Managing the Life Cycle of Business Semantics

Xiaofeng Wang, Xtensible Solutions
Steve Van Ausdall, Xtensible Solutions
Joe Zhou, Xtensible Solutions

Introduction

The study of semantics has received much attention recently, largely due to efforts toward building the "Semantic Web". Driving these efforts is the realization that there is a wealth of information that is unavailable to systems and users who need it, because it is locked away in various unstructured and incompatible representations and formats. Semantics aims to unlock this information by building and applying consistent representations that clarify its meaning. In its simplest form, benefits can be achieved simply by documenting the concepts important to the business. A more useful manifestation is the ontology, which not only classifies information; it also ensures that data representations conform to a consistent template. Understanding and managing semantics throughout the information management life cycle provides the foundation for information governance and standards, and ultimately improves interoperability between systems, departments, and organizations.

Much of the expense of integration can be attributed to the time-consuming process of determining exactly what the data in one system represents, and how to represent it in another system. Semantic integration aims to cut those costs by creating a reusable, consistent view of all concepts important to the business, including structural entities, functional processes, business rules, and validation. Semantics provides a formal view of the *relationships* between concepts that clarify their meaning and representation. An example is the specification that "a transformer is a type of electrical equipment; a transformer has a phase". This high-level consistent view of the concepts gives developers a tremendous advantage when building interfaces between systems. Taken a step further, logical and physical models can be built which are correlated with the same concepts and semantics, and are therefore already mapped to other logical and physical models through those relationships. Conversely, additional physical models can be merged into the logical and semantic models to create a more complete, contextual view of the business. The following diagram provides an overview of how these models and concepts relate to each other.

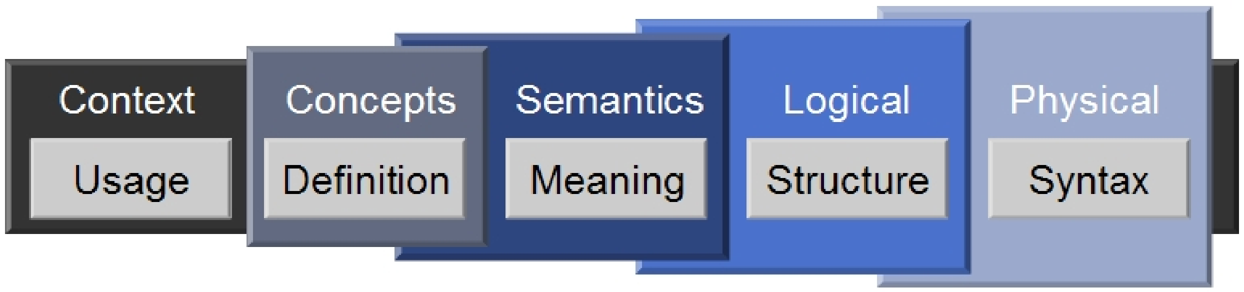

Figure 1: Business Semantics Management Framework

Goals

The goal is to consistently represent business semantics (concepts and meanings) in data models and applications. The solution is a framework to store and manage these models within a business metamodel, and a repeatable methodology that incorporates industry standard models and technologies to generate message schemas and other implementation artifacts. This approach provides improved integration of new and upgraded systems, lowers maintenance costs, and produces reusable interoperable models and artifacts.

Challenges

Business semantics specify, define, and govern the meaning of the terms comprising the domain knowledge, including business concepts, behaviors, structures, relationships, and rules. In a typical electrical utility IT environment, the business semantics can be found in various domain-specific systems including EMS, OMS, GIS, WMS, and SCADA. Unlike the Semantic Web, where the main problem is lack of structure, the core difficulty in an enterprise system integration project is determining and reconciling inconsistent representations. These inconstancies may result from using different terms for the same concepts, using the same term for slightly different concepts, grouping attributes by different concepts, or using different technologies, representations and models to depict the business semantics. Dealing with the inconsistencies, and bridging the gap between the disparate technologies is a great challenge.

Service-Oriented Architecture (SOA) is now the hot topic. Many companies are betting on SOA to solve the integration problem. Though SOA has great potential to decouple integration points, increase interoperability, and decrease integration costs over time, simply using the technology does not guarantee success. Technologies like XML, WSDL, and SOAP may at first appear simple, but the concepts, formats, processes, and dependencies can quickly become unmanageable [4]. Understanding, representing, and using business semantics consistently are the key to success [5].

Benefits

Semantic integration provides advantages and solutions to many problems found in system integration [1]. The concepts and ideas of semantic integration have been steadily gaining ground in the utility industry for many years [2, 3]. Representing business semantics in a set of models and messages so that the business meaning can be communicated unambiguously between applications, systems, processes, and people is one of the major advantages that differentiates the semantic integration approach from others.

A clear view of the business semantics provides many advantages, not just in IT, but to every department in the organization. First and foremost, having a definitive source for each business concept provides a dictionary of terms, with synonyms, so that people **understand** each other better across organizational divisions. But within IT, this information provides **increased productivity** for projects involving integration. Developers can search for conceptual data elements, then browse all structures (relational tables, XML messages, etc.) linked to them, and find information about what values are valid, and what they mean. This is now more important than ever, with development resources changing more frequently, and often having little familiarity with domain concepts, let alone legacy systems and formats. In addition, with standardization comes economic savings. Over time, integration components can be made to be **reusable**, providing an in-house library of common routines and examples from which new endpoints can be built.

Besides the “knowledge management” advantages listed above, the real benefits come once formal, consistent rigor has been used to model and store the semantics, making them machine readable. Having the ability to process semantics provides numerous opportunities for **automated and assisted** modeling and development functionality, such as:

- Improved Search and Navigation (“Smart” Search)
- Semantic Generation (Syllogism) and Validation
- Impact Analysis
- Entity Specification
- Message Building
- Even Mappings and Transformations!

Another benefit of the semantic model-driven integration approach is that it **simplifies management** of integration dependencies. Without a framework to manage the interrelationships between systems, it quickly becomes extremely difficult to account for and test all possible scenarios prior to deployment. Needless to say, finding problems due to unforeseen integration dependencies in a live system is not an enviable situation. By managing semantics and integration models carefully, it is possible to reduce artificial dependencies, and to isolate and control real ones.

In the following sections, we will first propose a framework to manage business semantics, and then explore the business semantics life cycle. Details are discussed, including gathering and creating business semantics, merging industry standard models and existing data models into an enterprise logical canonical integration model, mapping application or external contexts to the canonical model, and deriving implementation models from the logical model.

Business Semantics Management Framework

This section describes the layers of the business modeling framework, depicted in the diagram below. Obviously, this information quickly becomes a large web of elements. Only a few examples have been shown to give an idea of the content in each layer.

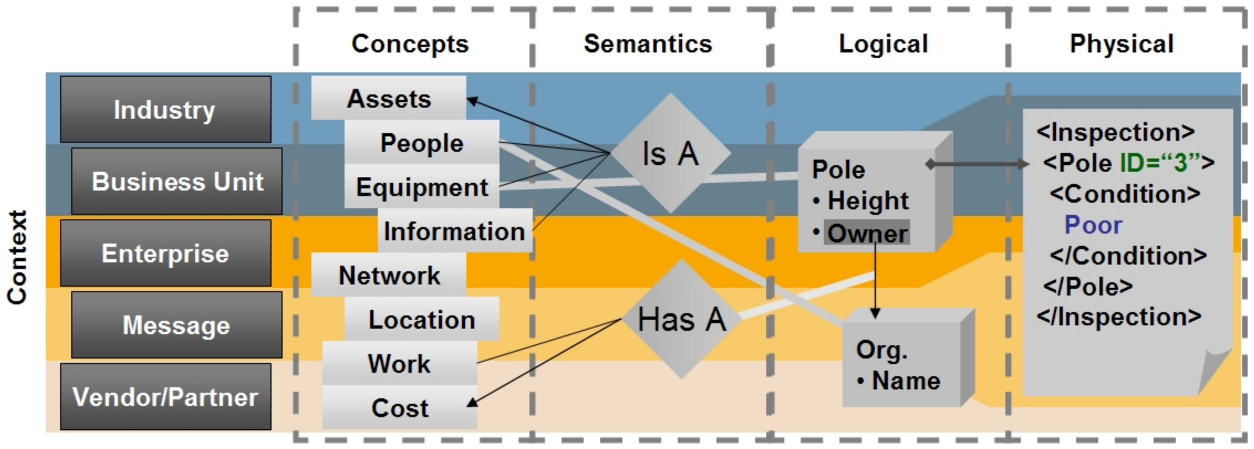

Figure 2: Business Semantic Management Framework

Context
In this framework, the term "context" is used to denote a subset of the overall collection of model elements, within which the terminology and semantics are consistent for one or more domain models, applications, or organizational groups. Ultimately, the goal for improving interoperability is to **find consensus** between groups, so that the important semantics are accurately represented within a single context. Transformations into and out of this enterprise canonical context become the adapters through which systems "plug in" to the information bus that transports data and orchestrates processes between users and systems. By storing all models in one place and maintaining relationships across contexts, it becomes much easier to map representations from one system to another and build translation processes.

Business Concepts and Semantics
The basic goal of the semantic management framework is to build a central repository of concepts and reusable structural models upon which to base integration exchange and other enterprise initiatives. The primary goal of the conceptual model is to reduce ambiguity in terminology, which makes it easier to relate one representation to another. Following is an example of a concept and its definition. Any amount of metadata can also be associated, such as contact, subject area, etc.

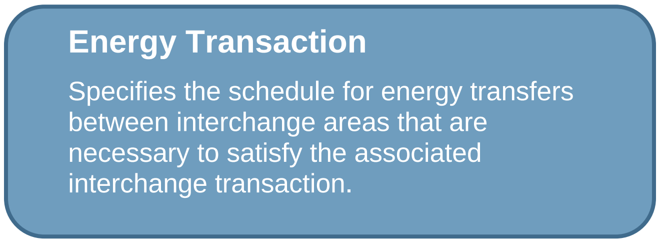
Energy Transaction
Specifies the schedule for energy transfers between interchange areas that are necessary to satisfy the associated interchange transaction.

Figure 3: Sample Concept

The conceptual model can include or be referenced by the semantic network / model, which relates the concepts to each other using relations such as "is a" to denote inheritance or subtype, and "has a" to specify a property. Other relationship types can be defined and used to show ownership, security requirements, dependencies, or any other characteristic important to the business. Following is an example of the semantics of the "Energy Transaction" concept.

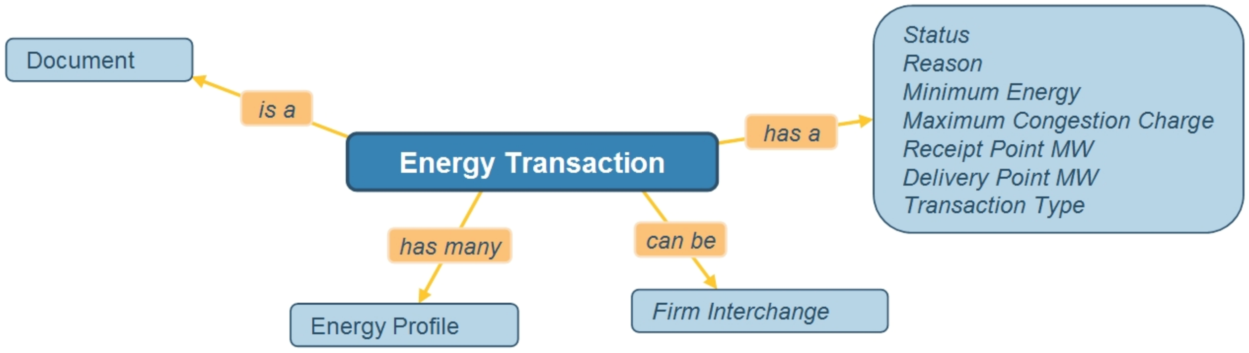

Figure 4: Sample Semantic Relations

Organization
It is important to realize that the semantic model can be as granular, specific, and detailed as one wants to make it, so the key to delivering benefit is to approach the modeling effort from a practical perspective. This is done by first spending just enough time to build and gain consensus on the backbone of the entire conceptual model, with the assistance of key business owners, in order to divide the model into appropriate **subject areas** that can act as anchors to hold the model together. Then, during implementation of a specific integration, the details for new parts of the model are iteratively added.

Some of the subject areas will not correlate to organizational divisions or business processes, since they are fundamental aspects of the business about which many groups need and share information. However, a good starting point is to identify the collections of reference data, and create subject areas for each set managed by a different application. Not all elements will receive the same amount of attention. For example, legal matters are often carefully modeled and tracked, but the information is not often shared between systems.

The initial modeling effort should focus on **master reference data**, the long-living entities involved in and referenced by multiple processes and transactions. Typically, for energy utilities, the subject areas will include the high-level dimensions listed below.

- People and Organizations (e.g. Customers, Workers, Partners)
- Assets and Equipment (e.g. Generating Unit, Meter, Vehicle, Database)
- Network Topology (e.g. Segment, Circuit)
- Geography and Locations (e.g. Boundary, Water Body, Transportation Route)
- Time (e.g. Reporting Period, Schedule Interval)
- Documents (e.g. Agreements, Standards, Procedures)

These subject areas are then tied together with **functional and transactional** concepts, that track what is planned to happen, what actually happened, when it happened, and what equipment or people were involved. Ultimately, managers get a much clearer view of how resources are being used and where improvements are necessary.

- Project and Work Management (e.g. Work Order)
- Logistics and Scheduling (e.g. Project Plan, Energy Schedule)
- Supply Chain and Procurement (e.g. Purchase Order)
- Financial Accounting (e.g. Invoice, Budget, Project)

Because these conceptual entities are used across a number of business processes, one of the most important pieces of metadata that can be gathered during the initial stages is the list of groups or processes that have a stake in each conceptual area or entity, and which would be most impacted by changing representations or processes. In addition, the governance process may require that a single individual is accountable for managing each part of the model, and is authorized to approve changes to the model after technical impact analysis has been performed.

In the first stage of development, the business concepts and semantics are defined using free text, diagrams, mathematical definitions, and other unstructured information. The business semantics are initially intended to provide the business meaning to be understood by humans. The conceptual model is useful for organizing and governing data entities, as well as building consensus on what terminology should be used, as well as common synonyms. Dictionary entries can be used directly in system help, especially in analysis and reporting applications to provide short-term benefits to the business.

Logical Model
In order to be understood by applications and systems, the business semantics are given shape using a platform independent modeling (PIM) language such as UML or IDEF, resulting in a logical model. These logical structural and process models, also organized by subject area and context, are used as the basis for building application, information, and service interfaces between systems and organizations. They can also be utilized to generate or build system database structures, taxonomies and ontologies, code, reports and documentation.

Building the enterprise logical model requires careful consideration of modeling constructs such as normalization, granularity, abstraction, inheritance, composition, and association. The model is organized so as to implement the data requirements in an efficient and manageable way.

An example is shown below of a portion of the IEC TC57 Common Information Model (CIM) for Energy Utilities. This model was used to build the message schema shown in the next section.

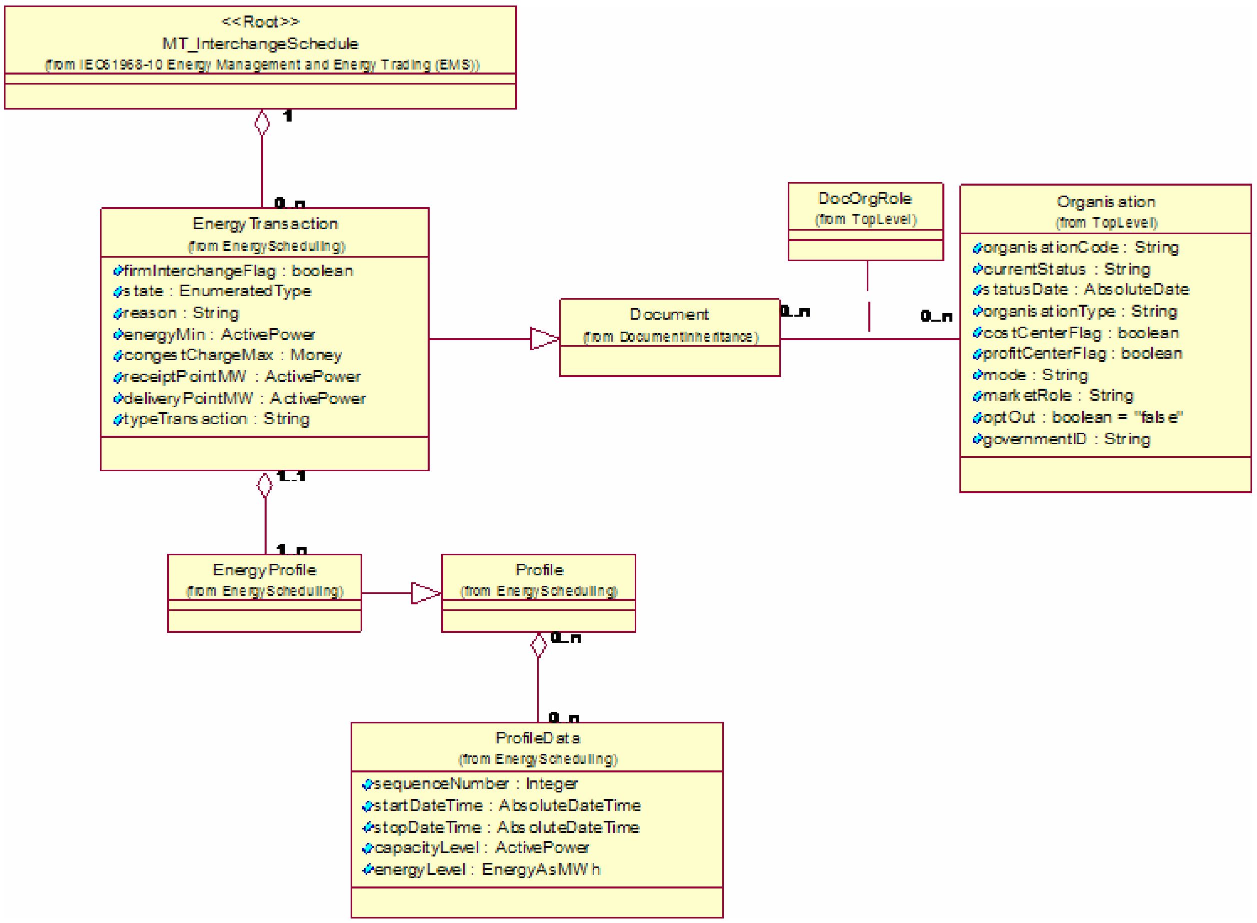

Figure 5: CIM Logical Model in UML Notation

The semantics of a fairly abstract logical model like the CIM are spread across many layers within the model, so that attribute concepts that are shared by a large number of “leaf” entities are typically held within an ancestor class from which all appropriate child entities inherit. This maximizes reuse within the model, but is often confusing to business users and even to developers. For example, breaker status (e.g. open / closed) is not an explicit attribute of the breaker class. Instead, it is held within another section of the model that allows measurements and other operational properties to be associated with the appropriate network entities. Managing the semantic model explicitly, independent of the logical models aids in conveying the business concepts without confusing them with a specific structure. It also provides stability, because process models can be loosely coupled to logical structures using the layer of indirection that the separate semantic model provides.

Options

It is possible to construct logical models without a separate conceptual or semantic view. In this case, the concepts and semantics are specific to each logical view, and it is more difficult to reuse those layers in other contexts. It is also possible to build the conceptual / semantic model independent of the logical models, but the lack of those relationships across domains and contexts reduces opportunity for reuse, impact analysis and other assisted or automated functions.

It is also worth noting that each logical model actually contains concepts and semantics, so it is possible to generate semantics from a logical model. However, the benefits of conceptual and semantic consistency can only be achieved by linking multiple logical model contexts to a common semantic view. The ultimate framework links each logical model context to the concepts and semantics that it implements, providing the most opportunity for consistency and reuse, and enabling the creation of automated and/or assisted mapping and validation functionality.

Physical Model

For each implementation technology target (such as web services or relational databases), platform specific models (PSM), such as physical schemas, are defined (using data definition languages such as XSD and SQL), which are based on, and hopefully generated from, the desired logical model context. For system integration purposes, message contexts are used, which are derived from the enterprise logical canonical model, but which contain only the information to be passed between systems. This process can be performed manually, but given the proper framework organization and tools, many of the steps can be automated, in order to allow semantic changes to be cascaded into each representation that uses those changed elements. A sample physical model (XSD) is shown below.

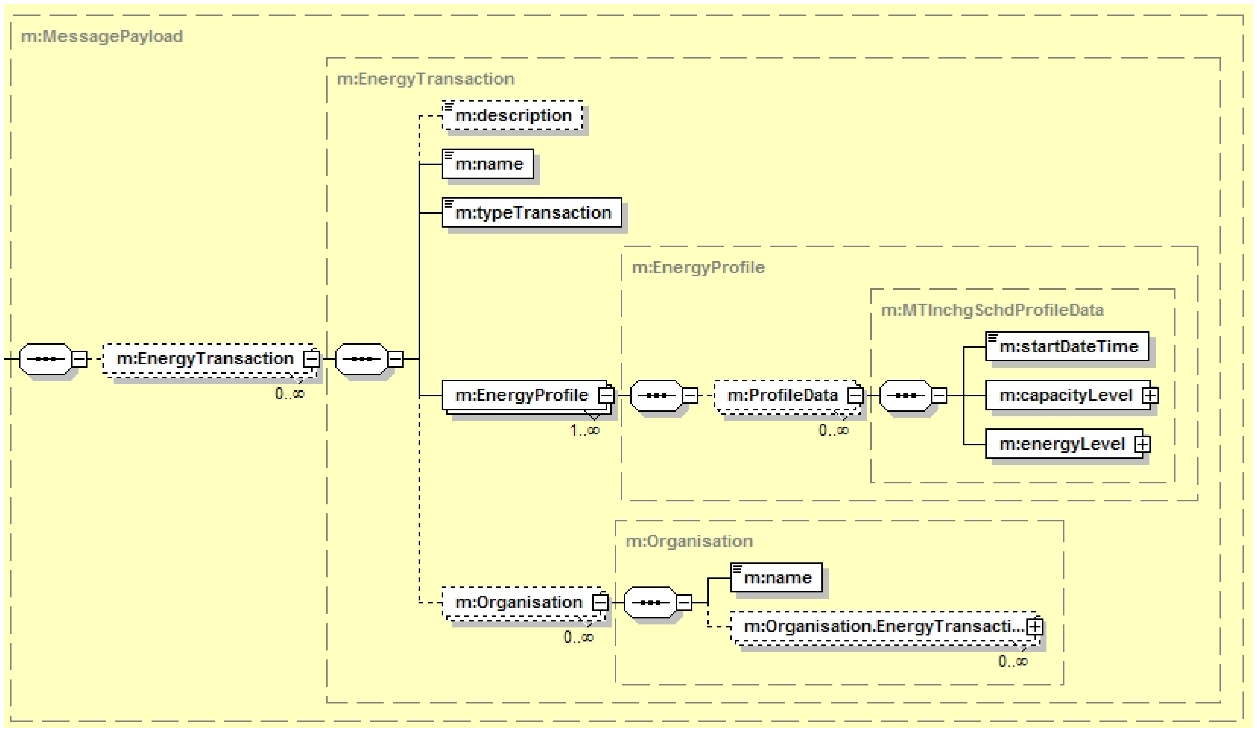

Figure 6: Sample XML Schema Physical Model

The physical model design process maps the more general logical modeling concepts to concrete physical modeling concepts, and then configures the physical model representation to address performance concerns, referential integrity, optionality and cardinality, restrictions of data types, and other implementation specific issues.

The framework shall address the following concerns:

- **Physical Model Naming and Design Rules** – The NDR is a specification that defines the general principles and transformation rules of deriving the physical model from the logical model. Based on the NDR, all implementation models will be derived consistently and predictably. This specification first guarantees that the business semantics can be conveyed from logical model to physical model unambiguously. Second, because the physical model is predictable and consistent, it is easy to set up code generation rules. These two advantages will let the business semantics flow correctly from the logical model all the way down to the code level.
- **Model Configuration Management** – Based on the NDR, the framework shall allow users to configure the physical model structure and representation. It also needs to provide users capability to address implementation specific issues. The configuration management needs to be a repeatable process so that users can use generate the final physical model correctly.
- **Automatic Testing** – Automatic testing provides functions to prove if the derived physical model validates against all utilized schemas (such as XML Schema, WSDL, etc.), conforms to the NDR, meets the performance requirements, and satisfies the data integrity checks.

General Considerations

The overarching design goal of the modeling framework is to separate the models into layers that maximize reusability, and to provide views appropriate for different roles and usage patterns. The conceptual / semantic layer is appropriate for business users. Power users and subject matter experts (SMEs) should not only be able to view this layer, but should also be able to contribute to the refinement of it. Logical models are in the next layer, allowing designers and architects to implement the semantics consistently in a technology independent way. Developers then use those models to build platform specific schemas. The following diagram provides a high level view of the business information metamodel by role, containing the three layers discussed.

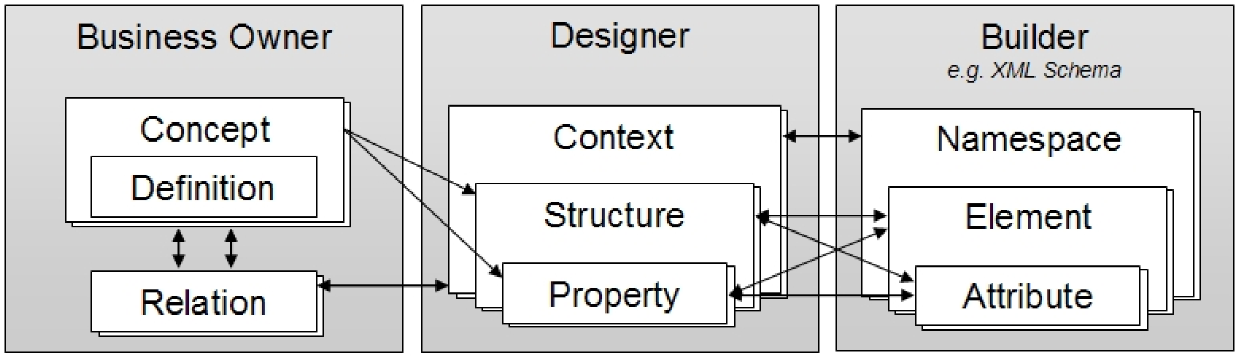

Figure 7: Business Metamodel Organization Layers

During the modeling process, the framework shall support:

- **Iterative Modeling –** It typically takes several modeling iterations to sufficiently capture all concepts and details required by the business. In the early stages, a high level model is built to capture the major concepts and serves as a starting point [6]. With each project iteration, more accurate models will augment and/or replace the early models.
- **Model Consolidation –** Building these models from scratch may not be practical given the complexity of the utility business domains. Leveraging industry standards such as CIM, GML and other existing models may dramatically reduce the modeling time and increase interoperability. Because these existing models are designed with different styles and modeling technologies, the framework shall first provide the capabilities to consolidate the differences and build a unified model which can be further crafted to serve as an enterprise logical model. Some of the capabilities required are listed below.
    - **Import –** The framework must be able to read models in various formats into the modeling environment. Common formats include XMI, XSD, and SQL. It must be possible to reverse engineer, include and modify these model elements within other contexts.
    - **Reference –** It must also be possible to build a new model out of elements referenced from imported models. Links to source models should be maintained in order to allow acceptance of external changes.
    - **Extension –** Extensions are useful when adding additional capabilities to a model that not all consumers or contexts will use. For many model consolidations, it is possible to stitch two domains together by extending a set of common elements with imported or referenced entities and associations.
    - **Overriding –** This capability allows data types or other model configurations to be changed within a specific context. The change may be to make an element

optional, or to change an enumerated list of possible values, or even to remove an attribute or element.

- **Collaborative Modeling –** The framework shall provide the capability for concurrent update access by modelers, analysts, and business owners. This may be implemented by partitioning the model by subject area and allowing users to lock one section for changes while others can lock and update other sections, or it may provide finer-grained control by allowing users to make changes in a workspace and post those changes as a whole when finished. Conflict detection and resolution may be necessary to ensure consistency and integrity of the models.
- **Software Configuration Management –** This includes controlling, tracking, and managing changes, as well as grouping them into releases.

Context Usage

Use of context to combine and refine models for a specific situation is a fairly new concept in modeling. In this section, several types of contexts are presented, with discussion around how they relate to each other and can be used toward creating a semantically consistent integration framework and platform.

The overall approach of combining and refining reference models is shown below.

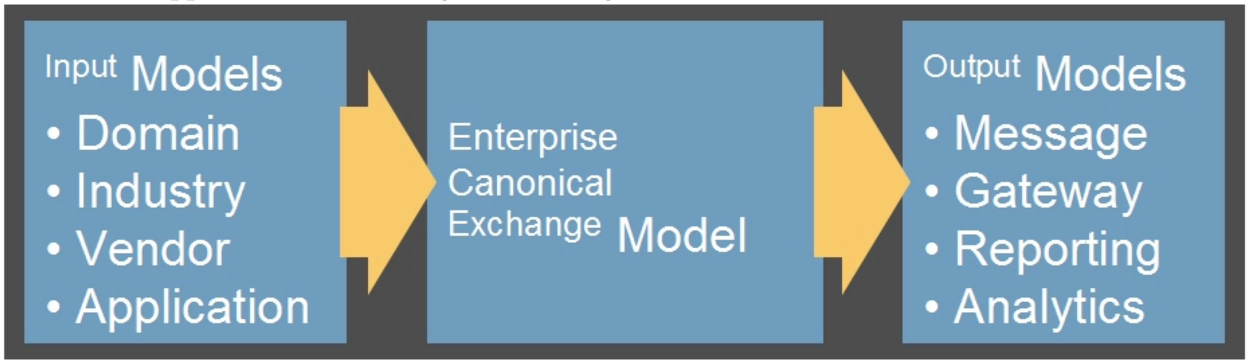

Figure 8: Relationship between Contexts

Ensuring consistency of the business semantics requires identifying and collecting all models and semantics currently in use, and then consolidating and combining them into a single, consistent model from which enterprise models can be derived. Using this approach, the mapping into and out of the enterprise context is done once for each input (endpoint) context, and is then maintained and reused for all applications, including integration (messages, gateway interfaces, etc.), business intelligence (reporting and analytics), as well as other enterprise initiatives such as business process and performance management, master data management, and others.

The following diagram shows an example of this process, for energy industry standard models. Input models at the top are combined, filtered and refined through the modeling process to construct the enterprise canonical model context. The semantics, shown along the left side, are linked to each model to ensure consistency, and to maintain the mapping back to a single cohesive set of concepts, through which transformations can be derived.

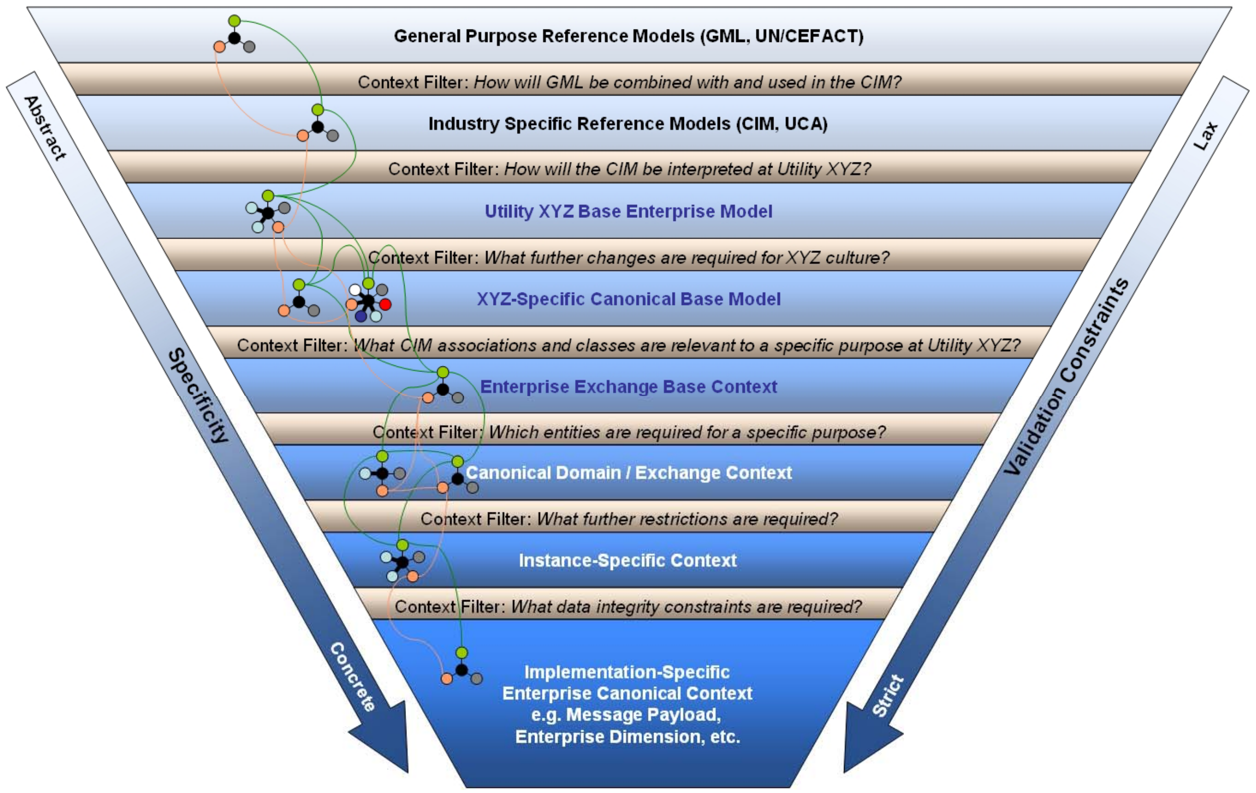
General Purpose Reference Models (GML, UN/CEFACT)
Context Filter: *How will GML be combined with and used in the CIM?*
Industry Specific Reference Models (CIM, UCA)
Context Filter: *How will the CIM be interpreted at Utility XYZ?*
Utility XYZ Base Enterprise Model
Context Filter: *What further changes are required for XYZ culture?*
XYZ-Specific Canonical Base Model
Context Filter: *What CIM associations and classes are relevant to a specific purpose at Utility XYZ?*
Enterprise Exchange Base Context
Context Filter: *Which entities are required for a specific purpose?*
Canonical Domain / Exchange Context
Context Filter: *What further restrictions are required?*
Instance-Specific Context
Context Filter: *What data integrity constraints are required?*
Implementation-Specific Enterprise Canonical Context e.g. Message Payload, Enterprise Dimension, etc.
Abstract
Specificity
Concrete
Lax
Validation Constraints
Strict

Figure 9: Enterprise Canonical Context Construction

These layers are just examples, meant to illustrate how contexts might be refined through a multi-step process in order to arrive at the final model for a particular purpose. The refinements would likely be grouped into fewer context layers, and are only shown individually to illustrate the refinement process. The changes within each context will ultimately be specified as extensions, overrides, and other transformational adaptations from a parent representation, instead of static views. This way, changes can be grouped and maintained at the appropriate level within the model so that they can be applied consistently to lower levels.

The Life Cycle of Business Semantics

In this section, we lay out the overall picture of the business semantic life cycle. It helps us understand how business semantics are created, captured, maintained, and used.

A business system can be summarized as interactions between humans and applications. Business semantics are represented, transferred, and interpreted throughout out the business system by humans and applications. The primary semantic flows are categorized in reference [1] as human to human (H2H), human to application (H2A), application to human (A2H), and application to application (A2A). The business semantics originate from the humans and applications that produce exchange messages. In an ideal situation, the business meanings would be carried by exchange messages, and delivered to human and application message receivers, who would correctly interpret the business meanings contained within the exchange messages. There are four important phases in the life cycle of business semantics, as depicted in the following diagram.

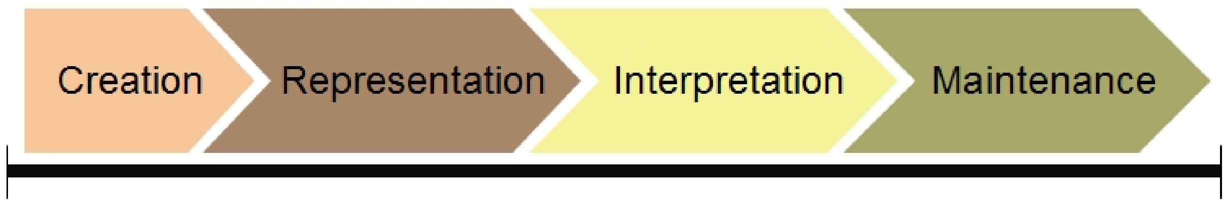

Figure 10: Lifecycle of Business Semantics

Creation of Business Semantics

Because the business semantics originate from humans and applications, business requirements are the primary source from which the business meanings should be discovered and captured. The discovery and creation of business semantics includes:

- **Requirements Gathering** – This process gathers business requirements from the subject matter experts and business analysts familiar with the business processes in a specific subject area. Data dictionary and other meta-data information (including existing models), system requirements and designs, and business processes are all useful for discovering and verifying semantics [1].
- **Discovering Business Meanings –** Based on the requirements collected in the previous step, this process uncovers the business meanings which are represented with terms, free text, diagrams, mathematical equations, and other unstructured information. These informal representations of semantics are the first step towards making them useful to machines.
- **Creation of Business Semantics –** After business meanings are discovered from various sources, enterprise-wide business semantics are created to specify relationships between concepts, constraints, and additional specifics. This process refines and consolidates the business meanings from different sources and creates standard, approved terminologies, vocabularies, data dictionaries, descriptions, and other information to represent the business semantics such that it can be agreed to and understood by humans across organizational boundaries.

Representation of Business Semantics

Business semantics play a key role in optimizing interactions between humans and applications. But understanding by humans is only one piece of the puzzle. Business semantics also need to be understood and commonly represented by applications. In order to make the business semantics understandable by machines, a modeling language is used to express the terms, descriptions, and relationships that unambiguously specify and convey the business meanings. This modeling process creates a representation of the business semantics with various modeling components and concepts which are used for code generation, validation check, mapping, and other functionality.

A wide variety of technologies can be used to represent the business semantics. The different approaches are listed in the following table.

Technology / Approach	Design Considerations
Free Text	Requires least modeling expertise and rigor, but provides the least opportunity for benefit. Does not allow machines to interpret or process the semantics.
Semantic Model (e.g. RDF, OWL)	These technologies associate concepts using a named relation such as “is A” (to denote inheritance), and “has A” (to denote structures such as properties and associations). Different tools support different relations, and new relation types can be defined and used. These technologies present the most risk, due to complexity and immaturity, but they also provide the greatest potential reward.
Logical Model (e.g. UML, IDEF)	Logical modeling technologies have become widespread, and they are used most often to represent business semantics. By deriving physical models from an enterprise canonical model, semantics can be preserved and reused across integrations. However, advanced functionality such as automated mapping will not be supported.
Physical Model (e.g. XSD, SQL)	Using this approach means there is no representation of the business semantics except that which is implied in the physical models. Though physical models do imply semantics, using this layer as the only representation provides the least ability to provide functionality and manage consistency between models.

Interpretation and Use of Business Semantics
Up to this point, we have explained the processes involved in creating and representing business semantics. We have created business semantics with the terms, vocabularies, descriptions, and other unstructured information to help human to understand. We have created logical/physical model to convey the business meanings. And we know that the ultimate goal is to have business meanings communicated and interpreted unambiguously by humans and applications. This section discusses some approaches that can be used to solidify the semantics within applications, developers, and business users.

Publication, consensus building, and training are required to provide a **visible repository** of the semantics in their various states of refinement and acceptance. Processes and rules should be well defined, so that the changes can be carefully controlled but highly collaborative. As concepts and semantics become finalized, they can be referred to and used in knowledge portals, help text, reporting applications, and other destinations.

In order to fully convey and maintain semantic consistency, and in order to fully utilize and benefit from the semantic model, **mappings** are required, to relate the new concepts and structures to those business users and IT developers are familiar with. Mappings between the semantic model and the logical model help ensure that the enterprise logical canonical model captures every aspect of business semantics. The semantic model can be used to assist in mapping domain and reference logical models (endpoints, messages, and industry standard or other external models) to the canonical model, and to each other through the canonical model.

These mappings specify the high-level transformation between system models, including structures, properties, and even values.

Maintenance of Business Semantics

If there is one constant in business, it is change. Inevitably, changes will be required to extend the model to augment existing entities, encompass additional entities, improve performance, or more accurately represent the business. There are a number of strategies that can be employed to control and minimize impacts of changes.

- **Governance** – The methodology assumes that all software changes are approved by an authorized owner or group, are then targeted for a specific deployment release, and are then specified and implemented. The process should be well defined and continually improved, so that coordination and control can be achieved to reasonably prevent expensive production problems, project rework, and cancellations caused by false assumptions, poor communication, or unmet dependencies; and so that quality, security, and compliance can be continually increased.
- **Extensions –** Once consumers have started accepting messages of a certain published format, it is best not to change the published format such that those consumers must update their adapters. This backwards compatibility is accomplished by only adding optional elements (extensions) to existing formats. Changes to existing elements, or removal of elements that would break existing consumers should be avoided.
- **Transformations** – Many representations can be easily adapted to another representation using a transformation within the orchestration layer that can provide a way for existing services to communicate with new services.
- **Artifacts –** One or more documents should be produced and approved as a result of the modeling process, including requirements, issues lists, service and interface specifications, model change log, and release notes, in addition to code artifacts.

Once integration interfaces have been built, changes to the model must be carefully controlled. This is one area where the hard work of building the model framework really pays off. Because the existing configuration is stored in a traceable repository, impact analysis can be performed to find systems using the impacted data elements.

Best Practices

This section provides some tactics for mitigating the risks that commonly cause problems.

Start Simple	It is easy to bite off more than you can chew. Start with simple tasks involving business owners, such as a conceptual dictionary, and build slowly.
Manage the Culture Change	Enterprise integration is a paradigm shift that requires cultural change as well as technology change. Often the culture shift is the more difficult hurdle to clear. Using the model to force change too broadly or quickly into the IT environment is usually met with resistance, and may ultimately result in the model's failure. Include training to explain the approach, slowly build the model using short iterations, and provide short-term benefits to all departments if possible.

Keep it Simple	It is easy to create a model and framework that is more complex than it needs to be. At every turn, try to use the simplest approach that will provide good benefit. Leave automation out in the early stages, until it becomes necessary.
Commitment	Managing an enterprise model is not easy, and is not a “project” activity, but a long-term strategy. The organization must have the resources, experience and commitment to build and maintain the model.
Leverage Investments	The enterprise modelling activity must leverage historical investments in existing models and metadata as well as existing resources and training.
Coordinate with Projects	Especially if usage of the enterprise model is mandated for new integrations, the modelling team must have enough resources to quickly respond to projects, to make sure that projects are not waiting for updates to or outputs from the model.

Conclusion

Enterprise modeling can be a daunting undertaking. A typical utility may have hundreds or thousands of data entities, utilized by numerous applications and databases. However, automating the flow of information between systems, and improving data accessibility, quality, and integrity can provide huge cost savings. Information management and integration have moved to the forefront of system engineering. This paper has explored one aspect of information management, the management of business semantics within information models. While this field is still young, tools are rapidly emerging that will enable data modelers and enterprise architects to create and maintain a consistent, reusable view of semantics and information. Once this model repository is built, the information can be used by every department to better understand application data, and to build standard integration interfaces, analysis, and reporting applications.

Glossary

A glossary of the key terms in this paper is included below.

Term	Definition
Concept	Concepts represent ideas and objects important in a system. Concepts are the top of the metamodel inheritance hierarchy, and are separate from the Terms (i.e. Data Entities, for structural models) used to refer to them. Concepts, Terms, and Contexts form a triangle, to which all other model elements are associated.
Conceptual Model	The conceptual model is a view of the concepts without relationships, but with textual descriptions. It is the least formal representation of information, but is useful in providing a common ground to which more explicit representations are related.
Semantic Model	The semantic model adds meaning to the conceptual model by building a directed graph connecting concepts together using semantic relationships such as "is a" and "has a". It is used to represent assertions about the concepts without regard for a specific structure. These facts can then be used for multiple purposes.
Logical Model	The logical model provides structure for the concepts, in an implementation independent language, such as UML or IDEF.
Structure	Structure is an abstract term for a class or table, in which each "record" (row) contains (a subset of) data fields (columns) defined to be valid.

Term	Definition
Physical Model	The physical model, or schema, is a representation of the logical model data design which uses the specific syntax of a given implementation platform, such as an RDBMS, format (i.e. XSD) or programming language.
Context	A context is a container denoting a subset of model elements and configurations for which the semantics and representations are consistent and applicable for a specific business scenario or domain. For example, depending on the country context, the address entity must have slightly different representations.
Context Model	The context model is an inheritance hierarchy of contexts, depicting the organization of classification schemes, such as domain ownership or security.
Enterprise Canonical Model	This phrase refers to a context used to derive models to be used to exchange data between systems in integration interfaces, or to represent concepts in enterprise reporting applications. It could contain views into any of the model layers, including conceptual, semantic, logical, and physical. It may contain multiple contexts, representing refinements to a base model, which may include elements from different contexts.

References

[1] McComb, Dave, Semantics in Business Systems – The Savvy Manager's Guide, Morgan Kaufmann Publishers, 2004.

[2] Newberry, Ken and Greg Robinson, "The Semantic Integration Framework at TVA," GITA 2004, Conference Proceedings.

[3] Eric Lambert and Greg Robinson, "Building the Business Case for Addressing Semantics in Application Integration," DistribuTech 2005, Conference Proceedings.

[4] Brain Leonard and Greg Robinson, "ACHIEVING RELIABILITY OBJECTIVES WITH THE AID OF A SERVICE ORIENTED ARCHITECTURE," DistribuTech 2005, Conference Proceedings.

[5] David S. Linthicum, "10 Best Practices for Creating a Service-Oriented Architecture," Business Integration Journal, October 2005, PP. 24-27