Use of the CIM Ontology

Scott Neumann, UISOL
Jay Britton, Areva
Arnold DeVos, Langdale Consultants
Steve Widergren, Pacific Northwest National Laboratory

Abstract

There are many uses for the Common Information Model (CIM), an ontology that is being standardized through Technical Committee 57 of the International Electrotechnical Commission (IEC TC57). The most common uses to date have included application modeling, information exchanges, information management and systems integration. As one should expect, there are many issues that become apparent when the CIM ontology is applied to any one use. Some of these issues are shortcomings within the current draft of the CIM, and others are a consequence of the different ways in which the CIM can be applied using different technologies. As the CIM ontology will and should evolve, there are several dangers that need to be recognized. One is overall consistency and impact upon applications when extending the CIM for a specific need. Another is that a tight coupling of the CIM to specific technologies could limit the value of the CIM in the longer term as an ontology, which becomes a larger issue over time as new technologies emerge.

The integration of systems is one specific area of interest for application of the CIM ontology. This is an area dominated by the use of XML for the definition of messages. While this is certainly true when using Enterprise Application Integration (EAI) products, it is even more true with the movement towards the use of Web Services (WS), Service-Oriented Architectures (SOA) and Enterprise Service Buses (ESB) for integration. This general IT industry trend is consistent with trends seen within the IEC TC57 scope of power system management and associated information exchange. The challenge for TC57 is how to best leverage the CIM ontology using the various XML technologies and standards for integration.

This paper will provide examples of how the CIM ontology is used and describe some specific issues that should be addressed within the CIM in order to increase its usefulness as an ontology. It will also describe some of the issues and challenges that will be faced as the CIM ontology is leveraged for integration purposes by IEC TC57 standards.

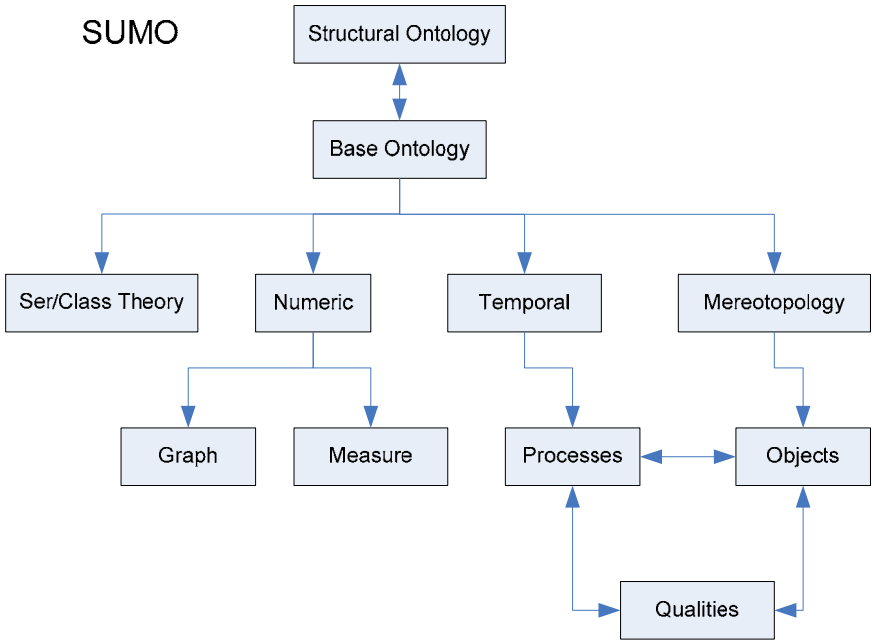
Ontologies

An ontology is an explicit formal specification of how to represent the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them. Another definition that is often used is “a specification of a conceptualization”. Ontologies provide the means to describe knowledge in a form that can be leveraged by both humans and intelligent agents.

There are many ontologies that have been developed or are under development. These are typically categorized as either ‘top-level’ ontologies or ‘domain-specific’ ontologies. There are typically three different levels to the content of an ontology:

1. An ‘is-a’ taxonomy of concepts
2. An internal concept structure and the relationship between concepts
3. Explicit axioms

There are many top-level ontologies that have been developed. Given that the focus of the CIM is for the electric utility industry and the role of the IEEE in the development of standards for the utility industry, a top-level ontology being developed by the IEEE may be appropriate for use in conjunction with the CIM. The Standard Upper Ontology (SUO) working group of the IEEE is defining a standard upper ontology through IEEE P1600.1. The following diagram provides an overview of the SUO.



Ontologies can be formally defined using mechanisms such as the OWL Web Ontology Language. The basis for an ontology could also be provided through other means. This reflects the fact that not all standards or industry groups defining domain models have adopted emerging ontology standards such as OWL and that ontologies have existed

before OWL. It is common for industry standards to be used as the starting point for an ontology. The basis for an ontology could potentially take many forms, including:

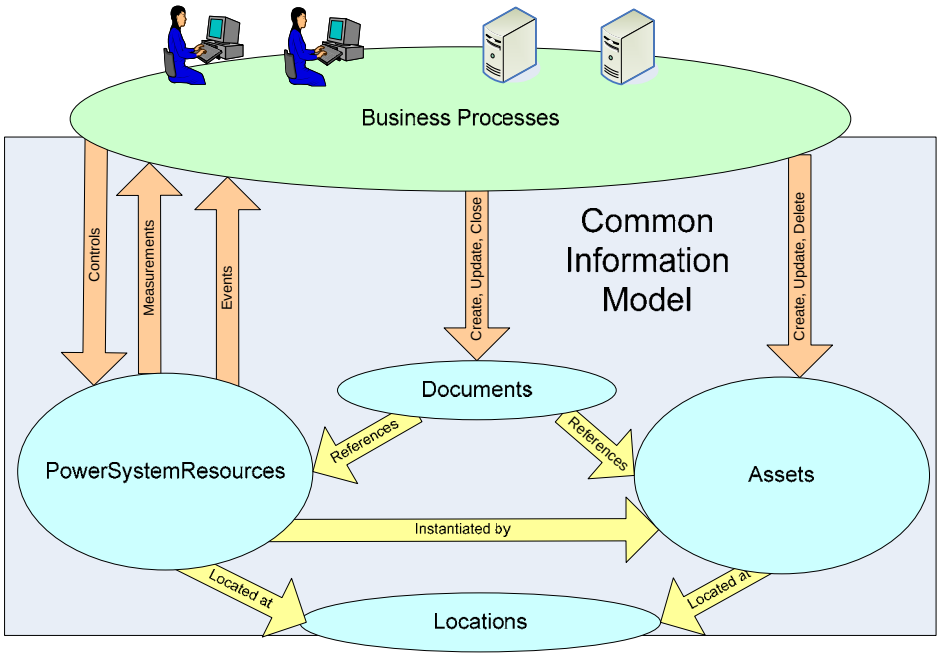
- A controlled vocabulary
- A database schema
- A UML model
- A set of related XML schemas
- An RDF Schema

For example, the class diagram section of a UML model can be expressed as OWL given a few, straight forward assumptions. Tools exist to perform this transformation, taking UML XMI as input and producing OWL RDF/XML output. We would want to do this in order to add mappings to other models or modeling detail that can't be expressed in UML

The CIM as an Ontology

The CIM is a model with coverage for the domain of electric generation, transmission and distribution. In its standard form, the CIM exists as a UML class diagram. However, the CIM can also be formulated as an ontology. There is already a depth of experience with an RDFS formulation of CIM. More recently an OWL formulation has been created. Once formulated as an ontology a number of new possibilities for management and application of the CIM open up.

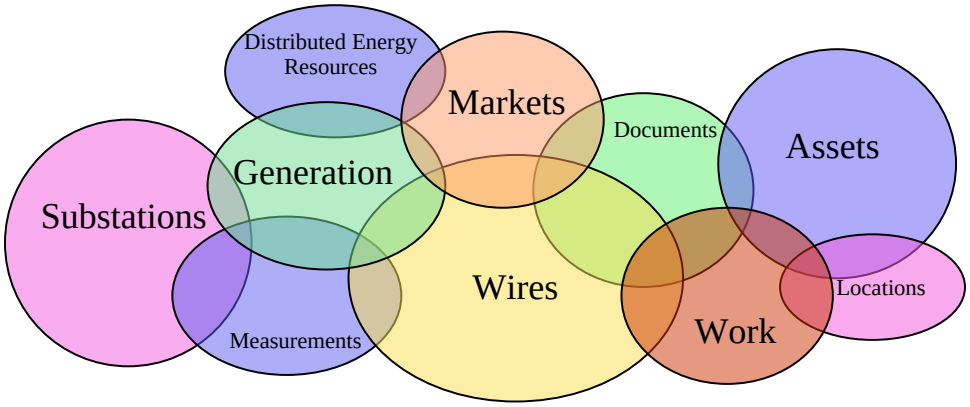
The following diagram provides a simplified overview of the CIM, key relationships and related information flows.



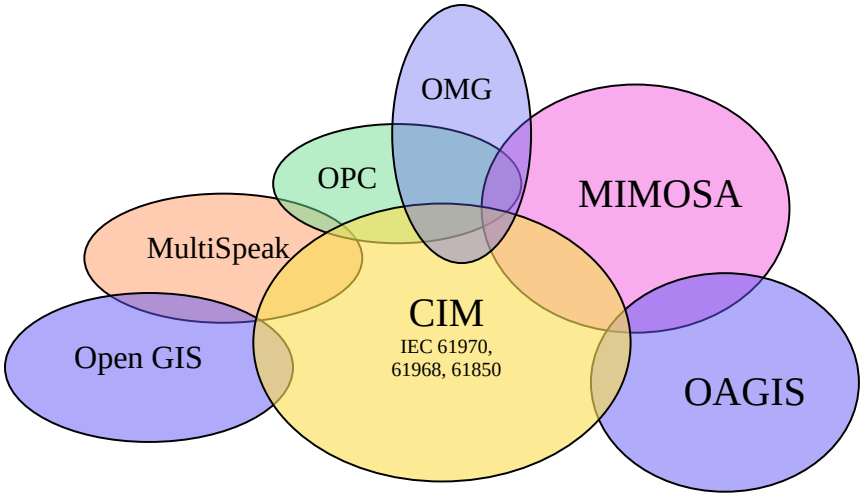
Electrical transmission and distribution networks are modeled in terms of Power System Resources. Power System Resources are logical entities that typically correspond to physical Assets. Business processes may control and monitor power system resources. Central to most business processes is the notion of a ‘document’ as a unit of information exchange, where a document typically contains references to Power System Resources and/or Assets.

The CIM has evolved through several revisions, in both time and space. It has grown beyond a transmission-centric wires model for use by EMS applications to include models needed for distribution, markets, energy scheduling and business processes. The initial focus of intra-application integration has been extended to include intra-enterprise integration and inter-enterprise integration. As the CIM has evolved, there have also been significant advances in the area of integration technologies and architectures.

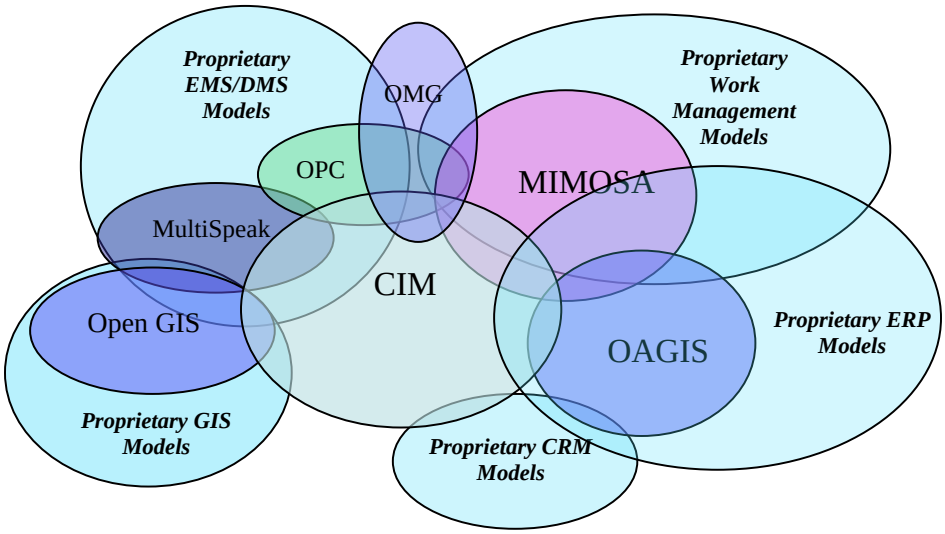
There have been questions as to the appropriate bounds of the CIM. It could be viewed or structured as a federated ontology, where many of the current CIM UML packages form the basis of individual ontologies. The following diagram provides an overview of CIM packages.



The following diagram shows the CIM in relation to other related industry standards, from the perspective of a federation of ontologies. There are cases where other ontologies either support or overlap the CIM, just as industry standards themselves may either be supportive or overlap. For example, the CIM, OPC and MultiSpeak all handle measurements differently. However, it is a relatively simple matter of mapping measurements between the CIM and OPC. OpenGIS and OAGIS are two examples where models have been leveraged by the CIM in a supportive manner.



The following diagram expands upon the previous, showing the general relationships between standards-based ontologies and the ontologies provided by proprietary and standards-based products.



It is important to recognize that many application vendors, and consequentially their customers, will continue to offer products that use proprietary information models.

Technologies for Expressing Ontology

The following sections describe standard languages for expressing ontology. They all use XML syntax, are related to one another, and have strengths and weaknesses in different applications.

RDF and RDF Schema

RDF is a language for representing information about resources (objects and the relationships between them). RDF uses an XML syntax. RDF Schema is a vocabulary for describing properties and classes of RDF resources. Both provide an important starting point when trying to understand technologies for representing ontologies and the CIM as an ontology. RDF and RDF Schema (RDFS) are the basis of current CIM interoperability tests, where the Xpetal tool is used to generate RDFS for the CIM from the Rational Rose MDL files using IEC 61970-501. The following is an example of a description of a breaker class using RDF Schema.

```
<rdf:Description rdf:about="http://iec.ch/TC57/2001/CIM#Breaker">
    <rdfs:subClassOf rdf:resource="http://iec.ch/TC57/2001/CIM#Switch"/>
    <rdfs:isDefinedBy rdf:resource="http://iec.ch/TC57/2001/CIM#Package_Wires"/>
    <rdfs:label xml:lang="en">Breaker</rdfs:label>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
 </rdf:Description>


<rdf:Description rdf:about="http://iec.ch/TC57/2001/CIM#Breaker.ampRating">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:range rdf:resource="http://iec.ch/TC57/2001/CIM-schema-cim10#CurrentFlow"/>
    <rdfs:isDefinedBy rdf:resource="http://iec.ch/TC57/2001/CIM#Package_Wires"/>
    <rdfs:label xml:lang="en">ampRating</rdfs:label>
    <rdfs:comment>Fault interrupting rating in amperes</rdfs:comment>
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
    <rdfs:domain rdf:resource="http://iec.ch/TC57/2001/CIM#Breaker"/>
  </rdf:Description>
```

The following is an example of the definition of a circuit breaker using RDF.

```
<cim:Breaker rdf:ID=”_7">
        <cim:Breaker.normalOpen>false</cim:Breaker.normalOpen>
        <cim:Naming.name>LBS</cim:Naming.name>
        <cim:SubstationComponent.MemberOf_VoltageLevel rdf:resource="#_5"/>
        <cim:ConductingEquipment.Terminals rdf:resource="#_31"/>
        <cim:ConductingEquipment.Terminals rdf:resource="#_32"/>
</cim:Breaker>
```

OWL

OWL is a relatively new W3C recommendation that grew out of RDF Schema, the DARPA Agency Markup Language (DAML) and the Ontology Inference Layer (OIL). Owl adds more vocabulary for describing properties and classes, including relations between classes (e.g. disjointness), cardinality, equality, richer typing of properties, etc. OWL has three sublanguages:

- ***OWL Lite***, which is focused on class hierarchies and simple constraints. The OWL Lite vocabulary is extended beyond RDF Schema to include equivalenceClass, equivalentProperty, sameAs, differentFrom,AllDifferent, inverseOf, TransitiveProperty, SymmetricProperty, FunctionalProperty, InverseFunctionalProperty, allValuesFrom, someValuesFrom. OWL Lite has restrictions on cardinalities and intersections.
- ***OWL DL***, which is focused on maximum expressiveness with guarantees for computability. The OWL DL (and OWL Full) vocabularies are extended with one of, hasValue, disjointWith, unionOf, complementOf, intersectionOf, minCardinality, maxCardinality, cardinality.
- ***OWL Full***, which provides for maximum expressiveness and syntactic freedom without any computational guarantees. In OWL Full, classes can also be treated as instances.

The relationship between these sublanguages is in part that every legal OWL Lite ontology is a legal OWL DL ontology and every legal OWL DL ontology is a legal OWL Full ontology.

XML Schema

XML Schema is used to specify or restrict the structure of XML documents. Most integration tools support or require the use of XML Schema for the definition of messages. Many, integration standards under development or recently adopted use XML Schema. There are some aspects that need to be considered, which would include but not be limited to:

- How simple and complex types will be generated from a model, which might be specified using UML, XMI, RDFS or OWL
- How relationships will be handled
- How extensibility and versioning will be handled
- How the schemas will be packaged for use within message or interface definitions

When generating an XML Schema type definition from a model, there are a few issues and options. One issue is whether or not a given property is required. When generating a complex type definition in an XML Schema from a class defined in UML it is often unclear as to whether a given property is required or optional. The following is an example realization of a complex type for a ‘Breaker’ object, where a Breaker inherits from the class ‘Switch’. Each property is optional by default. This may be appropriate for some applications but not for others.

```
<xs:complexType name="Breaker">
 <xs:extension base="cim:Switch">
  <xs:sequence>
   <xs:element minOccurs="0" name="ampRating" type="cim:CurrentFlow"/>
   <xs:element minOccurs="0" name="inTransitTime" type="cim:Seconds"/>
  </xs:sequence>
 </xs:extension>
</xs:complexType>
```

A related, often encountered issue concerns versions and extensions. Extensions to a model, creating a new version, may involve the creation of,

- new classes
- new subclasses of existing classes
- new properties within an existing class
- new relationships
- new enumerated values

For example,

- The creation of a new subclass of either Switch (or Breaker) would not be readily recognizable as a Switch by an application as the notion of inheritance is 'lost' in the XML instances.
- The addition of a new property to an existing class would cause the validation of the XML to fail against a previous XML Schema even in cases where the 'extra' information is not needed by the application.

In the following rendering of an XML Schema complex type definition, extensibility is provided through the use of the 'any' keyword:

```
<xs:complexType name="Breaker">
 <xs:extension base="cim:Switch">
  <xs:sequence>
   <xs:element minOccurs="0" name="ampRating" type="cim:CurrentFlow"/>
   <xs:element minOccurs="0" name="inTransitTime" type="cim:Seconds"/>
   <xs:any namespace=”##other” minOccurs="0" maxOccurs="unbounded”/>
  </xs:sequence>
 </xs:extension>
</xs:complexType>
```

There are many ways in which the CIM could be realized using XML Schema, with the key differences around the management of associations. With respect to associations, there are three commonly accepted approaches:

- Representation of an association through hierarchical containment, where instances of a child class can be nested within a parent class.
- Use of XLink or URI's for inter and intra-document references
- Use of XML Schema keys and keyrefs

The differences and trade-offs between these approaches are related to the use of industry tools, interoperability, consistency with other standards. Hierarchical containment is the most common approach, and is taken by OAGIS, but does not readily address situations where the object of the relationship may appear more than once in the message.

There are a number of additional issues that need to be addressed within the CIM for the future. The following are examples:

Applications and the CIM

While the CIM is an abstract model, its applications are concrete and use specific technologies. Recent applications of the CIM coincide with the emergence of technologies such as XML, RDF, XML Schema, web services, SOA and ESB. Some of the common ways in which the CIM is used are:

- To define database structures
- To define internal application data structures
- To define message structures (e.g. for communication within an ESB or SOA, often using XML schema)
- To define message bus and application interfaces (e.g. web services in an SOA via WSDL)
- For the encoding of models (e.g. using XML as defined through IEC 61970-501)
- For the encoding of updates to models
- For the encoding of data in request and response messages
- For the encoding of data in event messages

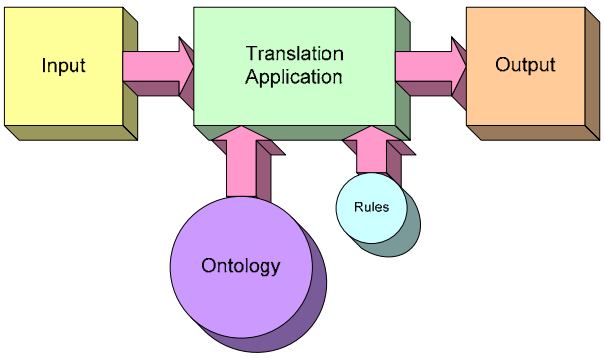
Fragile XML Syndrome

A common problem shared by all of these applications is their sensitivity to change in the CIM or other models on which they are based. The XML is fragile and easily broken (as far as the application is concerned) by a change in the CIM. There is a need for the applications, to which software has been committed, and the CIM, which represents broader concerns, to evolve on their own respective timetables. Ontologies can be exploited here to provide the necessary flexibility. Rather than hard coding application or integration logic, software can be programmed to use configurable rules, providing the ability to adapt to changing models more easily.

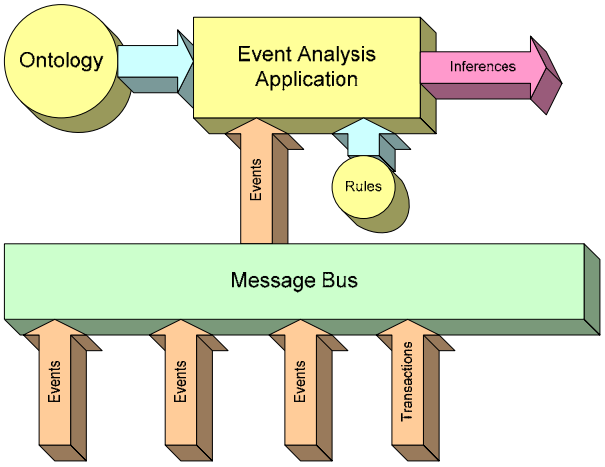
Ontologies for Translation

The following diagram shows a common integration pattern using the CIM (or any ontology). A software component is used to translate incoming data from a given combination of format, model, and content to a desired combination of format, model, and content.

Using a formal ontology and associated tools much of the translation can be driven by rules that are based on the ontology.



The following diagram shows the implementation of another usage pattern, with an application that analyzes events, measurements and transactions received on a message bus from a variety of sources. In most cases, the information will already be present on a message bus (e.g. an ESB). Selected messages are routed to the event analysis application as events. The event analysis application then uses an ontology to interpret the events, in conjunction with a defined set of rules to filter and correlate and draw inferences from them. This is often referred to as 'complex event processing'. An example of a product that addresses this problem space is TIBCO Business Events offering.



Dynamic Adaptation

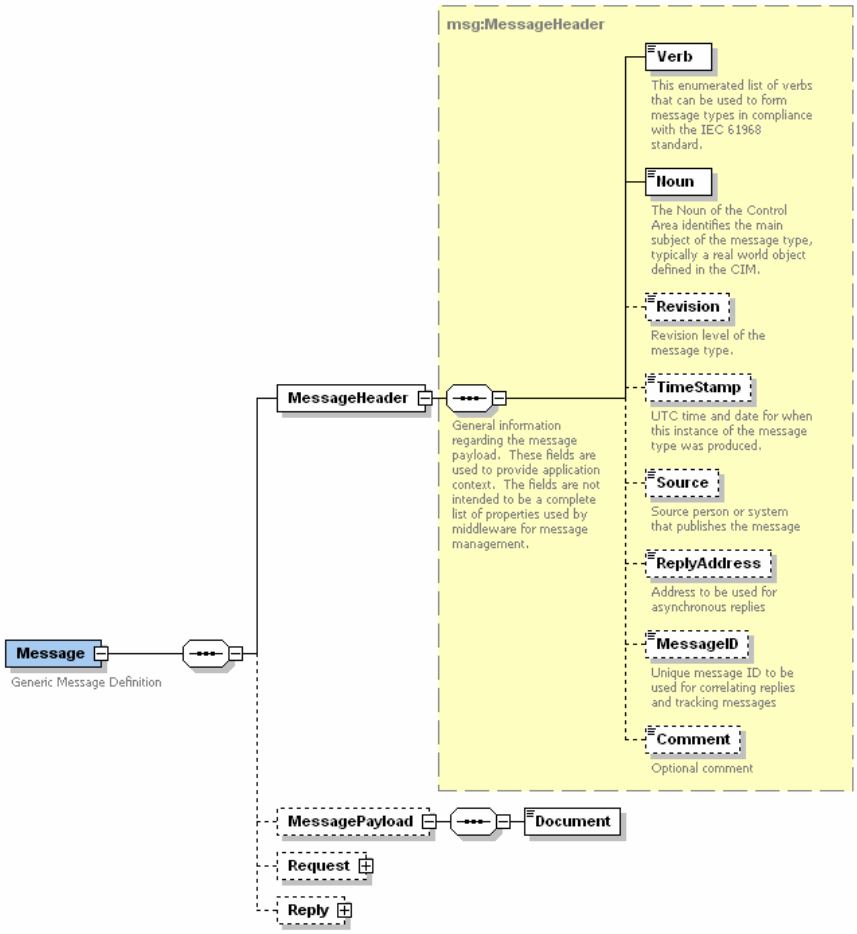
The key point in the foregoing is that the appropriate use of ontologies can enable the development of next generation software components that are flexible with respect to the evolution of models, business processes and functional requirements. Where Model Driven Architecture (MDA) has been used to drive software development from a model, the next step is to provide the means to dynamically adapt at runtime using ontology and rules. This will eventually fulfill the promise of Intelligent Agents to easily adapt to changing requirements and business processes.

Messaging

With the rapid movement towards to use of web services, SOA and ESB technologies, there is the need for TC57 to embrace these technologies and related standards when defining CIM integration standards. We have mentioned some of the issues already. In addition, TC57 should consider:

- Web services profiles for TC57 APIs
- Standard usage of WSDL
- Standard constructs and/or guidelines for message envelopes
- Standard constructs and/or guidelines for support of the various integration patterns, such as request/reply, publish/subscribe, point-to-point, etc.
- Guidelines for secure XML messages
- Specification of profiles for interoperability testing

Currently most messaging leverages XML with structures defined using XML Schema and this is the approach supported by current industry SOA and ESB integration tools. However, the definition of SOA interfaces using WSDL often results in an interface being defined for each type of information to be exchanged. The resulting array of interfaces is multiplied again by the specific actions to be performed with each type of information.

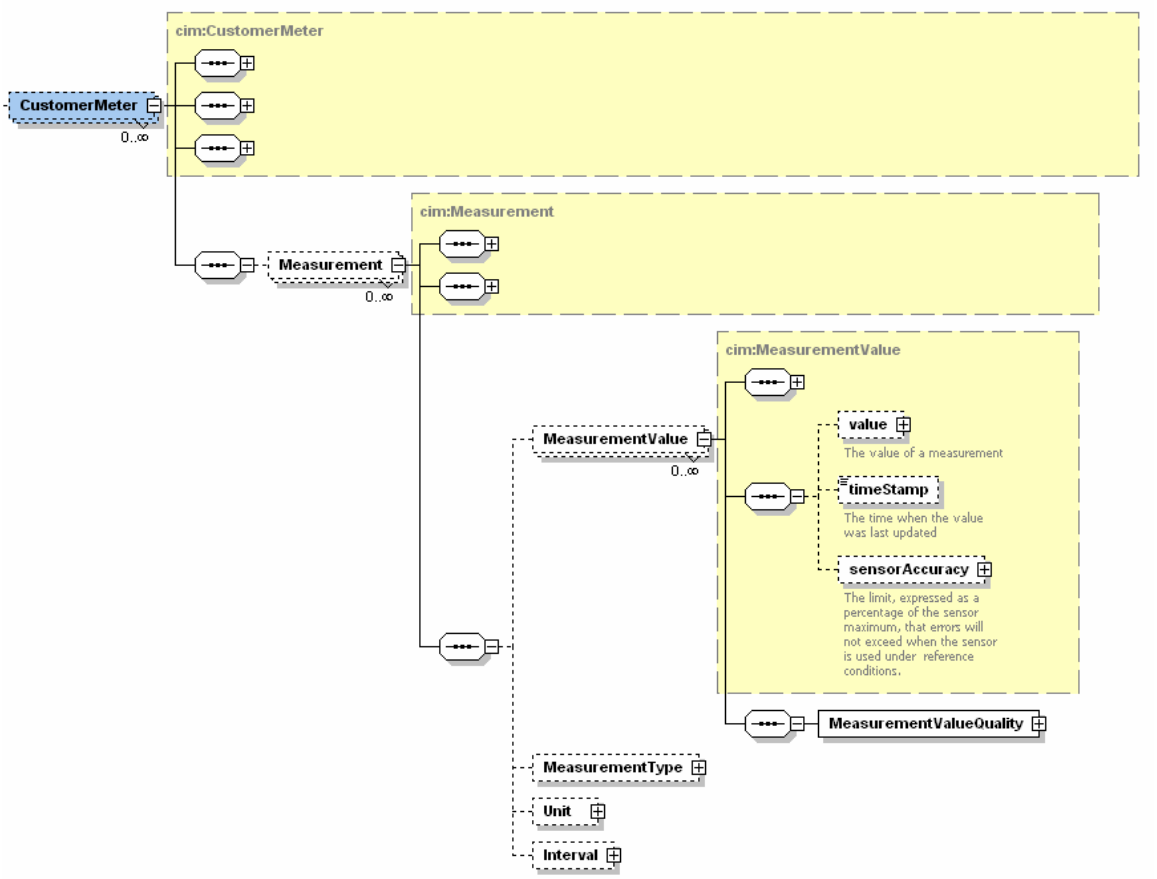


As a partial answer to this issue, the IEC 61968 standard for XML-based information exchange will factor the out separate header and payload message sections. The message header contains a verb, identifying a generic action , and a noun identifying the type of the payload.

The previous diagram (generated from an XML Schema using XML Spy) provides an example implementation of an IEC 61968 message structure, where the message is defined in a generic manner as opposed to being strongly typed to a specific type of payload. There are trade-offs in either approach that need to be weighed, although the approach shown provides significant opportunities for the reduction of the number of interfaces required and the ability to leverage ‘common infrastructure’ within an ESB implementation.

The message payload is defined itself using XML Schema, with a different type definition being used in conjunction with each noun. The message types are typically defined using an XML editor (such as XML Spy or XML Authority), where the content of a message is a combination of complex and simple types generated from the CIM or other ontologies. The following example describes the payload for customer meter readings, where only the classes and relationships of interest are used within the message

definition. This approach avoids bloating the message with relationships that are defined in the CIM, but not relevant to the message.



However, as their use becomes more common, RDF and OWL can be used directly to define content within this overall messaging approach.

Extensions and Versioning

There are two forces for change in the CIM. Specific applications cause the CIM to be augmented with local extensions. Simultaneously, the CIM evolves over time as a consequence of ongoing standardization efforts.
Local extensions would typically be in the form of:

- Locally defined properties
- Locally defined classes
- Locally defined relationships
- Locally defined enumerations and other restrictions
- Inclusion of classes and related properties from ontologies other than the CIM, including those from legacy enterprise models, proprietary software products and related standards efforts
- Inclusion of proposed CIM extensions

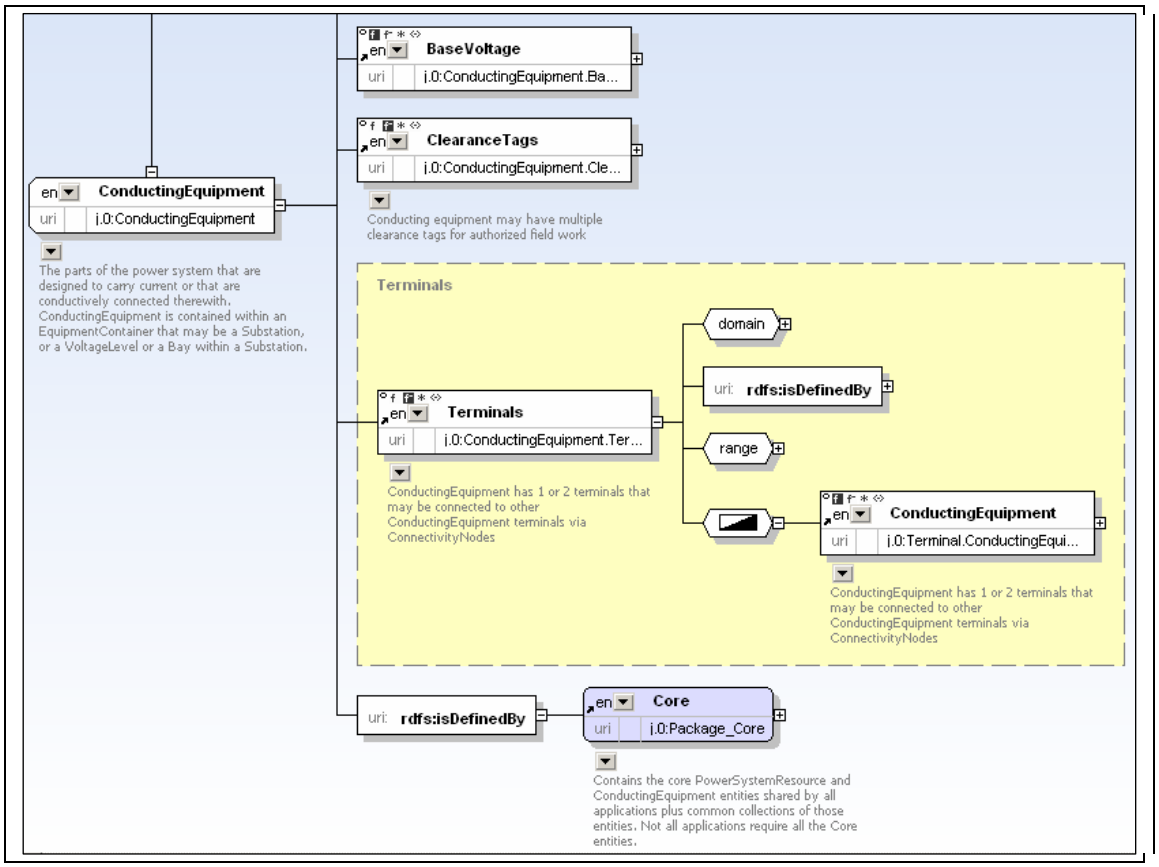
It is important to recognize that where some of the CIM has been formally standardized, other portions of the CIM are a work in progress. In all cases, the CIM will continue to evolve through ongoing standardization efforts, newly identified needs, usage experience and the evolution of relevant industry standards and technologies.

Tools

There are a variety of longer-term issues that arise as a consequence of the evolution of software and integration technologies and related standards over the last few years. Currently the CIM is maintained using Rational Rose, however there is an urgent need to support a broader set of tools. Issues related to tools include:

- The ability to use XML Metadata Interchange (XMI) 2.0 for exchange of CIM UML models, as supported by tools such as MagicDraw, Unicorn and Protege
- The generation and maintenance of CIM RDF Schemas, as supported by tools such as SemanticWorks and Xpetal
- The generation and maintenance of the CIM using OWL, as supported by tools such as SemanticWorks, Unicorn and Protege
- The generation and maintenance of XML Schemas from the CIM, where there is a common form of the resulting simpleTypes and complexTypes
- The definition and maintenance of WSDLs and XML Schema message definitions, using tools such as XML Spy and XML Authority
- The management and access of CIM-derived XML artifacts, as supported by tools such as Systinet and XML Canon
- The merging, mapping and version management of federated ontologies, as supported by tools such as Unicorn

There will be many issues that need to be resolved in order to support such a broad use of the CIM while retaining the ability to evolve and version manage the CIM and mappings to other ontologies. The following diagram shows a portion of the CIM using an OWL-based semantic editor.



Future

Within the IEC TC57, a task force was recently formed to develop a long term roadmap for the CIM. This recognizes that as the CIM will evolve and the evolution and lifecycle should be managed in a similar manner to a software product, with major and minor releases, deprecation and other change management devices. A couple examples of CIM issues that need to be recognized and addressed include the following:

- There needs to be a consistent use of interoperable identifiers. This is a consequence of the CIM:Naming class and the lack of a globally unique ID or mechanism to map between the multiple identities potentially used for an object instance.
- The CIM currently uses only single inheritance, which consequentially forced the creation of parallel classes in cases where a class would otherwise inherit from both PowerSystemResource and Asset. The adoption of an upper level ontology would certainly require the adoption of multiple inheritance.

The longer term aspects to be covered by the task force include:

- The CIM as an ontology in a global federation of ontologies

- Lifecycle of the CIM, with a roadmap for major and minor releases, with initial focus on a roadmap for a future major CIM release
- Use of UML 2.0, XMI 2.0 and tools
- Reconsider the structure of the CIM to divide it into pieces that are easier for humans to maintain, using federation approaches to manage the seams.

From the perspective of an overall CIM-based technical architecture, the following key principles are proposed:

1. CIM be treated as a federated set of ontologies and related XML schemas
2. CIM ontologies are intended to change continuously as new work is done or better ideas emerge in the community at large
3. CIM derived schemas are intended to remain stable and, after development, will pass to the IEC TC57 working groups for standardization
4. CIM technical architecture would be:
    - A core ontology defining the most widely used concepts surrounded by interlinked domain ontologies.
    - XML schemas and other implementation-level specifications in a third, outer tier.
    - Linkages between ontologies consist of equivalences, sub-classes, sub-properties, and property and class restrictions. Linkages between the XML schemas and the ontologies consist of rules, definitions and/or profiles, which are updated as the ontologies change.

References

- "*What are Ontologies and Why Do We Need Them*?", B. Chandrasekaran, John R. Josephson and V. Richard Benjamins, IEEE Intelligent Systems
- "*The State of the Art in Ontology Design*", Natalya Fridman Noy and Carole D. Hafner, American Association for Artificial Intelligence
- "*The Enterprise Ontology*", Dave McComb
- *"OWL Web Ontology Language Overview"*, W3C Recommendation 10 February 2004
- *"OWL Web Ontology Language Guide"*, W3C Recommendation 10 February 2004
- *"Common Information Model"*, IEC 61970-301
- *"Standards for the Integration of Distribution Systems"*, IEC 619768-1
- *"CIM RDF Schema"*, IEC 61970-501
- *"XML for CIM Model Exchange"*, A. DeVos, S.E. Widergren, J. Zhu, IEEE PICA 2001