Overcoming Challenges Using the CIM as a Semantic Model for Energy Applications

Andrew Crapo (crapo@research.ge.com)[1]

Katrina Griffith (KatrinaM.Grifith@ge.com) [2]

Ankesh Khandelwal (ankesh@cs.rpi.edu) [3]

John Lizzi (lizzi@research.ge.com) [1]

Abha Moitra (Abha.Moitra@research.ge.com) [1]

Xiaofeng Wang (wang.xiaofeng@ge.com) [2]

[1]GE Global Research, Niskayuna, NY [2]GE Energy, Melbourne, FL [3]Rensselaer Polytechnic Institute, Troy, NY

Keywords: Smart Grid, Interoperability, Semantic Models

Abstract

CIM is the key to smart grid interoperability. This being the case, we are experimenting with different approaches to utilizing the CIM as a semantic model for Energy applications. In this paper we identify some of the challenges we have encountered applying an OWL representation of the CIM to a common industry application for network model validation. Some challenges have to do with differences between UML, the language in which the CIM is maintained, and RDF/OWL, the semantic representation we have chosen for our experiment. We highlight some of differences between UML and OWL and discuss what they might mean in a smart grid context and the advantages and disadvantages of using each representation in Energy applications. Still other problems have to do with the processes around translating the CIM models from UML to OWL and managing model changes. Finally we propose some necessary conditions for addressing these challenges.

1. INTRODUCTION

It has been proposed that shared semantic models will be the foundation of interoperability for the smart grid [1]. This is supported by the GridWise Architecture Council's Interoperability Framework, where semantics is not only a central layer but also addresses many of the cross-cutting issues [2]. Semantic models ensure that the individual data sources do not define the semantics and the syntax of the data [3]. Rather these are defined by shared models. The International Electrotechnical Commission (IEC) Common Information Model (CIM) represents the most complete and widely accepted model for generation, transmission, and distribution of electrical energy. Using the CIM as a starting point, we are evaluating the potential of semantic technology, as envisioned by the Semantic Web community, as a basis for building model-driven applications for smart grid functionality such as network model validation and network tracing. Our network tracing use case focuses on determining what equipment in an electrical network is topologically (electrically) connected to other equipment. This paper reports our initial experiences and conclusions from our investigation. We have found both promise and challenge in applying semantic technology to our smart grid use cases. Some challenges have to do with the CIM and the processes around its evolution. Other challenges have to do with applying semantic technology and suggest the need for better tools to support modeling and model application.

2. THE CIM AS A SEMANTIC MODEL

The CIM is expressed and maintained in the Unified Modeling Language (UML). However, the Web Ontology Language (OWL) is better suited to our objectives of model-driven applications and interoperability for several reasons. An OWL model can be checked for consistency and validated against given criteria using available reasoners whose behavior is specified by open standards. OWL models provide a conceptual foundation upon which to layer domain knowledge captured as rules. For example, domain knowledge for our use case included how to trace a network taking into account switch positions, transformers, and phase.

While there is an IEC standard covering the generation of RDF Schema from the CIM-UML, the standard does not address generation of an OWL version of the CIM [4]. We were able to obtain two versions of the CIM in OWL representation, which we used in our research. We are aware

that these translations were not guided by an accepted standard. In fact, one of the purposes of this paper is to suggest some of the issues that need to be addressed by such a standard.

While a model in either UML or OWL might be considered a "semantic model", there are some important differences. The two modeling languages were designed for different purposes from differing points of view and have different strengths and weaknesses. UML was born in the software engineering domain as a language for designing object-oriented software artifacts. It has also been used extensively for data modeling. OWL has its origin in formal modeling and logical inference and classification. It is instructive to explore some of the important similarities and differences between UML and OWL, especially as they relate to modeling for the smart grid. Unless otherwise noted, the primary reference for the comparison that follows is the *Ontology Definition Metamodel, Version 1.0* [5].

The meta-model of UML (expressed in OMG's Meta-Object Facility, or MOF) captures four main concepts: 1) classes, 2) associations, 3) datatypes, and 4) packages. Also expressible in MOF [6], the OWL meta-model includes 1) classes, 2) properties, 3) instances, 4) literals, and 5) ontologies. Each meta-model includes other concepts such as cardinality. There is a reasonable correspondence between the two meta-models. The differences of significance are in the details.

A UML class and an OWL class are both based on set theory and so correspond reasonably well. However, OWL's set semantics are more complete. UML associations can be translated to OWL properties although the UML meta-model is much more complex. UML associations are defined only in the context of the classes at the end points whereas OWL properties are first class citizens and can exist independent of any domain or range specification. Individuals in OWL can exist and have properties independent of any class membership while UML runtime instances exist only at the MOF M0 level and so exist as instances of a specific class. A UML package, which corresponds to an XML namespace, is more or less equivalent to an OWL ontology, although the details of naming and importing are not identical. UML lacks a formal model-theoretic semantics and what might be expressed as class restrictions in OWL require constraints in the Object Constraint Language (OCL) for a UML model. As OCL has neither a formal model theory nor a formal proof theory, automated reasoning over UML models is not possible, or at least not well defined.

It has been suggested that the CIM will eventually be expressed and maintained as a set of ontologies within a federation of ontologies [7]. Such a modular and integrated view of the CIM leverages the modular and extensible capabilities of OWL and the envisioned distributed nature of ontologies in the Semantic Web. OWL is designed to facilitate the organization of models in a hierarchy of imported models with "core" or "upper-level" models at the root. More specialized (e.g., CIM package-specific) or lower-level models then extend the root models with additional classes, subclasses, properties, rules, etc.

A hierarchical model structure is consistent with the nature of UML, where packages create a hierarchy of imports. Each package corresponds to a single namespace. Figure 1 shows some of the package structure of the UML CIM model. Of course the overall model structure is not always a hierarchy—packages may import each other, directly or indirectly, creating a lattice of submodels.

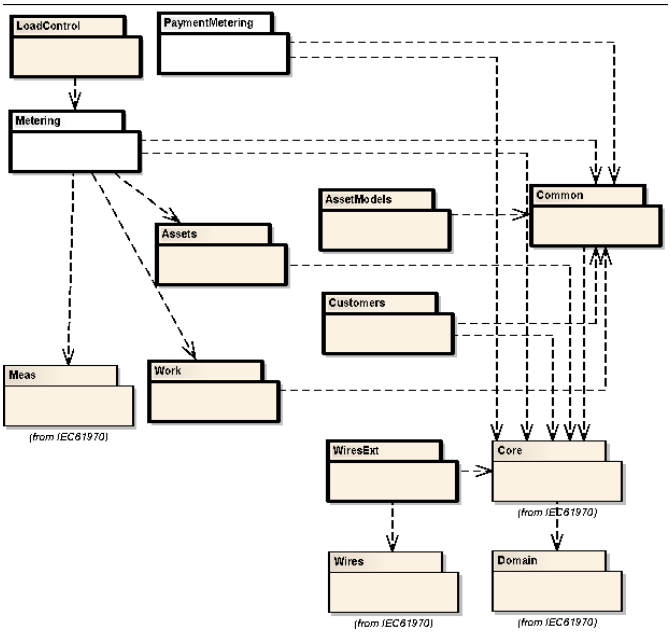

Figure 1: Partial Package Structure of CIM in UML

At a more abstract level, Figure 1 in Becker and Saxton [3, pg 3] shows three-layer enterprise semantic model architecture. The CIM resides in the top Information layer while profiles reside in the next-lower Contextual layer and "define a subset of the models in the Information layer needed for a particular business purpose". A UML profile can both extend a UML model and identify a subset of the base model in an additive manner [8]. Since UML profiles can import other profiles, it should be possible to use UML profiles to create a set of OWL models which, via the owl:imports functionality, form a hierarchy of models with the most specific models being created by profiles not imported by any other profile. OWL imports will also support a lattice model structure.

In summary, while there are differences in UML and OWL models, the UML representation of CIM provides most of

the information that one would want in a set of OWL models, provided that the translation preserves modularity.

3. BENEFITS OF USING CIM IN OWL FORMAT

Several benefits of using an OWL version of the CIM are apparent from our research.

3.1. Extensibility of the CIM

We found it very easy to extend the CIM model by defining new subclasses of existing CIM classes and by defining additional properties on existing CIM classes. These extensions, defined in new ontologies that imported the CIM OWL model, worked very well as the basis for composing rules that utilize both CIM concepts and new concepts and/or new properties in the same rule. This made it easy to make small, independent extensions of the CIM to achieve specific purposes.

3.2. Executable Models

One of the biggest benefits of using OWL was the executable nature of the models. Using the SADL-IDE [9] as a development environment, we generated test cases that imported the appropriate CIM concepts and/or extensions. (SADL models are translated to OWL and rule files, which are then used by the reasoner to do logical entailment, rule-based inference, and answer queries.) A test case explicitly tested for specific inferred values and/or included queries that displayed inference results. Test case results were accompanied, if desired, by derivation information, at either a shallow or a deep level, to provide a logical explanation of how values were derived. This made the building and testing of models and rules a very interactive process, leading to a much shorter design cycle than one would expect if UML models were used to generate source code, which would then require additional test code to be written to exercise and validate the models.

3.3. Model Validation

Validation of OWL models can occur at multiple levels. At one level an OWL reasoner, such as Jena or Pellet, can detect logical inconsistencies in models or instance data. At another level a modeler can explicitly generate a set of validation rules that test instance data to make sure that it conforms to cardinality or other constraints. We found it relatively easy to do instance data validation using either method or a combination of the two.

3.4. Integrating Domain Logic as Rules

In our investigations, we found that we were able to incorporate domain logic into rules in a straightforward manner for both network model validation and network tracing. For example, we were able to formulate phase compliance checks as part of network tracing.

4. CHALLENGES USING THE CIM

Some of the challenges we encountered in our research were due to the nature of the OWL version of the CIM model. In large measure these issues arose from the way the CIM was translated from UML to OWL rather than to fundamental differences between UML and OWL.

4.1. Modularity and Namespaces

Modularity is very import in large, complex models. Proper modularization of a model enables parallel development and maintenance of the individual submodels, makes it easer for people to grasp the overall model structure, and allows appropriate parts of the model to be utilized for different purposes without requiring that extraneous model elements be included. Of course the interfaces between the modules must be well-managed.

The OWL CIM models we used had all available namespaces (packages) mixed in a single model file with no use of the namespace (base URI) of this file in the file content. This was unexpected based on our previous experience using hierarchical models expressed in OWL. It necessitated that we load the entire model regardless of what parts were actually needed. It also made it difficult to comprehend what was in the model as there was no higher-level view of model structure readily available.

This problem clearly derived from the translation of UML to OWL rather than in the CIM itself. However, it did highlight the need to preserve the CIM structure in the translation process so that the designed capacities of OWL to organize models into a lattice or hierarchy can be fully leveraged. Interestingly, in one of the translations we utilized, the OWL model included a namespace for "UML" and within that namespace defined a Package class used only in values of the annotation property rdfs:isDefinedBy. (Annotation properties are used in OWL to document the model but are not normally used in logical inference.) Nesting of annotations was used to capture the UML package hierarchy. For example, the CIM concept HydroTurbine was defined by the Package instance Package_GenerationDynamics, which in turn was defined by the Package instance Package_Generation, which in turn was defined by the Package instance Package_IEC61970. As a result, the package hierarchy of the UML model was not actually reflected in a hierarchy of ontologies and OWL imports. As annotations, the package structure was explicitly excluded from normal reasoning processes. It is important to generate OWL models that enable the modularity and reasoning capabilities that are OWL strengths.

4.2. Datatypes

Another challenge was found in the inconsistent handling of datatypes. When datatypes such as strings, numbers, dates,

etc. specified in the UML model are translated to OWL, it is highly desirable that they appear as XML Schema primitive datatypes. This allows the built-in capabilities of reasoners, rule engines, and query engines to properly handle routine tasks such as arithmetic operations, comparisons, and regular expression matching. It also permits model consistency checking to be more useful. The alternative is to custom-build new plug-in capabilities to handle essentially equivalent but model-specific datatypes or classes.

Vestiges of this problem were apparent in our OWL CIM models. In one version, an rdfs:Datatype named String was defined in the model namespace but no further definition was provided, making it impossible to relate it to an xsd:string. A comment annotated String: "A string consisting of a sequence of 8 bit characters…" In the other version of a CIM OWL model, properties were defined as being of type owl:DatatypeProperty but then given an owl:Class as a range. For example, the owl:DatatypeProperty TransformerWinding.xground had range Reactance, an owl:Class. This violates the definition of DatatypeProperty in OWL. The latter was clearly a translation issue but the former translation may be consistent with IEC 61970-501 [5]

5. CHALLENGES USING SEMANTIC MODELS

Some challenges we encountered were due to the nature of semantic modeling itself, and not CIM or smart grid specific.

5.1. Efficient Representation of Rules, Efficient Queries

We found that performance of rules and queries depended dramatically on their exact formulation Getting these formulations right is knowledge that may be difficult for a domain expert to acquire. Note that by rule we mean a set of premises, also known as the rule body, associated with a set of conclusions, also known as the rule head. If all of the conditions in the premises are satisfied, the rule "fires" and the conclusions are effected, potentially inserting new triples into the knowledge base (triple store). The individual conditions of the premise and the individual consequences of the conclusions are referred to as clauses. While this section addresses only rules, queries that select information from the knowledge base are subject to similar differences in performance depending on how they are structured and therefore are candidates for optimization.

We found that the techniques and styles of rule writing for efficiency can be broadly classified into two categories – those that are independent of the reasoner being employed and other techniques that are specifically geared to a specific reasoner. Some of the reasoner independent techniques correspond to well known techniques from software engineering, database query optimization, etc. Reasoner-dependent techniques were driven by reasoner characteristics, i.e., whether the reasoner used forward or backward reasoning. Some of the techniques for writing rules efficiently are as follows.

- Reorder the clauses in order to reduce the amount of matching instance data as early as possible. For example, if one clause matched on (a) the property cim:ConductingEquipment.ClearanceTag having a value and another class matched on (b) the property cim:ConductingEquipment.Terminals having a value, and it was known that few instances would match clause (a) while many more instances would match clause (b), then clause (a) should appear first.
- Refactor the rules to pull out common clauses from multiple rules as a separate rule so that they are evaluated fewer times.
- Convert expensive checks into equivalent but less expensive checks if possible. For example, if the type can be inferred by checking for a property assertion, then expensive type inference through subclass inference can be avoided.
- Formulate recursive definitions for predicates as a combination of a non-recursive part (also known as a stop predicate) and a recursive part.
- Use tabling for recursive rules if backward rules are being used. We found that left recursion (evaluating recursive clause before stop predicate) works better for reasoners that support tabling. If the reasoner doesn't support tabling then right recursion (evaluating stop predicate before recursive clause) gives better performance.
- Some reasoners natively support OWL inference, but if one is interested only in partial OWL inference then turning off native support and introducing explicit rules for OWL inferences of interest may improve performance.
- If the reasoner supports backward rules (goal directed reasoning), convert forward rules into backward rules whenever possible.
- In backward reasoning, duplicate rules and reorder clauses differently for different combination of variable bindings. The order should be dictated by the variables that are bound. For example, the rule that infers the transitive "ancestorOf" relationship can be duplicated for four different scenarios. When the ancestor is bound, the following rule is used:

```
#left bound
[ANC-lb: (?p1 ancestorOf ?p3) <-
 bound(?p1),
  (?p1 ancestorOf ?p2) ,
  (?p2 parentOf ?p3)]
```

Whereas when the descendant is bound, the following rule is used:

```
#left unbound
[ANC-rb: (?p1 ancestorOf ?p3) <-
  bound(?p3),
  (?p2 ancestorOf ?p3) ,
  (?p1 parentOf ?p2)]
```

Note that the ordering is such that the clause with a bound variable occurs before the rest of the clauses. This helps in reducing the matching instance data. When neither of the variables, or both the variables are bound the order does not matter.

- Reorder "not" and boolean built-ins for late but eager evaluation. That is, these clauses should be ordered so that they appear as soon as all the needed variables are bound.

5.2. Open World versus Closed World

OWL reasoners normally operate under an open world assumption (OWA). This essentially means that something cannot be concluded to be false just because it isn't currently known to be true. Geared towards distributed knowledge in a World Wide Web, this makes sense. It is unlikely that one can ever assume that all facts are known—we know what we know and not anything else (yet).

However, the OWA can lead to some initially surprising results. For example, we placed a cardinality restriction on cim:EnergyConsumer requiring it to be related by cim:Terminal.ConductingEquipment to an instance of cim:Terminal. Then we created a test case with an instance of cim:EnergyConsumer without such a relationship and asked the reasoner to validate the model. The lack of any complaint was surprising until we realized that under the OWA we could not expect the reasoner to conclude that just because it didn't know about such a relationship did not mean it could conclude that there wasn't one and so the cardinality constraint was not identified as a violation. Note that we were able to implement constraint checking on existing data using a custom Jena built-in function as a validation rule premise. The built-in, which is a black box to the reasoner, views the data under the Closed World Assumption (CWA).

5.3. Understanding Negation and How It Can Be Used

OWL is based on classical negation, which is monotonic under the OWA. This means that if something is proven to be false, addition of new information does not change that. Under the CWA negation is non-monotonic. There are algorithms of differing complexities to evaluate rules with non-monotonic negation. The difference in reasoning approach (algorithm) leads to differences in the semantics of rules with negation, and differences in the semantics leads to differences in expressiveness.

Examples of different semantics/negations include, in order of their expressiveness: monotonic negation, predicate-stratified negation, locally-stratified negation, well founded semantics (WFS), answer set semantics, and stable model semantics. The latter 3 are most general and equally expressive. When an algorithm is applied to rules with higher expressiveness, non-sound results may be deduced.

6. REASONER/RULE ENGINE EVALUATION

Network tracing was our most complex use case from a performance and scale perspective and was utilized to evaluate possible reasoners/rule engines. The network tracing use case consisted of two parts. The first, simpler use case was focused on being able to define the set of instances of cim:ConductingEquipment (CE) "directly connected to" a given instance of CE. The second, more complex use case was defined as "topologically connected to". This use case focused on defining the set of instances of CE electrically connected to a given instance of CE. Both use cases were executed using data sets ranging from 10,000 to 240,000 triples.

We evaluated several reasoners/rule engines for potential use in executing our use cases. These fell into three categories: 1) Java-based generic rule engines, 2) general-purpose logic programming systems, and 3) more targeted "semantic" reasoners (reasoners that accept RDF data and natively support OWL semantics). The goal of this evaluation was to illuminate the CIM model execution landscape with respect to these technologies and key attributes of interest (execution time, scalability, etc.), and to subsequently direct the next generation of proof of concept development.

In the first category we evaluated Drools and JRules. In each case, we translated the CIM-OWL model into Java code (classes) and a network data set containing about 40,000 triples into an in-memory collection of inter-related Java instances accessible to the rule engine. While these rule engines worked well on very small problems, we were unable to get either to work on a larger data set (> 25,000 triples) without running out of memory, even for the simpler use case (directly connected) running on a 64-bit machine.

In the second category we evaluated XSB and YAP. For a relatively large data set, YAP ranged from very slow (> 30 minutes) to quite fast (~10 seconds) on the very next run, reflecting its adaptation of indexing to the problem at hand.

In the third category we initially considered AllegroGraph®, OntoBroker®, Pellet/SWRL, and Jena. We did more extensive testing on AllegroGraph®, OntoBroker® and Jena. AllegroGraph® provided query results for "topologically connected to" on a data set of over 130,000 triples in approximately 20 seconds. On the same data set, Ontobroker® (without performance optimization) gave asymmetrical performance: ~2 seconds for left-bound (EC1 topologicallyConnectedTo ?X) vs. ~78 seconds (36 secs on a 64 bit machine) for right-bound (?X topologicallyConnectedTo EC1) queries. After employing several of the techniques described in Section 5.1, Jena answered the same query on a data set of 180,000 triples in ~5 seconds.

It is important to note that these rule languages and engines employ different reasoning methods and support different expressivities for negation. SWRL doesn't support any form of negation, Pellet performs sound OWL reasoning and supports classical negation, Ontobroker® supports WFS negation, and XSB, YAP and AllegroGraph® are Prolog implementations that support stable model semantics (default negation). Jena, through built-ins functions, allows a rule premise to test for the absence of a triple matching a simple pattern in the model. Drools and JRules support negation that can be referred to as being non-logical, because the evaluation of rules is not based on (mathematical) logic.

Furthermore, although XSB, YAP and AllegroGraph® are Prolog implementations, and Jena evaluates backward rules in similar fashion, XSB, YAP and Jena support tabling of predicates but AllegroGraph® does not..

We initially scoped the technology landscape along five dimensions of interest:

- Model storage – CRUD (create, read, update, delete) performance, support for transactions, security, storage type, etc.
- Model import/extension/development - model harmonization, storage standards support, support for distributed models, modularity, batching capabilities, etc.
- Model interaction – usability for subject matter experts (model construction, validation, maintenance), support for model validation and inference, model query expressivity and standards support.
- Business rule extension development – the ability to leverage existing models to build, validate, and execute rules.
- Business rule management and integration – the ability to manage and integrate business rules into existing components and systems.

Each of the dimensions of interest was further broken down into individual attributes with defined scales to be used in scoring the various technologies.

The use cases were utilized to evaluate the aforementioned technologies versus each attribute. Our initial findings indicate an integrated stack of technologies (leveraging a back-end triple store coupled with one or more semantic rule engines, and a SADL front-end) will provide the best performance for our attributes of interest. Findings indicate Ontobroker® and AllegroGraph®, coupled with a SADL front end scored highest, and warrant the next level of evaluation. We've also found the forward-chaining RETE-based rule engines do not provide the execution performance required for our use cases at the scale of interest due to runtime memory limitations. This is consistent with findings from previous benchmarking activity [10].

7. MINIMAL REQUIREMENTS OF SEMANTIC CIM, DESIRABLE TOOLS CAPABILITIES

Our research to-date suggests that a standard for OWL generation from UML CIM is very important and that this standard should ensure that the OWL generated is usable by semantic tools ranging from model viewers/editors to reasoner/rule engines and query engines. In addition, our experience indicates that the success of semantic technology as a modeling paradigm for smart grid will be significantly enhanced if the tools enable domain experts to build models and capture domain rules without being required to gain deep knowledge of the intricacies of reasoning or rule and query efficiency. We make the following recommendations.

7.1. Cleaner Tie to Standard Datatypes

The IEC standard for generation of RDF from UML (IEC 61970-501, see [4]) does not appear to make the connection to XML Schema datatypes. It is important that the standard for generation of OWL from the UML CIM make that connection so that the resulting OWL models will be subject to standard OWL model validation, consistency checking, reasoning, and querying. It appears that the groundwork for this is being laid as the CIM|EA Web site reported on September 1, 2010, "The most exciting new addition to the tool for this public release is the inclusion of XSD support…" (see http://www.cimea.org/).

7.2. Modularity and Extensibility

Modularity is key to a model's ability to be maintained and used by a diverse set of stakeholders. That modularity must allow "slices" of the model to be used and extended for specific purposes. The package structure of UML has the capability to translate to a lattice or hierarchy of OWL models that can be used in a federation of ontologies [7]. It is essential that the translation produce a set of OWL models that in every way maintain the modularity and structure of the CIM UML model.

7.3. Versioning and Feedback Mechanisms

While this requirement for a robust shared model is not a direct observation of our research, we will include it for completeness. Versions of CIM models should be versioned using the model versioning mechanisms so that reasoners and other tools that load imported models can verify that supported versions have been retrieved. A relatively primitive version mechanism is part of OWL and is a good starting point. An importing model can specify what version(s) of an imported model are acceptable.

CIM model users will extend the CIM models to meet their specific objectives. This will include creating subclasses of a CIM class when finer granularity is needed and/or adding new properties when additional information about class members must be modeled. Some of these extensions will be model fragments that would be widely useful and should therefore be factored into future versions of the CIM either by direct contribution to core submodels or by providing shared user contribution models. The process for consolidating feedback to create next versions should be well-founded and well-documented so that all stakeholders can contribute to the shared models. Since this is a problem common to semantic modeling in other domains, a solution should be synthesized in cooperation with the larger semantic community.

7.4. Automatic Optimization of Rules and Queries

One highly desirable capability of a semantic modeling environment is that it reformulate rules and queries to be efficient in the target-reasoning environment. One of our lessons learned is that a high level of skill is required to tune rules and queries and it is unlikely that smart grid domain experts will want to or have the time to become expert in this activity. Just as SADL seeks to make model and rule expression as natural and easy as possible, the next version of SADL will pair a rule/query translator with each reasoner plug-in so that translation from the English-like representation of SADL to the target representation will result in a rule or query that performs well in the target execution environment. Our initial assessment is that this goal should be achievable in at least the majority of cases. Work in this area is planned in our future activities.

8. SUMMARY AND CONCLUSIONS

In this paper we report some of our observations from applying an OWL version of the CIM to network model validation and network tracing use cases. We found semantic technology to be promising because it allows standard, shared models (the CIM) to be easily extended for specific purposes (our use cases) and made executable for interactive development. We discovered that much better translations of the CIM from UML to OWL are needed so that model structure is maintained and so that standard XML Schema datatypes are used. We also discovered that it is insufficient to provide environments for model development and extension and rule/query authoring but that the environment should also optimize the translation of rules and queries to perform efficiently in the target reasoner/rule engine/query engine. This capability is important if domain experts are to be successful in building and maintaining the complex models necessary to deliver the promise of the smart grid across its many facets.

References

[1] Crapo, Andrew, Xiaofeng Wang, John Lizzi, and Ron Larson. *The Semantically Enabled Smart Grid* (2009). Grid Interop 2009. http://www.gridwiseac.org/pdfs/forum_papers09/crapo.pdf (accessed September 9, 2010).

[2] *Interoperability Context-Setting Framework*, GridWise™ Architecture Council Interoperability Framework Team, March 2008. http://www.gridwiseac.org/pdfs/interopframework_v1_1.pdf (accessed November 13, 2010).

[3] Becker, David and Terrence L. Saxton. The Missing Piece in Achieving Interoperability—a Common Information Model (CIM)-Based Semantic Model, *Grid-Interop Forum 2007, pg 125-2*. http://www.gridwiseac.org/pdfs/forum_papers/125_paper_final.pdf (accessed September 9, 2010).

[4] *IEC 61970-501 Ed.1: Energy management system application program interface (EMS-API) – Part 501: Common information model resource description framework (CIM RDF) Schema*. Available from IEC Store, content description at http://webstore.iec.ch/preview/info_iec61970-501%7Bed1.0%7Den.pdf (accessed September 15, 2010).

[5] *Ontology Definition Metamodel, Version 1.0.* (May 2009). http://www.omg.org/spec/ODM/1.0/PDF/ (accessed September 9, 2010).

[6] Brockmans, Saartje, Peter Haase, and Boris Motik. MOF-Based Metamodel, W3C Wiki document, http://www.w3.org/2007/OWL/wiki/MOF-Based_Metamodel (accessed September 14, 2010)

[7] Neumann, Scott, Jay Britton, Arnold DeVos, and Steve Widergren. "Use of the CIM Ontology", *DistrubuTech 2006.* http://www.uisol.com/uisol/papers/Use_of_the_CIM_Ontology_DistribuTech_2006_Paper.pdf (accessed September 13, 2010) (see also accompanying presentation at http://uisol.com/new/wp-content/uploads/2008/05/use_of_the_cim_ontology_distributech_2006.pdf)

[8] *Catalog of UML Profile Specifications*. http://www.omg.org/technology/documents/profile_catalog.htm (accessed September 9, 2010).

[9] Semantic Application Design Language (SADL), Open Source project on Source Forge, overview at http://sadl.sourceforge.net/sadl.html.

[10] Liang, Senlin, Paul Fodor, Hui Wan, Michael Kifer. *OpenRuleBench:An Analysis of the Performance of Rule Engines,* WWW 2009, Madrid. http://www2009.org/proceedings/pdf/p601.pdf (accessed September 15, 2010)

Biography

Andrew Crapo received a B.S. in Physics from Brigham Young University in 1975, an M.S. in Energy Systems from the University of Central Florida in 1980, and a Ph.D. in Decision Sciences and Engineering Systems from Rensselaer Polytechnic Institute in 2002. He is a senior professional information scientist at the GE Global Research Center where he has worked since 1980. His work has focused on applications of information science to engineering problems including applied artificial intelligence, human-computer interactions, and information system architectures. More recently he has focused on modeling and the application of Semantic Web technologies to engineering and business problems.

Katrina Griffith is an NTI Technologist with GE Energy's Software Systems Engineering Group. Her current work focuses on the application of data services, data modeling and semantic technologies for GE's Smart Grid and Total Plant Optimization initiatives.

Ankesh Khandelwal is a PhD candidate, in his 3rd year, at RPI under Professors James Hendler and Deborah McGuinness. He is most interested in rule based inferencing over the web, in particular distributed rule based inferencing, although his current focus is on developing theory to declaratively model natural and scientific processes. In the past he has worked on scalable OWL inferencing, inference engines for OWL 2 RL and OWL 2 QL, and formalization of a rule based policy language (Accountability in RDF- AIR). He was involved with this project as a summer intern at GE Global Research.

John Lizzi received a B.S. in Computer Science from Siena College in 2001, a M.S. in Computer and Systems Engineering from Rensselaer Polytechnic Institute in 2003, and a M.B.A. from the State University of New York at Albany in 2008. Since 2000, John has been working as a Research Scientist in the Computing and Decision Sciences Group at General Electric Global Research, in Niskayuna, NY. John has worked on developing technology and solutions in a variety of domains including maintainability engineering, air traffic management, television broadcast operations, healthcare, and energy services. His primary interests include software architecture, enterprise integration, modeling, and simulation.

Abha Moitra received a M.Sc. in Physics from Birla Institute of Technology, India in 1977, and a Ph.D. in Computer Science from Tata Institute of Fundamental Research, India in 1981. She is a Senior Computer Scientist at the GE Global Research Center where she has worked since 1989. Her research interests are in Social Networks, Knowledge Management, Natural Language Processing, Semantic Analysis, Optimization and Data Provenance. Recently, she has been leading a project that is developing a system for monitoring and measuring online discussion and understanding how information spreads on the web.

Xiaofeng Wang received B.S. and M.S. degrees in electrical engineering from Tsinghua University, Beijing, China, in 1995 and 1998, respectively, and the Ph.D. degree from the Electrical and Computer Engineering Department of Michigan Technological University, Houghton, in 2001. Currently, he is a System Engineer with GE Energy. His interests include power system modeling, enterprise integration, Smart Grid, and Semantic Web Technology.