RiseClipse: why Working at the Model Level is Better for Validating Data Conforming to IEC Standards

Dominique Marcadet
Laboratoire de Recherche en Informatique
Univ. Paris-Sud, CNRS, CentraleSupélec
Université Paris-Saclay
Gif-sur-Yvette, France
dominique.marcadet@lri.fr

Éric Lambert
Département MIRE
EDF Lab

Palaiseau, France
eric.lambert@edf.fr

Abstract— **There is a clear move towards the use of UML (Unified Modeling Language) models for the definition of standards applying to the electrical domain. This is true for the CIM (Common Information Model), and it will also soon be true for the IEC 61850 standard. There are clear advantages to the use of UML: this semi-formal language is an open standard supported by several tools and is quickly understood, at least its class diagram part used in the context of these IEC standards, by people even if they are not working in the software engineering domain. However, several XML (eXtensible Markup Language) based standards are derived from these UML based standards in particular to enable exchange of data. To validate such data, most tools work at the XML level. We will show why it is useful to put back data at the model level for validation or for other purposes.**

Index Terms-- **data exchange, IEC standards, model driven engineering, UML, XML.**

I. Introduction

French utility EDF and the engineering institute CentraleSupélec established in 2012 a joint research institute to prepare for and support the development of smart grids. RISEGrid (Research Institute for Smarter Electric Grids) is dedicated to the study and modeling of smart distribution networks. The research program of RISEGrid covers four areas: study of smart electric systems, observability of the electric system, information systems for smart grids, modeling and advanced simulation. Exchange of data is key for these four areas. This need calls for standards on the format and meaning of data. The electrical domain is highly concerned by this trend, and the IEC (International Electrotechnical Commission) has a long activity on standards related to exchange of data ([1]).

The CIM (Common Information Model, [2]) is a well-known example of such a series of standards. At the top level, it defines with UML (Unified Modeling Language) the different elements of electrical networks: this is the semantic level. The syntactic level is derived from the semantic level using rules to map UML classes and properties to XML (eXtensible Markup Language) elements, allowing for the exchange of data.

Another example is the IEC 61850 series of standards: UML is used, but mainly for documentation purpose, as a way to present the syntactic level defined in XSD (XML Schema Definition language).

CIM and 61850 have been recognized as key smartgrid standards by IEC [3] and are used at different level and different smartgrid domains as described in Fig. 1. In both cases, correct exchange of data coming from different participants and different tools is critical, therefore interoperability tests [4] are conducted to verify that everyone is conforming to these standards. There are several level of verification: the basic level checks that documents are well-formed in respect to XML; the second level verifies that XML element and attribute names are valid; the last level is concerned with semantics: presence of some specific elements, values belonging to some intervals, right number of elements associated with another etc.

While XML tools are perfect for validating the first two levels, they are much less adapted to the semantics validation. The main reason for this is the tree nature of data stored in an XML document, which contrasts with the graph nature of stored data; while there are ways to have XML links that cross trees, theses are fragile compared to the basic containment link implemented by XML trees. Moreover, these cross-links, and even XML containment links are unidirectional, thus they impose complicated algorithms to check some semantics constraints. We argue that the right tools must be used at the right places: if XML is used at the syntactic level, XML tools must be used to validate this level. But when the semantics is defined by another language, UML in this case, data must be brought back to this level for effective validation. We will describe a tool, RiseClipse, dedicated to this task.

The rest of the paper is organized as follows: Section II is dedicated to the CIM, and section III to IEC 61850. We will also present relevant aspects of UML and XML in these two chapters. EMF (Eclipse Modeling Framework) and the RiseClipse tool, which is based on the former and implements the model level validation of data, are presented in section IV. Finally, the paper is summarized and concluded in Section V.

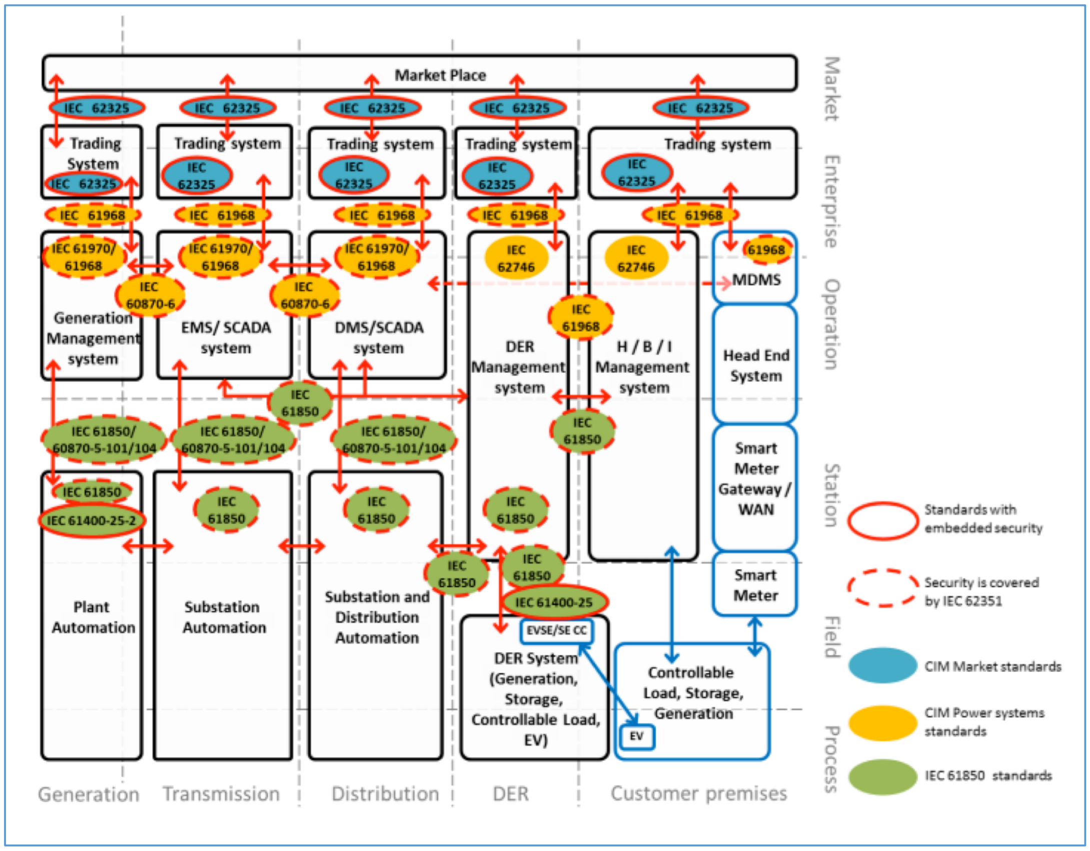

Figure 1. Smartgrid domains

II. Common Information Model

The CIM (Common Information Model) was developed in the context of an EPRI (Electric Power Research Institute) project named CCAPI (Control Center Application programming Interface) and was the model of a network simulator. The need to exchange data between utility companies has become a major need since the start of the deregulation of the power industry. Now, the CIM is an IEC standard, more precisely a set of standards with three of them defining the core semantic model: IEC 61970-301, IEC 61968-11 and IEC 62325-301. There are already several well written descriptions of CIM (see [2] for example), therefore, we will only highlight points which are important in the context of this article.

The CIM is now a very big UML model, with thousands of elements. To enable its effective use, the notion of profile has been defined by IEC: a profile is a subset of the full CIM model that retains only elements that are pertinent in a specific use-case. When several use-case are related, a family of profiles can be defined. For example, CGMES (Common Grid Model Exchange Standard) defines seven profiles to cover the needs in exchange of data at ENTSOE (European Network of Transmission System Operators for Electricity).

The CIM is a semantic model. A model is a description of a system made for a specific objective. A model should show only things that are relevant to this objective and hide the other characteristics of the system. A semantic model is a model done for defining and naming concepts in a domain and links between them. A model can be expressed in natural language, but a formal or semi-formal language is often used to avoid ambiguities. There are a lot of such languages that have been designed for this purpose, but UML is not one of them. UML was initially designed as a tool for designing and developing software. But its class diagram part may be used as a language for defining semantic models, and as it is a standard with several tools available, it is often used for this task. An interesting point in the modeling domain is that a model, more precisely the language used to make models, is itself a system, and therefore can be described by another model: we call it a meta-model. Of course, one can imagine meta-meta-models, and so on.

In CIM, kind of elements used in power systems, like *Fuse* or *Line*, are modeled as classes in UML (Fig. 2) and elements are said to be instances of these classes. These elements have valued characteristics: attributes of classes are used to describe them. For example, the *ACLineSegment* class has a resistance (named *r*) attribute. Classes are organized in a specialization hierarchy: *Fuse*, *Jumper* and others are kinds of *Switch*, all of them have an *open* attribute defined in *Switch*. Finally, UML associations between classes describe potential links between elements: for example, every *ConductingEquipment* is associated with zero or more *Terminal*, and all *Terminal* are linked to zero or one *ConnectivityNode*. A special kind of association may be used to express containment: an *EquipmentContainer* contains several *Equipment*.

In UML, an association is mainly a set of (most often two) ends, each one holds properties like multiplicity or name (in Fig. 2, only the names of the two ends of the association between *Terminal* and *ConnectivityNode* are shown). The only visible property of an association is its name (often as a verb), but it is very seldom used. An important point is that there is no order in the ends, that means that an association is bidirectional at the semantic level.

Using classes and associations defined in CIM, we can describe a power system with instances and links (Fig. 3), that is we can have a model that represents this power system.

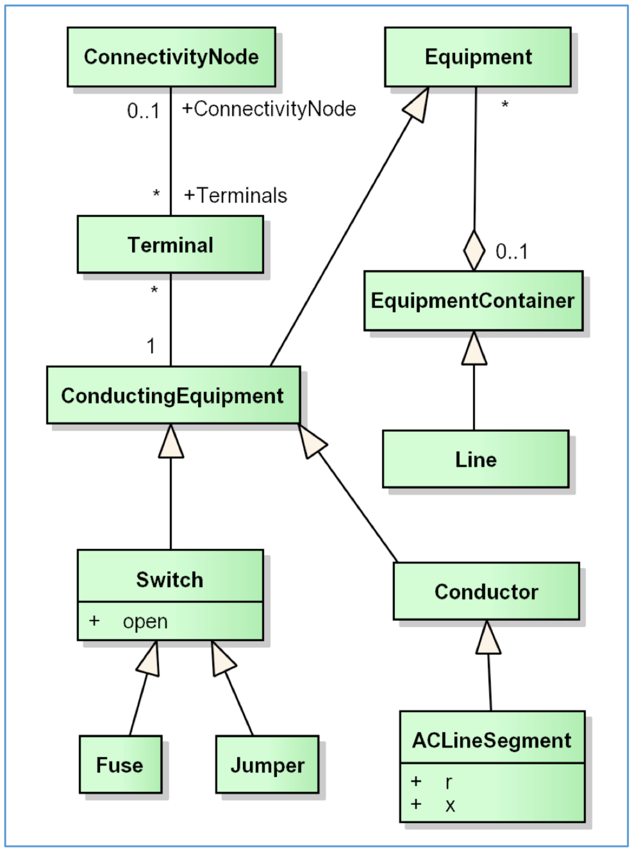

Figure 3. Some CIM classes and associations

CIM is the language used to make such models of power systems. The model of CIM is an UML model, it is the meta-model of the power system. From the system to UML, we have four layers (see Table I).

As a semantic model, the CIM defines an ontology, a set of names (classes, attributes, association ends) and their relationships. To enable the exchange of data conform to this model, a syntactic model, derived from the semantic one, have to be chosen. XML based syntactic formats are now quite common, and the IEC defines two such XML grammars based on the CIM model: one (standard IEC 61970-552) is used for exchanging power system model data and is based on XML RDF, the other (standard IEC 61968-100) is used in the context of enterprise application integration and is based on XML XSD. We will look at the first one because it is the standard used to exchange network data between utilities and chosen in Europe by ENTSOE.

TABLE I. CIM MODELING LAYERS

UML
UML models Model of the CIM language / metamodel of power systems
CIM models / models of power systems
Power systems

RDF (Resource Description Framework) is a W3C (World Wide Web Consortium) specification for modeling of information, one of its serialization format is based on XML. RDF use so-called triples consisting of the subject (what is described), the predicate (name of the property) and the object (value of the property). RDFS (Resource Description Framework Schema) is another W3C specification used to define the vocabulary for RDF; it can describe classes with inheritance and typed properties for these classes. IEC has defined, in standard IEC 61970-501, a mapping from the CIM UML model to an RDF Schema.

Here is a simplified extract of the mapping of some classes and properties in Fig. 2:

```
<rdfs:class rdf:ID="ConductingEquipment">
  <rdfs:label>ConductingEquipment</rdfs:label>
  <rdfs:subClassOf rdf:resource="rdfs:Equipment"/>
</rdfs:class>

<rdf:Property rdf:ID="Switch.open">
  <rdfs:label>open</rdfs:label>
  <rdfs:domain resource="Switch"/>
  <cims:dataType resource="Boolean"/>
</rdf:Property>

<rdf:Property rdf:ID="Terminal.ConnectivityNode">
  <rdfs:label>ConnectivityNode</rdfs:label>
  <rdfs:domain resource="Terminal"/>
  <rdfs:range resource="ConnectivityNode"/>
  <cims:multiplicity resource="M:0..1"/>
  <cims:inverseRoleName
      resource="ConnectivityNode.Terminals"/>
</rdf:Property>
```

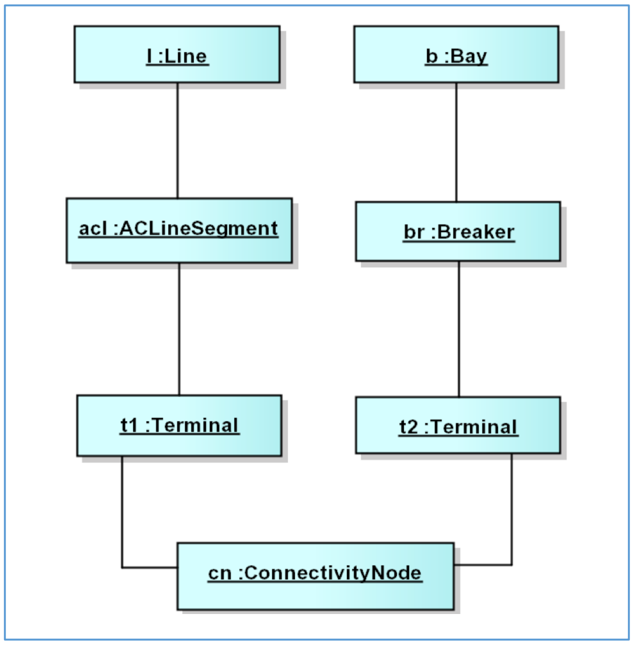

Figure 2. Some CIM instances and links

The mapping is complete for the part of the UML class diagram notation used for defining CIM, including inheritance and opposite relationships (using an extension of RDFS as shown by the use of the *cims* namespace). It has to be noted that this RDF schema is at the semantic level (like the UML model), it defines the vocabulary to be used for exchanging power system model data in XML, not the XML grammar itself. This is in part because there are several ways to serialize RDF triples in XML; the IEC 61970-552 standard defines the so-called CIM RDF XML format as a subset of the RDF syntax.

Here is an extract of the serialization of some instances depicted in Fig. 3:

```
<cim:Line rdf:ID="l"/>
<cim:ACLineSegment rdf:ID="acl">
  <cim:Equipment.EquipmentContainer
                         rdf:resource="l"/>
<cim:ACLineSegment/>
<cim:Terminal rdf:ID="t1">
  <cim:Terminal.ConductingEquipment
                         rdf:resource="acl"/>
  <cim:Terminal.ConnectivityNode
                         rdf:resource="cn"/>
<cim:Terminal/>
<cim:ConnectivityNode rdf:ID="cn"/>
```

We can see that links are saved in only one way, which makes sense because, using the schema, one can infer the other way.

For validating such instance files, standard RDF tools can be used, but they will not take into account extensions made to the RDFS language. In the semantic web domain, another language, which is more expressive than RDFS, is often used: OWL (Web Ontology Language). However, its main domain of usage being resources on the web, the directed link problem is still present: on a web page, it is easy to add a link to another web page, but it is not possible to find all web pages that have a link to your page.

For example, one classical tool to validate CIM XML files is CIMTool ([5]). It uses Apache Jena ([6]), a framework for reasoning with RDF and OWL data. It is able to check directly the cardinality of a property; for example, it will verify that a *Terminal* is always connected to a *ConnectivityNode*. But to check the inverse end of this same association (a *ConnectivityNode* is always connected to at least 2 *Terminal*), some extensions have been made and the rule is written:

```
problem("Isolated node" ConnectivityNode
    "expect two or more terminals. Subject "
    ?n " has less.")
<-(?n rdf:type ConnectivityNode)
   countLessThan(2 * Terminal.ConnectivityNode ?n)
```

It clearly shows that the *ConnectivityNode.Terminals* property is unknown.

This example makes evidence that, while a standard for the syntactic level is needed (and XML and derived standards are perfect for that), a full validation of data up to the semantic level is difficult to achieve when the tool works only at this syntactic level.

III. IEC 61850

IEC 61850 is a set of standards for the design of electrical substation automation and intelligent electronic devices. We will focus on one of them, IEC 61850-6: Configuration language for communication in electrical substations. SCL (Substation Configuration description Language) is the language and representation format for the configuration of electrical substation devices. It includes data representation for substation device entities; its associated functions represented as logical nodes, communication systems and capabilities. The interoperability of IEC 61850 devices and systems is an important goal of this standard, and tests are often conducted to check correct exchange of data.

The IEC 61850-6 standard specifies the exchange format using XML Schema. Even if some UML diagrams are present, they just display some part of the full XML Schema using a graphical notation; they do not define a semantic model.

To illustrate this point, the Fig. 4 shows an extract of some UML classes (which are in fact XML Schema types) present in SCL. The black diamonds used here means composite relationships, and correspond directly to the inclusion of elements in XML. *tNaming*, which is an indirect superclass of all the others except *tTerminal*, enforces the presence of the *name* attribute.

Like in CIM, equipments are electrically connected using

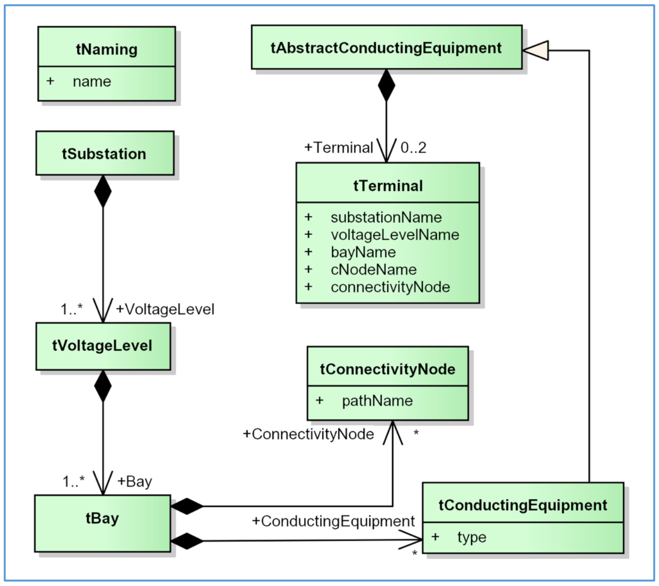

Figure 4. Some SCL classes and associations

the concepts of *tTerminal* and *tConnectivityNode*. However, there is no association between these two classes. While CIM RDF XML uses unique identifiers and references to them to represent vertices in the graph modeling the power system resource, SCL use either containment or specific values of attributes. In the above example, the link between a *tTerminal* and its *tConnectivityNode* is stored in the *connectivityNode* attribute of *tTerminal* which contains the pathname starting from the *tSubstation* name, then the *tVoltageLevel* name, then the *tBay* name and finally the *tConnectivityNode* name. The *tTerminal* must also contain as attributes the individual name of each of these elements; and the *tConnectivityNode* stores also its *pathName*.

A conforming SCL file looks like:

```
<SCL>
 <Substation name="S12" desc="Baden">
  <VoltageLevel name="D1">
   <Bay name="Q1">
    <ConductingEquipment name="I1" type="CTR">
     <Terminal
        substationName="S12"
        voltageLevelName="D1"
        bayName="Q1"
        cNodeName="L1"
        connectivityNode="S12/D1/Q1/L1" />
    </ConductingEquipment>
    <ConnectivityNode
        name="L1"
        pathName="S12/D1/Q1/L1" />
   </Bay>
  </VoltageLevel>
 </Substation>
</SCL>
```

The classical way to validate such a file is to use an XML validator that supports XSD, like Xerces ([7]). However, if the schema can specify some semantic constraints (for example, the uniqueness of *tSubstation* names), it cannot go further, and the validator is for example unable to detect an incoherent value of the *connectivityNode* attribute.

In the case of IEC 61850, we have therefore an implicit semantic level that is not formally expressed in the standard, and validation above the syntactic level given by XSD is left to implementers of tools.

IV. Eclipse Modeling Framework and RiseClipse

We strongly believe that an international standard semantic model defined in order to exchange data between independent parties (with clear rules to deduce the syntactic level from the semantic one) must have operational rules (and therefore tools) to check the conformance of data against the standard. Fortunately, the growing usage of UML as a tool to define the semantic level opens the door for the use of associated standards for the validation task.

In this context, OCL (Object Constraint Language) is the main candidate if we can prove that an operational and non-proprietary tool using this language can be built for this job.

The following sections describe how RiseClipse was designed to fulfill this task. We start with a description of the OMG (Object management Group) modeling architecture, then we present a framework conformed to this architecture, and finally we give an overview of RiseClipse, our tool that implements semantic validation.

A. OMG Modeling Architecture

The MDE (Model Driven Engineering) approach focuses on the use of models for designing systems. It is now recognized that this is the only way to deal with the growing complexity of systems. Different models, and thus modeling languages, are needed depending on the activity in the development process; such an approach can succeed only if there are tools to verify some properties in models, to transform one model to another or to some programing language for the software parts.

The OMG is the main contributor of standards in this domain; it is the publisher of the UML standard. To enable an effective use of an MDE approach, it has defined a four layers architecture (table II).

The UML language is defined using yet another language, MOF (Meta Object Facility) which is reflexive (MOF is defined with itself). MOF can be seen as a subset of UML class diagrams. The need for this M3 layer comes from the need to have several modeling languages besides UML; CWM (Common Warehouse Metamodel) is for example such another language standardized by the OMG. The existence of MOF enable the definition of transformations from a model defined using one language to another model using another language.

TABLE II. OMG Modeling layers

M3	MOF
M2	MOF models. UML (and other modeling languages)
M1	UML models (like CIM)
M0	Systems

UML (and MOF) are semi-formal languages: they are not mathematically defined, but OCL, a quasi-formal language ([8]) is extensively used to complement their abstract and concrete syntax definition. An OCL constraint defined at the M2 layer (the M3 layer is needed for that) applies on elements in the M1 layer. This possibility is widely used in the definition of the UML language to specify for example that the inheritance hierarchy of classes must be an acyclic graph.

Another use of OCL is possible one level down: OCL can be used on the CIM model to restrict the cardinality of some association ends:

```
context Switch inv:
    self.Terminals->size() = 2
```

self represent any instance of the *Switch* class. The *Terminals* property used here is inherited from the *ConductingEquipment* class (the name of this end is not shown on Fig. 2). This constraint defined at the M1 layer applies to objects in the M0 layer, but the M2 layer is needed to know that *Switch* is an instance of an UML *Class*, *Terminals* an instance of an UML *Property*…

B. Eclipse Modeling Framework

Eclipse is an IDE (Integrated Development Environment) which is language neutral (even if it is itself written in Java) and highly extensible.

EMF (Eclipse Modeling Framework) is one of the most used frameworks for MDE, it can be considered as an operational implementation of the OMG Modeling Architecture. For technical reasons, Ecore replaces MOF at the M3 layer, but these two languages are very similar. EMF provides the glue between three worlds: the world of models (with Ecore, a subset of UML), the world of XML (for serialization and deserialization of model instances) and the world of Java (for manipulating model instances). Two of them are automatically generated from the third, which is often an Ecore model, but can also be an XML Schema.

Based on EMF, there are UML and OCL components. It is not possible to use this OCL component to check constraints defined on models in the M1 layer, because the M0 layer contains “real things” which are not accessible to the Eclipse runtime. But constraints defined at the M2 layer can be verified on M1 elements.

C. *RiseClipse*

RiseClipse is born as CimClipse with the idea of using Eclipse OCL to validate CIM XML files. For this to be possible, we had first to move the CIM model from M1 to M2, it is done with a model to model transformation: there exists such tools based on EMF. This transformation has to take care of all kind of tiny details like setting associations navigable in both directions if they were not, deciding which end of the association will be saved… After this step, CIM became a DSML: a Domain Specific Modeling Language. The standard serialization and deserialization format of EMF is based on another OMG standard: XMI (XML Metadata Interchange), which is not the same as CIM XML. We have therefore adapted the generated Java routines to be able to read and write CIM XML files.

This tool allows us to reason at the semantic level, rather than at the syntactic one. For example, we don't care which end of an association is stored in the file, we can access both and write constraints like:

```
context Terminal inv:
    self.ConnectivityNode <> null

context ConnectivityNode inv:
    self.Terminals->size() >= 2
```

We can also take into account the inheritance hierarchy to verify some rules at the right level: to check that a *Line*, as an *EquipmentContainer*, contains only *ACLineSegment* and that an *ACLineSegment*, if in a container, is contained in a *Line*, one can write:

```
context Line inv:
    self.Equipments->forAll(
        e : Equipment |
        e.oclIsTypeOf( ACLineSegment )

context ACLineSegment inv:
    self.EquipmentContainer <> null implies
    self.EquipmentContainer.oclIsTypeOf( Line )
```

CimClipse was presented at a CIM User Group meeting in 2010 in Milano, and has been used during CIM transmission interoperability tests hosted by ENTSOE in 2010 and CIM distribution interoperability tests hosted by EDF in 2011. It was also enhanced with support for CIM difference files (a CIM XML derived IEC standard which allows for exchanging only the difference between a source and a target model).

The support for CIM profiles and group of profiles has also been added. In fact, a profile being a restriction of the full model is evidently implemented by a set of OCL constraints. However, the need to save data in different files when sub-profiles of a group are used can only be done if some metadata is added to elements of the CIM model. This solution was also used to support extensions to the CIM model like the ones specified by CGMES.

In 2013, we thought that the same approach could be used to validate SCL files. We first started with the XML Schema, letting EMF generate an Ecore model and the corresponding Java code. We were able to load SCL files, and write some OCL constraints, but we were limited by the missing links between objects resulting from the low level of semantic information in the schema: the inclusion of XML elements was represented by directed links from the parent to the children, and implicit links (those represented by values of attributes) were not accessible.

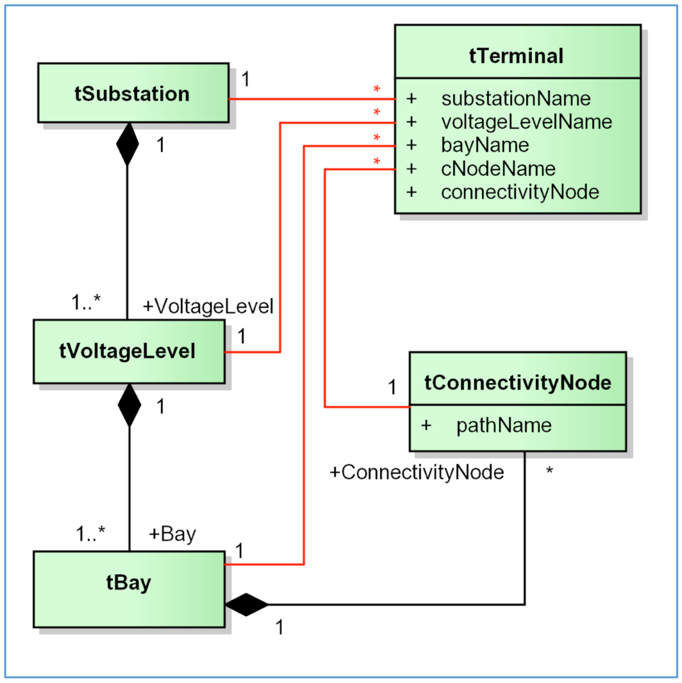

Figure 5. Some SCL classes and associations

In 2014, we decided to refactor the tool, so that any language like CIM or SCL can be added to the core functionality, and named it RiseClipse because its development is done inside the RISEGrid institute. We also restarted the SCL support by making a clean Ecore model for it with all the semantic information that was missing in our first try: inclusion links were navigable in both directions, and explicit links were added to make the model more navigable. Fig. 5 shows in red some of these new explicit associations that have been added to our SCL model.

As we were working at the model level, we were able to use some EMF standard metadata, like the *transient* characteristics of a property that, if true, means that its value must not be saved when serialization is done. The properties added by these new explicit links had therefore no consequences on the serialization step. Indeed, the construction of these links, which is done when the SCL instance file is loaded, is part of the validation goal because an element identified by some specific value of an attribute must be found for the link to be created.

RiseClipse was successfully used during the IEC 61850 IOP (interoperability testing) in Brussels, September 2015. It was able to detect errors that other XSD based tools cannot detect. An example of such an error is the number of *tTerminal* in a *tConductingEquipment*: this number depends on the kind of the equipment, which is given by an enumeration. For a circuit breaker, the OCL constraint can be:

```
context ConductingEquipment inv:
    self.type = 'CBR' implies
    self.Terminals->size() = 2
```

This kind of validation cannot be done with XSD based validators because such tools are unable to specify a constraint that depends on the value of an attribute. This is however an effective semantic rule given by the IEC 61850-6 standard.

RiseClipse (and CimClipse before) is partly developed by students of the French engineering school CentraleSupélec. The tool can be used either in the Eclipse environment, for navigation in the model and interactive validation, or in a command line context for batch validation. We expect to be able to release it as open source soon, it will be available at http://riseclipse.foundry.supelec.fr.

V. CONCLUSION

We have demonstrated that, when an information model is used to defined standards for exchange of data, validation of such data is better done at the semantic level. We have shown that, when UML is used as the language to define the information model, OCL is the perfect companion for expressing these constraints. We have described RiseClipse, an open source tool that can be used to validate such constraints without being bound to a specific standard.

Future works on RiseClipse are planned such as using it to validate new network data sets related to CIM distribution profiles and adding new models to support EDF internal needs. A graphical display of networks (Diagram Layout profile of CGMES) was also developed on CimClipse, and will have to be re-integrated into RiseClipse.

RiseClipse is also a complementary tool of MODSARUS which is an Enterprise Architect add-in which allows to define profiles from any UML model. The way to automate the generation of OCL rules from MODSARUS that could be used by RiseClipse is also foreseen.

Concerning the IEC standards, an ongoing work is done for the so-called CIM-61850 harmonization, it includes the definition of an UML model for the whole IEC 61850 set of standards. MultiSpeak – CIM Harmonization have also been initiated by IEC. As RiseClipse is able to work simultaneously with models from these different standards, we have an interesting challenge for providing solutions to help for maintaining coherency between all these models that partially represent the same system.

REFERENCES

[1] IEC 62357-1 Reference Architecture for Power System Information Exchange

[2] EPRI Common Information Model Primer Third Edition 2015 Technical Report

[3] IEC Smart Grids Standardization Roadmap, IEC Smart Grids Strategy Group (SG3), 2010

[4] https://www.entsoe.eu/major-projects/common-information-model-cim/interoperability-tests/Pages/default.aspx

[5] http://www.cimtool.org

[6] https://jena.apache.org

[7] https://xerces.apache.org

[8] Achim D. Brucker, Frédéric Tuong, and Burkhart Wolff. Featherweight OCL: A Proposal for a Machine-Checked Formal Semantics for OCL 2.5. In Archive of Formal Proofs, 2014. http://afp.sf.net/entries/Featherweight_OCL.shtml, Formal proof development