# **Ontological Approaches for Semantic Interoperability**

Michael A Zang Technical Leader CDM Technologies, Inc. <u>mzang@cdmtech.com</u>

#### Abstract

This paper provides a basic description of the concept of an ontology. It then describes how ontologies are structured and employed in the context of interfaces between software based information systems. This usage is discussed in the context of three successive levels of semantic interoperability between two example systems. The paper goes on to suggest that the interfaces between information systems should perhaps be viewed and implemented as systems themselves. The paper concludes by providing a brief summary of what was discussed.

## Keywords

Agent, Data, Information, Interoperability, Knowledge, Object Model, Ontology, Semantics

## Introduction

An ontology is an explicit specification of a conceptualization. The term is borrowed from philosophy, where an ontology is a systematic account of existence. For a software application, what "exists" is that which can be represented. When the information and knowledge of a domain is represented in a declarative formalism, the set of objects that can be represented is called the universe of discourse. This set of objects, and the describable relationships among them represents all the information and knowledge that can be known in the context of the applications that employ them. In such an ontology, definitions associate the names of entities in the universe of discourse (e.g., classes, relations, functions, or other objects) with human-readable text describing what the names mean, and formal axioms that constrain the interpretation and well-formed use of these terms.

In terms of semantic interoperability, an ontology defines the vocabulary with which queries and assertions are exchanged among applications. Ontological commitments are agreements to use the shared vocabulary in a coherent and consistent manner. The applications sharing a vocabulary need not share a knowledge base; each knows things the other does not, and an application that commits to an ontology is not required to answer all queries that can be formulated in the shared vocabulary.

An Interface Domain Ontology is an ontology specifically geared towards interfacing multiple domain specific software systems. The concepts in an Interface Domain Ontology can be organized in a hierarchical structure of three layers as shown in Figure 1. In the Upper Level Ontology are generic concepts, such as 'process', 'agent', 'set', 'proposition', and 'goal'. In the Lower Level Ontology are the elementary concepts, such

as 'SSN', 'NEC', 'street number', 'cost' and 'internet Address'. Generally, for two cooperating partners, it is relatively easy to reach a consensus on the concepts of these two parts especially if they both operate within a common overarching domain such as the Department of the Navy, which provides for common terms and concepts. The difficult section is the Application Level Ontology. The concepts defined at this level depend strongly on the specific application domains to be encompassed by the interface, which dictates the kind of problems to be addressed, the method used to solve them, and the underlying technology which often contaminates the model of a particular application domain. Typical concepts in this layer are 'supply requisition', 'maintenance action', 'efficiency rating', and 'reliability index'.



Figure 1: Ontology Semantic Levels

## **Interoperability Example**

In order to better understand these concepts a simple example is in order. This example considers the relationships between supply and maintenance activities and the corresponding information systems that support them. Assume the supply and maintenance systems were initially developed in complete isolation from one another with the respective goals of automating the internal processes of the supply and maintenance departments. These processes were based on the flow of standardized paper forms through the various sections of the two organizations. The forms are delivered to inbox of a particular section whose members typically perform some real-world action that is recorded on the original form or on a new form resulting from the action. These records are then placed in the outbox of the section to begin the next leg of the journey specified by the department process. The automation provided by the information system of each of these two activities essentially mirrors the respective manual processes but replaces the physical entities of the process such as paper forms and in/out boxes with virtual representations that exist only within the confines of a computer.

Additionally assume that automation provided by these systems is internal to the corresponding activity, which requires the generation of physical artifacts to interface with dependent activities. The maintenance activity produces paper based supply requisitions for delivery to the supply activity, which acts to fulfill the request eventually producing a paper based shipment order that is returned to the maintenance activity indicating the requested physical parts that are to be delivered. Maintenance system users

that wanted to know the status of their shipment would contact Supply Department personnel by phone. Supply personnel would then query the system to provide a verbal status report to the Maintenance Department caller.

Internal to each of these activities, many other types of documents, and tables are employed to manage them such as maintenance and delivery schedules, shipping rates, and trouble-shooting protocols. In ontological terminology these two different sets of entities and artifacts are known as domains, which this example further specifies as the Maintenance Domain, and the Supply Domain. For the purposes of this discussion these domains can be thought of as having three layers. The semantic layer describes the structure of the domain entities and the relationships between them that together comprise a model for representing the corresponding real-world problems within the computer. This is a conceptual layer that is somewhat independent of the physical implementation.

The data or information layer contains specific instances of the semantic layer entities linked together in a manner that describes the complete contextual state of a given domain. This is a physical layer that requires a specific implementation paradigm. The entities and relationships defined in the conceptual layer may manifest themselves as linked class instances in an object-oriented paradigm or as related table records in a relational paradigm. The Agent layer contains the domain users and software agents that leverage the information layer to perform useful tasks. The data and information that flows between these domains can be called the Maintenance and Supply Interface Domain or just Interface Domain for short. The Interface Domain is the focus of this paper and subsequently of this example.

As both the example systems evolve and the internal automation is nearing completion or is at least well understood, it is natural that they look to extend the automation across the activity boundaries. In the proceeding sections this paper will use the domains and layers just described to present three successive levels of system to system interaction: Data Level System Interface, Information Level System Interface, and Information Level System Interoperability to characterize different ways in which this automation could be realized.

## **Data Level System Interface**

A data level system interface is characteristic of most of the interfaces between DOD systems at the present time. As the information to be exchanged enters into the interface domain, it looses most context because this type of interface views each exchange as unrelated chunks of data. In this case assume supply system developers are responsible for developing an interface to the maintenance system to periodically pull supply requisitions and the maintenance system developers are responsible for developing and interface to obtain supply shipment status information as requested by maintenance system users. Each group of developers design a record set for the required data, which together define the interface specification that is depicted in Figure 2.



Figure 2: Data Level Interface specification

Although the focus of the interface was the exchange of data rather than semantic content, there is still an ontology associated with the interface. The explicit record specifications (Figure 2) represent the application level of interface domain ontology for the example interface. Most of the attributes found in the record specifications are referenced to entries in the Defense Data Dictionary System (DDDS), which in this case serves the role of the Lower Level Interface Domain Ontology. By marking up the interface attribute definitions in terms of DDDS entities one can easily determine that NSN is the National Stock Number, JSN is the Job Serial Number, and DODAAC is the Department of Defense Activity Address Code. One can reference these in other documentation and data specifications to further ascertain the conceptual meaning associated with them.

While there is no explicit Upper Level Domain Ontology there is an implicit one, which greatly assists developers in finding the common ground to implement this interface. This upper level ontology is an implicit artifact of the standardized processes of the underlying domain, the Department of the Navy in this case, which defines common conceptual entities such as a Maintenance Action Form and a Supply Requisition Form. These common conceptual entities in combination with the attributes defined in the DDDS provide the developers of the two information systems a common vocabulary with which to discuss, design, and develop specific interfaces between their respective systems.

At this point the Semantic Layer of the Data Level System Interface has been defined and is depicted as the Part and Shipment Interface Specification in Figure 3. The Semantic Layer depicts the internal data models of the Supply and Maintenance domains as well but these fall short of an ontology or even of a specification because they are considered the private proprietary property of the individual organizations responsible for developing the respective systems. It is also likely that these internal models are more focused on the individual forms and tables that users want to appear on their screens rather than on the underlying semantic entities the screens were designed to display information about. This makes it difficult to understand the models outside the context of the applications they were designed to support. While the interface specification appears well defined, the context from which the data is extracted on one end of the interface and then inserted on the other is not addressed by the specification at all.



Figure 3: Data Level System Interface

The Data Layer of the Data Level System Interface realizes the interface specified in the semantic layer. In the case of this example, the maintenance system developers must be responsible for developing the report generator code that pulls the requisite data from the context provided by the maintenance data model to generate the list of part orders that constitutes the interface to the supply system and for developing the report translator code that translates the part shipment interface records into the context of the maintenance data model. Similarly, the supply system developers must be responsible for developing the report generator code that pulls the requisite data from the context provided by the supply data model to generate the list of part shipments that constitutes the interface to the maintenance system and for developing the report translator code that translates the part order interface records into the context of the supply data model. Neither group of developers is really sure how the other group generated the data they need nor are they sure of what the other group does with the data they generate as they have no visibility into each others data models. The report translators and generators depicted on the figure are representative of these hidden context shifts into hidden proprietary data models.

As data level system interfaces such as that described in this example go to the field problems often arise. Since developers are often guessing about the context on either end they often don't quite get it right. This requires the Logisticians and Mechanics that use the systems to perform in the field work-arounds to such as additional filtering or hand tweaking to the records generated from the external system before processing them. Users of these types of data centric systems are use to this sort of data massaging and their systems are well suited to this as the meanings of fields in a data level system are easy to use in locally defined ways; of course this further complicates the problem, as these sorts of local modifications require local tweaks to the interfaces and ultimately produce an interface that marginally accomplishes the intended purpose, is not well understood, and is brittle and difficult to maintain as the corresponding systems evolve.

#### **Information Level Interface**

An information level interface differs from a data level interface in several regards. Primary among these is the requirement for the systems being interfaced to be information centric rather than data centric. Information centric systems are based on explicit ontologies that model the underlying semantic entities of the domain rather than the data crunched by the currently favored domain processes or displayed on the screens of particular applications. The developers of an information level interface consider all the information to be exchanged (parts and shipments in this case) in a singular context which not only relates the entities to be exchanged to each other but to the context in which the entities are related at both ends of the interface. This is show in Semantic Layer of the Information Level System Interface depicted in Figure 4 by an Interface Ontology that overlaps into the Supply and Maintenance Domains. The Interface Ontology is marked up in terms of the shared (public) Supply and Maintenance ontologies and vice versa.



**Figure 4: Information Level System Interface** 

The interface ontology itself will now consist of multiple interrelated entities derived from a Upper Level Interface Domain Ontology that provides higher level semantic context to each entity type concretely defined and used in the interface proper. The Interface Ontology should also define the entities required to pull the interface information from the context of one system and to place it into the context of another, although these constructs may not be present in the physical implementations that transport the information from one system to another it is important that they are defined in the ontology to fully capture the semantic context of the information. The Upper Level Interface Domain Ontology may have existed prior to the development of the interface or may have been developed in conjunction with it. In either case it is important that the application level ontologies specific to the individual Supply and Maintenance Domains in turn utilize it directly or at least reference it by semantically marking up the entities in the ontologies of these domains to correlate them to the concepts defined by the Upper Level Interface Domain Ontology.

With a semantic layer thus defined the information level interface can do much more than generate simple fixed reports. Each system can expose a much more generalized query interface. The queries are formulated and the responses returned in terms of the entities defined in the interface domain ontology. This allows for a much more flexible interface that is more likely to survive evolving interface and system requirements over time. Note that in order to support a generalized query interface at least one additional interface ontology must be defined that defines the semantics of the queries, or commands that in turn uses the interface domain ontology as logical arguments. For this purpose, many well defined standards exist such as Structure Query Language (SQL) and Knowledge Interchange Format (KIF), or systems may expose their own proprietary but publicly defined interface such as the Object Management Layer (OML) employed by many of the systems developed by CDM Technologies.

Between the information and agent layers in Figure 4 are depicted synchronization channels. In this example the Maintenance and Supply systems are information centric systems that provide for the development of software agents by providing subscription services to client applications. A subscription service is key to agent development as it lets agents register for the ontological patterns that trigger it to action. In this manner agents can always be operating in support of their users, as they are always ready to act in fulfillment of their responsibilities without having to perform needless busy work querying the information store for conditions that may never arise.

#### **Information Level Interoperability**

The Information level interface of the previous section has a shortcoming in that the Supply System and Maintenance system must both explicitly query each other to receive new information. Whether or not any new information is available a query must still be run just to find out. On the flip side, immediately after a query has been run information could change in the source system that would not be reflected in the querying system until after processing the next query which may take awhile depending on the polling scheduled employed. This situation can be remedied by employing the same sort of synchronization channel used between the individual information centric systems and the agents they support that is show in Figure 5. The addition of a synchronization channel for the interface allows for the development of interface brokers. Interface brokers serve as agents in the systems they support by automatically synchronizing the state of the



system to the state of interest in an external system via the defined interface between the systems.

Figure 5: Information Level System Interoperability

This approach allows for true interoperability between the systems but is not without its own difficulties. Many of the entities exchanged between the systems correlate to items in the real world and thus have unique identities whose keys must be managed within the confines of a real system implementation. In this sort of information level interface this is typically accomplished by designating a single specific source for each type of unique entity. While this approach works well for interfaces in which only a few systems are participating in starts to break down in larger interoperability scenarios as each system broker must know about all the other systems participating and which system is designated to be the definitive source of which data. This approach requires much duplication of effort within the individual data brokers and introduces an undesirable coupling between the systems. One approach for dealing with this is the introduction of an interoperability server.

## **Interoperability Server**

An interoperability server elevates the interfaces between systems to the level of information centric systems themselves. This approach provides one common implementation of the individual system data brokers that know in which system or combination of systems to find information defined within the Interface Domain

Ontology. It also provides specifically for the management of unique entities that are shared across two or more of the interfacing systems.

Employing the concept of an interoperability server leads to a system of systems architecture that groups collections of systems that need to regularly exchange information into loosely coupled federations whose central hub consists of an specific instance of an agent based interoperability server that is configured to address the specific needs of the federation. This concept views the interoperability server as just another system which allows one to layer a hierarchy on this system of systems architecture where higher level federations may include as systems zero or more interoperability servers typically from lower level federations. Within the defense domain, one could envision the proposed system of systems hierarchy following along the lines of existing unit hierarchies within the individual services with a the top level of the hierarchy operating at the level of the joint chiefs of staff, and commander in chief. Of course, crossties between the individual units and services at the lower levels of the hierarchy may also exist. The end state of this vision is a single, albeit large and distributed, system of systems that incorporates the entire information infrastructure of the DOD. This system of systems is tailored to meet the specific needs of user communities at all levels, by utilizing the systems specifically developed to meet their local needs, and is adaptable to change due to the loose coupling between systems.



Figure 6: Interoperability Server

There will be several distinct ontologies associated with each Interoperability Server. System Interface Ontologies that are unique to each system participating in the federation will be used to define the interrelated logical constructs within the corresponding system that are targeted to participate in external interactions. For example, the System Interface Ontology for an air load planning system may include constructs to represent air transports, stow areas, and cargo items. Also associated with each participating system is an ontological map that defines the transformations required to translate information represented in the corresponding System Interface Ontology both to and from the Federation Interface Ontology. Federation Interface Ontologies that are unique to each federation will be used to define the interrelated logical constructs with which the client systems to the Interoperability Server may interact. This ontology defines all the information of common interest to the entire federation as opposed to the specialized interests of the individual systems that are participating in it. For example, the Federation Interface Ontology for a joint logistics transport federation, which interfaces specialized air, rail, and sea load planning systems may define a transport construct which is a generalization of the specialized air transport, rail transport, and sea transport constructs that may be defined by individual System Interface Ontologies of the participating systems. This ontology once established serves as a standard for the domain represented by the federation similar in concept to the enterprise models that were popularized in the late eighties and early nineties such as the DOD Logical Data Model but exist within a more manageable scope and are driven by the interoperability requirements of the Finally, an Interoperability Ontology shared by all Interoperability Servers federation. will be used to define the interrelated logical constructs associated with interoperating systems and the services provided by the interoperability server. These constructs are independent of the logical domain entities associated with a specific federation. For example, the Interoperability Ontology may define constructs such as query, constraint, system, and ontology.

#### Summary

The key to the interoperability between systems lies with well-defined system and interface ontologies. An ontology makes explicit the conceptualizations used and shared by the interoperating systems. The shared conceptualization is known as the interface domain ontology. The interface domain ontology and the individual system domain ontologies should both be well marked up in terms of each other and ideally share both an upper level and lower level interface domain ontology. In this manner, the mappings that determine the context of interfaced entities on either side of the exchange are made explicit and are more likely to endure evolutionary changes to the systems and local modifications or special case usages. For systems to truly interoperate rather than just interface some sort of synchronization channel must be provided. As the number of systems participating in an interface grows even a well-designed information level interface can become unmanageable and an interoperability server approach should be considered.