From: AAAI Technical Report WS-99-14. Compilation copyright © 1999, AAAI (www.aaai.org). All rights reserved.

Semantic Context for Object Exchange

Leo Obrst, Gregory Whittaker, Alex Meng

The MITRE Corporation
1820 Dolley Madison Blvd., W640 McLean, VA 2210
{obrst, greg, meng}@mitre.org

Abstract

Enterprise wide interoperability means distributed computing on heterogeneous hardware using heterogeneous implementation languages. Successful information exchange between interoperating systems today depends on data standardization. There are several problems, however, with this approach. This paper looks at some ingredients for a more complete notion of semantic interoperability. Semantic interoperability is defined as the enablement of software systems and in particular object-based systems to interoperate at a level in which the exchange of information is at the enterprise level. This means each system (or object of a system) can map from its own conceptual model to the conceptual model of other systems, thereby ensuring that the meaning of its information is transmitted, accepted, understood, and used across the enterprise. We argue that the primary way by which semantic interoperability can be realized is by defining a notion of context which includes the object to be exchanged and its internal state, its interpretation with respect to both the source and the target system object models, and the particular use of and intent for the object in both the source and target systems.

Introduction

Our entry point into a discussion of the notion of context is by way of considering the requirements of semantic interoperability for interacting distributed object systems. Enterprise wide interoperability means distributed computing on heterogeneous hardware using heterogeneous implementation languages. Successful information exchange between interoperating systems today depends on data standardization. There are several problems with this approach. In order to satisfy the needs of all consumers a particular standard data item ideally provides the union of all consumer needs, while at the same time the data standard can only be depended upon for the intersection of all producer output data elements. One can view this problem as a dispute over a mapping from one multidimensional information space, or context, to another where neither producers nor consumers agree on all of the pertinent dimensions or basis vectors. A data standard is in effect a particular rigid translation between these spaces, or contexts. The data value or meaning of such a translation is dependent on the basis vectors assumed by the mapping. The fact that one set of basis vectors does not fit all needs results in tension between multiple users and has led to a plethora of closely related data standards.

For the purposes of this investigation, semantic interoperability is defined as the enablement of software systems and in particular object-based systems to interoperate at a level in which the exchange of information is at the enterprise level. This means each system (or object of a system) can map from its own model to the conceptual models of other systems, thereby ensuring that the meaning of its information is transmitted, accepted, understood, and used across the enterprise. This is analogous to providing the basis vectors of the originating information in addition to the particular embodied values so that the basis can be compared, added or transformed into the information space of the receiver and properly translated for application. Misinterpretation often arises from an implicit projection of local meaning on foreign data. We differentiate, however, our object system-based view of semantic interoperability from that of the database community, though of course this literature has much to bear on the topics of this paper.[1]

The fundamental problem for inter-system semantic interoperability is to define what a context is, the necessary elements of any specific context, and how one can map between or compare contexts. Although our focus in this paper is on the notion of contexts for semantically interoperable object-based systems, we believe our discussion has applicability to the wider "context" research community.

The format of this paper is as follows. First, we discuss the dimensions of semantic interoperation for distributed object systems, arguing that a context for semantic interoperation needs to include interpretations of the object with respect to its source system and virtually with respect to its target system which involves both the semantics and the use and intent of those systems. Also required then is a mapping from the source interpretation to the target interpretation of that object. We illustrate the notion of semantic interoperation with some increasingly more complex examples, thereby demonstrating the need for our hypothesized contextual components. Second, we describe more formally the components of the individual aspects of context and point to future elaboration.

[1] See, for example, (Kashyap and Sheth 1997, 1998; Mena et al.. 1998; Ouksel and Naiman 1994; Subrahmanian 1994; Sheth and Ouksel 1999; Smith and Obrst 1999).

Basic Idea

The fundamental concept of object exchange is that semantic interoperation happens when objects, viewed as messengers, are properly *interpreted and used by interdependent applications* (for general notions of formal interoperability among systems, see (Meseguer 1998)). We envision the future in terms of objects traveling across dynamically determined domain boundaries and application *contexts* arising in the course of execution. Conceptually, a context embodies the mechanism necessary for a system to receive an object at runtime and to examine, interrogate, and interpret it to determine if it can be assimilated. More formally, a context can be viewed as a tuple with the following components:

(1) Def. Context := <O_1, O_2, [[O_1]], [[O_2]], map([[O_1]], [[O_2]]>

A. Object O_2, a variable or virtual object, the kind of object that the receiver system can currently handle
B. Interpretation of object O_1 in sender object model OM_1 (language L_1, models M_1)
C. Interpretation of object O_2 in receiver object model OM_2 (language L_2, models M_2)
D. Mapping between the interpretations of objects O_1 and O_2 (accessibility relations R_i defined in a language L_3, having models $M_3 \subseteq M_1 \bigcap M_2$.

This is a complex context with respect to two (possibly) interacting systems. Abstracting away from multi-system interaction, an object always exists in a simple context in its source system. This simpler context is minimally a 4-tuple <object O_1, object model $OM_1$1, environment E, interpretation(O_1, OM_1, E)>, where interpretation(O_1, OM_1, E) denotes the interpretation of O_1 with respect to OM_1 and dynamic environment E. The environment here includes how O_1 is being used and the specific dynamic objects it is interacting with. The environment is dynamic because if O_1 is persistent, it can be used in many ways and can interact with many different objects over the course of its lifetime, producing and undergoing changes of state within the same system. In this paper, we abstract away from the environment, though it should be remembered as being implicitly always present.

An application's context is fractal in nature, consisting of conceptual abstractions of its world situated in domain dependent relations to other worlds each of which has its own constellation of logical models and interpretations. Exchange of information in today's execution environment depends on an *implicit sharing* of context during execution. *Values are explicitly* exchanged through a combination of formal data standards, application program interface specifications, interface control documents, and common interface definition language specifications. *Meaning is implicitly projected* onto the exchanged values. Meaningful exchange of information is therefore limited to what can be anticipated during software design and prescribed at build time. Conceptual models of the mission space are compiled away; context is not represented in the execution environment. Today, rapid response to novel or unpredictable situations is therefore limited to deployment of a set of fixed but inefficient or even inappropriate configurations of enterprise systems.

Our Hypothesis

> *Providing semantic interoperability in the execution environment will greatly enhance the speed of customized deployment and the operational effectiveness of deployed systems. This semantic interoperability will require a formalization and implementation of contextual mechanisms.*

Emerging technology is making it possible to dynamically discover and use network visible services. Modern programming languages and middleware have long permitted remote invocation of functions and services, but it is now possible to dynamically embed remote objects along with their methods. New systems will build on this technology in order to significantly enhance flexible and opportunistic exchange of *information and capabilities*. A primary key to the success of this exchange is tracking and mapping the domain specific models' design, making explicit what has formerly been implicit.

Modes of Semantic Interoperation

In the following sections we list several modes of interoperation considered in our investigation. Here we emphasize the effects, not the mechanics. There are many approaches to the end of semantic interoperability. Some technical approaches focus on an interlingua, others on underlying meta-models, still others on families of translations in combination with a priori fixed points of common agreement (standards), and some on either a global ontology (which we will avoid here) or a set of ontologies. The crucial issue that must be confronted, however, is that there are many different models and many different languages which express the semantics of would-be interoperating systems. Can an object migrate from one context, its source system having its own semantics, to another context, a target system having a different semantics, and be utilized? Can the semantics of two or more systems be compared when their representation languages and models are sometimes quite different? What are the issues involved in interpretation and mapping between interpretations when the context changes?

Semantic Interoperation via Conceptual Pivoting

This mode of interoperation could also be called *the alignment of terms problem* where terms correspond to the

nouns and noun phrases of an English sentence; when we know we are speaking about the same things, we have come a long way in understanding foreign expressions. Consider the following:

(1) an object representing concept x from domain X travels to an application in domain Y containing concept y

(2) concept x = concept y

(3) logical description x • logical description y

Assumptions:

(a) ***concepts*** are first class entities

(b) there is a universal equality test for concepts that allows us to test concepts from different domains to see if they are equal (better still would be a conceptual proximity measure so that we can tell if we are getting *close*)

(c) concepts are represented as objects with faceted attributes

It will be very important for systems to delineate the conditions under which two objects can be said to model the *same concept*. Semantic interoperation in this mode is fundamentally based on conceptual identification followed by elucidation of essential attributes or *essences* of the concept. The trouble is that from one domain to the next what appears to be essential to the same concept is not constant. This leads to a quandary over completeness of the conceptual representation. If a concept is essentially described by a set of attributions in one domain can it be said to be the same concept if those attributions are not translatable to attributions of the candidate concept in another domain? Even if two concepts are linked by string-equal terms, in general, no correlation can be assumed. Term equality cannot furnish semantic information unless individual terms are uniquely defined either in an indirect standardization document which assumes human-supplied semantics (and which typically is not machine-represented) or within an online namespace or ontology (which is machine-represented). A namespace here is taken to be just an impoverished ontology: an ontology makes explicit not only the entities of a domain, but their relationships and constraints on those relationships.

Further refinements of the *same concept* notion apply to the facets of roles, responsibilities, limitations, expectations, potential and intended uses of concepts modeled as objects, attributes and methods.

Semantic Interoperability via Method Adoption

Once we have achieved conceptual unification and have mapped the foreign attributes to local attributes that our application *understands*, we may venture to interpret and use foreign methods. Mapping the terms of a foreign method to types in a local application will assist in the interpretation of foreign methods by comparison of method names and signatures. An entire spectrum of interpretation mechanisms is possible starting with syntactical level mappings of object attribute and method names to local attribute and methods and going on to deeper analysis of logical models. Implementing alternative interpretation techniques in accord with conceptual model scopes is a reasonable approach. Some terminological mappings may be factored into a sequence of commonly used translations between frequently interoperating models; these factors may in turn be stored and recalled for partial composition of more extensive, less frequently used translations.

Semantic Interoperation via Object Substitution

In this mode we are using a foreign object in lieu of an object from our own domain to accomplish some purpose. As an example, we are adapting a Navy model object (submarine) to an evacuation task not in the Navy model. The supposition is that in this rescue operation the expected evacuation vessel would be a ferryboat existing in the Commercial Port model. Prior to the submarine object arriving in the new Commercial Port model, the old context includes the object instance Military_Sub and its class hierarchy (and state models, constraints, etc.) After arriving, the new context is the following:

New Context = Old Context + CoastalFerryShip

This new context might be partially expressed by interpreting Military_Sub as filling the same *role* in a larger plan that is typically filled by the object instance CoastalFerryShip. More generally, the object class of which CoastalFerryShip is an object instance (in this case 'ship' in the Commercial_Port_App model) could be specified in the template. So, a general action template

<plan Identifier: Action(type$_i$ Role$_i$, type$_{i+1}$ Role$_{i+1}$, …, type$_{i+n}$ Role$_{i+n}$)>

is elaborated as:

Plan Y: evacuate (entity AffectedEntity, location Source, location Target, vehicle Instrument)

= evacuate (horse Horses, location Chincoteague_Island, location Virginia_coast, ship CoastalFerryShip)

where Horses, Chincoteague_Island, Virginia_coast, CoastalFerryShip are specific instances. This will get transformed to the new context:

Evacuate (horse Horses, location Chincoteague_Island, location Virginia_coast, ship substitute (Military_Sub,CoastalFerryShip))

Military_Sub will be substituted for CoastalFerryShip to fill the Instrument role. This process is similar to the employment of analogy: a linkage is established to a target space from a source space, i.e., 'X is Like Y', with the result that the source is interpreted in the conceptual space of the target. This means that source X is substituted for Y,

and Y's attributes methods, states are coerced (reinterpreted) to apply to X.

Formalizing Context[2]

We think that a context needs to be structured as we have done in (1) because it must refer minimally to paired semantic frames of reference and the mappings between them. In the following discussion, we briefly describe the components of a context and sketch some steps towards a formalization.

First, we treat the object model of a system as an idealized "theory" of that system. The theory is characterized as *idealized* because usually an object model of a system is not formally well-defined, neither complete (which is expected), nor consistent. We abstract away from actuality here, and posit idealized theories, i.e., what an object model *would be* if it were well-formed and coherent. Usually an object model is less specified, rather than more; hence, we consider it the base theory of the system. Note that we could analyze our object model in terms of *concepts* more directly, perhaps along the lines of lattice-theoretic formal concept analysis (Davey and Priestley 1991, pp. 221-236) in which a *concept* is an ordered pair of <its extension, its intension> within a *context* defined as a triple of <objects, attributes, has-relation>. But we believe that the language-theory-model approach is clearer.

A theory is a set of assertions in a first-order language L (which is a set of symbols S and inference rules for generating well-formed formulas from the symbols) and an interpretation of those assertions with respect to a model. L thus has a set of formal models $M' \subseteq M$ associated with it and an interpretation (assignment) function I by which objects in M' are assigned to the symbols of L. Care must be taken not to conflate the two uses of 'object' mentioned here. On the one hand, an 'object' is a construct in an object-oriented programming language or system; on the other hand, a formal 'object' is a construct in a formal model in formal semantics. We will specify the first kind of object as a 'distributed object' or within 'the object model of a system'.

M' represents the domain (of the distributed object model). Its formal objects are meant to represent the real-world or, more accurately in our case, the information-world objects of the distributed object system, i.e., symbolic correlates of submarines, coastal ferries, horses, locations, times, the relations among these objects, their attributes, states, methods, etc. The mappings by I are considered the interpretations of the symbols of L in M'. A formula or sentence φ of L (sequences of symbols permitted by the combinatoric rules of L) is said to be *satisfiable* in M' , written as M' |= φ iff there is some assignment φ_i which is true in M'.

In order to connect this terminology to the notion of an *ontology*, we follow (Guarino 1998) and define an *ontology* to be a logical theory accounting for the intended meaning of a formal vocabulary S in terms of an ontological commitment to a particular *intensional* (modal) conceptualization of the world, where a *conceptualization* C is a structure <M", W, R>, M" is the model or domain, and W is a set of relevant states of affairs of that domain (i.e., possible worlds), and R is a set of conceptual relations on < M", W> such that each ρ^n of R is a total function $\rho^n: W \rightarrow 2^{M"^n}$ from W into the set of all n-ary relations on M" (Guarino 1998). To this point, the only difference then between our M', which represents the particular formal model of the system object model, and an ontology is that the ontology is specifically meant to refer to an intensional conceptualization, i.e., it holds over a set of possible worlds, in which class-level assertions such as Man(X) can have many extensions or instances. In our view, this simply means that a specific system object model *could* participate in a larger ontology which spans other object models. However, by our current definition, a system object model represents a specific domain theory, which only holds in one world, but which could participate in a larger ontology, spanning other domain theories. All of this is just to say that, by our definition so far, an individual object model is not intensional. But, contrarily, we could promote an individual system object to an intensional interpretation, by which it then becomes equivalent to an ontology (and that perhaps is not so far-fetched, since such object models of systems, though independently constructed, do participate in the background, common, human interpretation of the world).

The problem arises when one wants to compare system object models (to gauge first and then create mappings between) which either do not participate in the same ontology (recall that we assume there is no global ontology), under the first view, nor are the same ontology, under the second view. It seems that by assuming there is no global ontology, then either we must assume a pair-spanning (local, intersective) ontology [Ontology (Object_Model$_1$ $\vee$ Object Model$_2$)], or two ontologies [Ontology(Object Model$_1$), Ontology(Object Model$_2$)]. If we take the latter tack and intensionally analyze individual system object models, then we are forced to consider not only *accessibility relations* between possible worlds (required for a single ontology), but accessibility relations between two universes of possible worlds, to determine the mappings between system object models. To talk about (let alone, determine) accessibility relations between two universes of possible worlds, one needs a language or a meta-language. Otherwise, the formal objects in the two universes are incommensurable. So (1.E) defines a set of mappings between the interpretations of objects O_1 and O_2 as the accessibility relations R_i defined in a language L_3,

[2] We acknowledge a large debt to others who have tried to formalize context: (McCarthy and Buvac 1997; Lenat 1998; Serafini and Ghidini 1997; Giunchiglia and Bouquet 1997, 1998; Giunchiglia and Ghidini 1998). Due to space requirements, however, we cannot discuss these other approaches here.

having models $M_3 \subseteq M_1 \| M_2$ each of which is implicitly subscripted by different worlds. Note that we would like an intersection-language, but initially may have to settle for a union-language.

The individual objects O_1 and O_2 are listed separately as components of the context because we recognize that the object from the sending system may have internal state not ascertainable by inspection solely of it original object model (even if state and/or process models exist, they may not be extensive or directly correlated with specific object instances), so a behavioral description of the object will tend to be incomplete. A fuller specification of the pragmatics of the source system (and the target), its intended use or the work it is meant to perform, will ultimately need to be incorporated into the interpretation. We have suggested a task/workflow structure shared by would-be semantically interoperating systems as a step in this direction, fully understanding this implies a rudimentary common language, which we seek to avoid. However, pragmatic commonality may be necessary. Concerning the O_2 object, we have characterized it as virtual, viewing it as a hypothetical or variable object (perhaps partially specified), that object of the target system which would most closely approximate semantically the O_1 object. This is of course what the contextual resolution is about: to find some way to coerce the O_1 object into the O_2 object, practicably perhaps by way of some intermediate interpretation arrived at by a unification (constraint solving) process.

We think that ultimately a more general formalization of context (directly applicable to the context of distributed object systems) can be constructed in the category theory-influenced Information Flow theory of (Barwise and Seligman 1997), a framework more general than possible worlds, allowing also *impossible worlds*. But this is a prospect for the future.

Acknowledgements

Portions of this work were performed as part of a technical investigation for the Information Systems Office of the Defense Advanced Research Projects Agency (DARPA). We'd like to thank Dr. Todd Carrico of DARPA for his support and technical suggestions, and John Anderson and Ann Jones for their encouragement.

References

Barwise, Jon; Seligman, Jerry. 1997. *Information Flow: The Logic of Distributed Systems*. Cambridge University, Press, Cambridge, UK.

Davey, B.A.; Priestley, H.A. 1991. *Introduction to Lattices and Order*. Cambridge University Press, Cambridge, UK.

Giunchiglia, Fausto; Bouquet, Paolo. 1997. Introduction to Contextual Reasoning: An Artificial Intelligence Perspective. Istituto per la Ricerca Scientifica e Tecnologica (IRST), Trento, Italy, Technical report 9705-19, May, 1997.

Giunchiglia, Fausto; Bouquet, Paolo. 1998. A Context-Based Framework for Mental Representation. Istituto per la Ricerca Scientifica e Tecnologica (IRST), Trento, Italy, Technical report 9807-02, July, 1998.

Giunchiglia, Fausto; Ghidini, Chiara. 1998. Local Models Semantics, or Contextual Reasoning = Locality + Compatibility. Principles of Knowledge Representation and Reasoning (KR'98), Proceedings of the Sixth International Conference, Trento, Italy, June 2-5, 1998, Anthony Cohn, Lenhart Schubert, Stuart Shapiro, eds., pp. 282-289.

Guarino, N, ed. 1998. *Formal Ontology in Information Systems*. Amsterdam.: IOS Press. Proceedings of the First International Conference (FOIS'98), June 6-8, Trent, Italy.

Kashyap, V.; Sheth, A. 1996. Schematic and Semantic Similarities between Database Objects: A Context-based Approach. VLDB Journal 5 (4), 1996.

Kashyap, V.; Sheth, A. 1997. Semantic Heterogeneity in Global Information Systems: The Role of Metadata, Context and Ontologies. In M. Papzoglou, and G. Schlageter, (Eds.), Boston: Kluwer Academic Press, 1997.

Lenat, Doug. 1998. The Dimensions of Context-Space. Cycorp, Austin, TX, Technical Report, October 28, 1998.

McCarthy, J.; Buvac, Sasa. 1997. Formalizing Context (Expanded Notes). In *Computing Natural Langauge*, A. Aliseda, R. van Glabbeek, & D. Westerståhl, eds., Stanford University. http://www-formal.stanford.edu.

Mena, E.; Kashyap, V.; Illarramendi, A.; Sheth, A. 1998. Domain Specific Ontologies for Semantic Information Brokering on the Global Information Infrastructure. In Guarino, ed., 1998, pp. 269-283.

Meseguer, José. 1998. Formal Interoperability. In Proceedings of Fifth International Symposium on Artificial Intelligence and Mathematics, Fort Lauderdale, Florida, January 4-6, 1998.

Ouksel, Aris M. 1999. Ontologies are not the Panacea for Data Integration. To appear in Journal of Parallel and Distributed Systems, 7, 1-29, 1999.

Ouksel, Aris M; Naiman, Channah. 1994. Coordinating Context Building in Heterogeneous Information. Journal of Intelligent Information Systems 3, 1, 151-183, 1994.

Serafini, Luciano; Ghidini, Chiara. 1997. Context Based Semantics for Information Integration. Istituto per la Ricerca Scientifica e Tecnologica (IRST), Trento, Italy, Technical report 9709-01, September, 1997.

Sheth, Amit; Ouksel, Aris, eds. 1999. SIGMOD special issue on Semantic Interoperability. March, 1999.

Smith, Ken; Obrst, Leo. 1999. Unpacking the Semantics of Source and Usage To Perform Semantic Reconciliation. In Sheth and Aris Ouksel, eds., 1999

Subrahmanian, V.S. 1994. Amalgamating Knowledge Bases. ACM Transactions on Database Systems, 19, 2, pp. 291--331, 1994.