brief-history- the-common-information-model/

[7] F. J. Gómez, L. Vanfretti, M. Aguilera, and S. H. Olsen, "CIM-2-mod: A CIM to modelica mapping and model-2-model transformation engine," *SoftwareX*, vol. 9, pp. 161–167, 2019, doi: https://doi.org/10.1016/j.softx.2019.01.013.

[8] F. J. Gómez, M. Aguilera Chaves, L. Vanfretti, and S. H. Olsen, "Multi-Domain Semantic Information and Physical Behavior Modeling of Power Systems and Gas Turbines Expanding the Common Information Model," *IEEE Access*, vol. 6, pp. 72663–72674, 2018, doi: 10.1109/ACCESS.2018.2882311.

[9] "IEEE 14 Bus Model CIM and PSS/E Source Files for Model Transformation Tool Testing," GitHub. Accessed: Oct. 16, 2023. [Online]. Available: https://github.com/ALSETLab/NYPAModelTransformation/tree/maste r/ModelTransf-Tool/Prototype/examples/bus-14

[10] G. K. Halley, "Tutorial for Using EditiX to Analyze and Translate CGMES XML." ALSETLab, Oct. 20, 2023. Accessed: Oct. 30, 2023. [Online]. Available: https://github.com/ALSETLab/XSLTConferencePaper