Models for semantic interoperability in service-oriented architectures

G. Vetere
M. Lenzerini

Although service-oriented architectures go a long way toward providing interoperability in distributed, heterogeneous environments, managing semantic differences in such environments remains a challenge. We give an overview of the issue of semantic interoperability (integration), provide a semantic characterization of services, and discuss the role of ontologies. Then we analyze four basic models of semantic interoperability that differ in respect to their mapping between service descriptions and ontologies and in respect to where the evaluation of the integration logic is performed. We also provide some guidelines for selecting one of the possible interoperability models.

INTRODUCTION

Modern information technology (IT) systems based on service-oriented architectures (SOAs) consist of a network of service providers. Services are invoked by client applications (consumers) by means of messages that conform to *descriptive schemas*. Although typically service descriptions are exported by providers to *registries* (or *directories*), some service descriptions may also be supplied directly to consumers. What is crucial—and different from client/server architectures—is that, for each service, a schema defines its functionality and registries are available for lookup and binding of services without knowledge beforehand.

The services supported by a provider give access to the provider's *state* and allow making changes to that state. Here we use "state" in the classic way, as the actual values of a given set of *attributes*. For instance, a data service can be seen as a database management system whose state is the actual data in the database, and in which client applications can perform read and write operations through a set of services described in terms of the allowed get and set operations, along with a portion of the database schema exported in a suitable way. In summary, by using the service descriptions available, applications can access and manipulate the state of providers.

In order for client applications to use services effectively, it is crucial that the designers of these applications understand the service descriptions that

correspond to the client's operations and data structures. Similarly, developers of distributed information systems that cooperate through services should have a good understanding of how service descriptions relate to one another. Methodologies, artifacts, and techniques aimed at the correct interpretation and implementation of service descriptions are generally referred to as the "semantic layer" of service-based infrastructures.

In a service-oriented environment, the semantic layer ensures that data embedded within messages are interpreted by providers and consumers as representing the same concepts, relations, or entities in a suitable abstraction of the real world. For instance, the semantic layer helps detect that the attribute *tour-cost* at *www.grand-tour.com* corresponds to *ticket-price* at *www.railways.it*. Rephrased in the "philosophical" jargon currently in use by the computer science community, the semantic layer is about how participants can interpret descriptions and data items in the system with respect to some *ontology*[1] of the business domain and how this interpretation can be shared and made transparent throughout the infrastructure.

The semantic layer also includes operational aspects, such as the definition of business transactions as presented in RosettaNet[2] standards. In fact, the semantic layer can be viewed to contain anything that can be entered in a data vocabulary with the purpose of characterizing the provider's service. In other words, it covers objects, events, states, and anything else that can be conceived, expressed, and exchanged over a communication network, which in fact amounts to the entire coverage of a standard linguistic dictionary. Designers—consciously or not—have to decide how to manage the semantic layer and have to make assumptions about it. The aim of this paper is to provide SOA developers with basic conceptual tools to better understand the semantic layer. Because handling semantics is not an easy task and important research issues are still under investigation, this paper aims at giving general guidelines rather than ready-for-use instructions.

Our interest in semantics is not primarily motivated by advanced functions such as intelligent service discovery or the composition and choreography of automatic services, but by the need to make heterogeneous information systems work in a networked world. For this reason, we focus here on the interoperability of services as is worked out by humans, and we do not survey the rich literature related to the use of semantic models for automating service integration tasks.[3]

Motivation

In current practices, semantics is usually relegated to the backstage of design and implementation activities and is not often given the place it deserves. Semantic correspondences are captured by message transformation rules that map names, values, and structures in these messages exchanged by services. In general, this approach can be seen as *semantically neutral*, in that it does not require the mappings to account for the way in which the source and the target refer to real entities. In other words, message transformation rules are written without accounting for the reason why the corresponding mapping holds. This strategy can be seen as a pragmatic way to address semantic interoperability in many common situations, when it is reasonable to assume that the interpretations given the service descriptions by cooperating participants are consistent. In many cases, this neutrality presupposes a sort of *realism*: the idea that attributes which shape the domain of the service infrastructure are given by nature, once and for all. Thus, because services implicitly use the very same ontology, the role of designers is just to "neutralize" a number of different naming and structural renderings of elements of a unique, universal, immanent conceptualization. This is probably why SOA technologies are mature in supporting rich-message mapping and transformation languages, but are "green" (immature) in providing standard means to drive the development of specific semantic-oriented artifacts, and are generally silent about the conditions that must be ensured in order for message transformations to be semantically sound. As a matter of fact, whereas current frameworks provide a fairly good basis to handle descriptive heterogeneity through a syntactic approach, the treatment of semantics is generally left to designers, if and when they feel the need to bring semantics to the foreground.

Being neutral with respect to semantics is reasonable in many situations, but it doesn't work in general. In fact, this approach assumes the reliability of some implicit agreement that lives outside the infrastructure, let us say, in the system's social

surroundings. For instance, guessing that two attributes refer to the same entity if they have the same name is based on the assumption that labels are uniformly interpreted by all parties. This could be acceptable if service infrastructures were built, deployed, and managed within tight organizational boundaries, as in enterprise application integration scenarios, where naming policies can be enforced and controlled. But SOAs could be used to implement information systems in large and geographically distributed organizations, such as supply chains of cooperating but independent companies, or even to manage structured information exchanges across quite "anarchic" Web communities. In these cases, designers can hardly make reliable assumptions about the infrastructure's social surroundings. Actually, they cope with information providers that are only requested to sketch the kind of services they make available, mostly from a functional standpoint. Semantics, which is the way services fulfill what their descriptions promise, is embedded in the "black box" of service implementation. In these scenarios, the reach of semantic agreements is problematic: accurate derivation of meaning cannot be implicit, and therefore the semantic layer requires specific tools and methods.

Semantic interoperability between services in an SOA is not very different from linguistic understanding between humans. Linguists and philosophers have been engaged in studying the foundations of natural language semantics for centuries, and many authors have even been skeptical about our real capability to understand each other.[4] The attractive idea of realism, which maintains that humans understand each other because the language reflects a commonly understood world and that differences are only at the linguistic surface, has been strongly rejected by many philosophers, especially in the last century. Quine, for instance, claimed that "ontological commitments" are relative, and there is no way to tell whether an expression uttered by a foreign speaker in the presence of a rabbit denotes the entity (i.e., the rabbit) or the event (i.e., the presence of a rabbit), very different elements from an ontological standpoint.[5] Applied to SOAs, Quine's assertions would amount to the statement that there is no unique interpretation of a service specification, just based on the specification itself. Although we will not engage in a philosophical discussion here, if we approach semantic interoperability from a relativistic standpoint, there are no *naturally safe semantic assumptions* when dealing with cooperating services. Therefore, relativism requires much more focus on semantics than realism, and this maybe explains the reason why this discipline has gained popularity within the computer-science community since the Web became pervasive.[6]

We believe that awareness of semantic issues when designing interoperating services helps deliver better solutions, not only for the Web, but also in restricted environments such as e-government or large enterprises, in which designers can rely on stable semantic conventions. With this in mind, in the next section we provide some basic concepts for understanding semantic interoperability. Then, we classify the models for semantic interoperability in service-oriented infrastructures, and take into account whether they reflect a hub-based or endpoint-based structure, and whether they rely on business domain models. We also show how problems of semantic interoperability are close to many of the issues that have been studied for decades in data exchange and integration under the lens of logic,[7] and how some of the concepts developed in this field can be usefully applied when designing the new kind of IT infrastructures.

SEMANTICS OF SERVICE INFRASTRUCTURES

We have shown in the previous section how services allow accessing and changing the state of their providers, that is, the values of some exported attributes, and how this access and manipulation is carried out by sending and receiving messages. We have also informally introduced the semantic layer as a framework that allows a consistent interpretation of messages sent to and received from services with respect to some conceptualization underlying the system. Now we discuss semantics in more depth, starting from the characterization of a single service and then considering the interoperability (integration) of services.

Informally, a service is a system with an internal state (e.g., values in a database instance) characterized by an internal schema (e.g., a relational one) that exposes a set of access and manipulation methods through an interface. The interface of a service is based on two alphabets of symbols—the *operational alphabet* and the *data alphabet*—and a set of structuring primitives and logic connectives

```
<message name="routeRequest">
      <part name="start" type="eutb:place"/>
      <part name="end" type="eutb:place"/>
</message>

<message name="routeReply">
      <part name="ID" type="xsd:token"/>
      <part name="fare" type="xsd:double"/>
</message>

<portType name="EUTBService">
   <operation name="getRoute">
         <input message="routeRequest"/>
         <output message="routeReply"/>
   </operation>
</portType>
```

Figure 1
A fragment of a WSDL description that references an XML Schema

that make up the *description language*. This provides designers with means for formulating a public characterization of the service called *description*. For instance, in ***Figure 1***, WSDL (Web Services Description Language) and XML (Extensible Markup Language) Schemas jointly provide a language that informs whether an operation called `getRoute` is available and corresponds to a function that maps *places* to *routes*. As such, descriptions can be regarded as "theories" that provide definitions and constraints to explain the service ontology and behavior. The interface is bound to the internal service state by means of a suitable implementation.

The problem of semantic integration results from the fact that, whereas service descriptions are public, service implementations are not accessible from outside. Thus, for client and provider alike, the correspondence between the service description and the internal state is essentially opaque (i.e., hidden). Consider, for instance, the client application Grand Tour Agency in ***Figure 2***. It uses services provided by European Travel Bureau, which acts as a broker and which uses services provided by Italian Railways. To plan a trip, Grand Tour Agency specifies *train* as the preferred means of transportation and requests from European Travel Bureau the set of routes (and the associated price) between two destinations. To fulfill the request European Travel Bureau uses Italian Railways services. As broker, European Travel Bureau has to be able to interpret the information on trains that cover the specified route supplied by Italian Railways, and thus that the fare, as interpreted by the client, amounts to the cost of a train ticket, as intended by the supplier. As shown in Figure 2, in order to manage this cooperation, the three participants export schemas and exchange data based on mappings. Schemas expose the relational predicates of each service in an abstract notation, whereas mappings represent logical implications between these predicates. To indicate that these mappings can be implemented by a variety of data transfer policies, we use squiggly arrows instead of the classic straight ones. Note that whereas both consumer and provider have internal systems (databases, files, programs) that implement functionalities and data structures to fulfill the exposed schemas, the broker system is mostly concerned with managing and evaluating schema mappings.

The client uses the service to get data, then processes the data and possibly updates its databases. It may also send messages containing data with the purpose of changing the state of the provider. The data and operational language that describes the service must be mapped to data structures and state transformations at the client. For developers at Grand Tour Agency this raises two semantic issues: producing proper messages for the service (either to get or set data), and correctly interpreting the information returned, which implies getting the meaning of the expressions accepted and produced by the service, making sure that this meaning is consistent with the way the service-provided data is used by the client application. Conversely, for developers at Italian Railways, the problem is fulfilling the service schema by interpreting the meaning of descriptive expressions provided by European Travel Bureau. But what does "interpreting the meaning" really mean? Broadly speaking, this is related to what, in logic, is called *interpretation*. To explain this, we start with first-order logic (FOL) semantics.

FOL semantics

An *FOL interpretation* of a service interface is basically a function that associates symbols of the data language to tuples of data items in the service provider's internal state (also known as the *extension* of the symbol[8]), and the meaning of a description is simply a calculus of its extension based on logic connectives. It is easy to see that this corresponds to standard database semantics. For

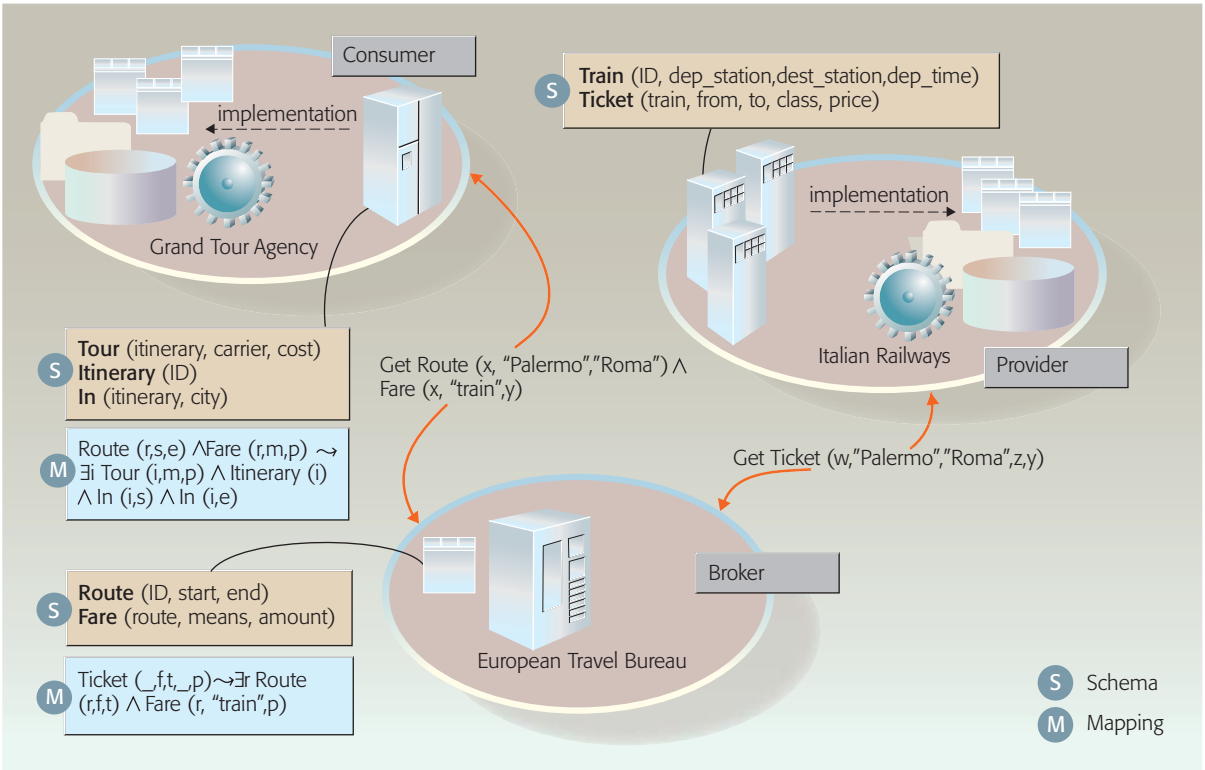

Figure 2
An example of semantic interoperability involving a client application, a service provider, and a broker

example, the meaning of the expression $\{id,tm \mid Train\ (id,$ "*Rome*", "*Milan*", $tm)\}$ (i.e. a set of identifiers and departure times on the relation Train with departure and destination constraints), under a FOL interpretation in the service provider's domain of Figure 2, would be a set $\{<id, tm>_1, \ldots, <id, tm>_n\}$ of tuples in the internal database. A *model* for a service based on its interface is an interpretation that satisfies all the formulas of the service description with respect to the service provider's state, and any state of a service, if consistent, is supposed to be a model in some suitable interpretation. Indeed, FOL semantics is straightforward: service providers can reasonably be supposed to provide data according to their models, and achieving semantic interoperability just requires consumers to preserve those models, taking the description language properly into account.

Unfortunately, a number of limitations affect the use of FOL semantics in loosely coupled scenarios such as Web Services interoperability. First, observe that the implementation of the provider is not visible to consumers. This means, for instance, that the extension of predicates (symbols in the data alphabet) can be changed by providers. Clients have little control on the way these extensions vary over time: they cannot poll their providers too often, and therefore models could easily get out of sync. Also, consider a data integration scenario in which a client wants to feed its model with data coming from different sources (suppose, for instance, two services exposing the same interface), and suppose that models contradict one another (e.g., service α says that the train will start at 1 p.m., and service β maintains that the same train will start at 2 p.m.). How can the client model be determined in a situation like this? Finally, consider the typical case in which a consumer integrates data coming from a provider into the model of a broader theory, which includes predicates and constraints borrowed from other services with mappings to bind them together; how can it be ensured that the ontological commitments made by each provider are correctly understood (e.g., *Train* is intended as *object* and not as

event); thereby ensuring that inter-schema mappings are sound?

It seems that extensional FOL semantics is not able to address problems that originate in the opacity of service implementation in SOAs. In summary, in order to manage semantic compliance in an FOL setting, all the implementations (of both providers and consumers) should collapse into a single one, which is clearly not the normal condition in service-oriented infrastructures. To overcome limitations of FOL semantics, service descriptions should be interpreted with respect to conditions that occur in sets of "possible states of affairs" by providers. This is the question we examine next.

Modal semantics

The German logician Gottlob Frege called *Sinn* (sense), as opposed to *Bedeutung* (meaning), the account of the *intension* of logic predicates, rather than their *extension*.[9] Informally, an intension is a condition that objects in a given domain must satisfy in order to enter the extension of a predicate; for example, whereas the extension of Train in Figure 2 is precisely a set $\{train_1, \ldots, train_n\}$ of records in the database of Italian Railways, the intension is the set of rules that qualify a generic data object as a Train specimen (e.g., trivially, the attribute "type" is set to "Train") in any database instance. Software developers can easily grasp the difference between intension and extension, considering the difference between an SQL query (intension) and the result set of that query against a particular database instance at a given time (extension). After Frege, intensional semantics was researched in an extension of FOL called *modal logic*,[10] which introduces the concept of *possible worlds* as an extensional model that is not fully present at the *agent* that performs the reasoning task, thus allowing a formal account of the interpretation of logic formulas with respect to a set of different situations.

Now, if we view a provider as a system to which a client can ascribe a set of possible states, then for each client we can replace the single FOL interpretation with a set of interpretations, one for each "possible state" in the provider. What really changes in this setting? The reason why logicians and philosophers embraced modal logic during the last century is the possibility to formalize *propositional attitudes*, which can model different ways in which an agent can take formal statements (propositions) into account, with respect to a set of possible worlds to which it has access. For instance, given a formula ϕ of the service description (e.g., informally, "The train Peloritano starts from Palermo"), an agent can *believe* ϕ, or *know* ϕ, or hold ϕ as *possible*, with respect to the set of states of affairs it is able to access, whereas FOL is just suited to talk about the truth of ϕ in a single, fixed, globally accessible situation. Such propositional attitudes are formalized by means of specific second-order operators, which are associated with rules that bind the interpretation of modal statements to the underlying FOL setting. For example, in standard epistemic logic, for an agent A to *know* a proposition ϕ, it is required that ϕ be true in all the worlds accessible by A; whereas to *believe* ϕ does not entail this condition.

Is modal logic useful for characterizing semantics of service-oriented infrastructures and thus, for enhancing interoperability? Certainly, because in the "open world" of loosely coupled and highly dynamic services, there is no unique, globally accessible situation, there are no alternatives: issuing a request to a service requires the client to assume an attitude with respect to the propositions the provider supplies. Nonetheless, even though modeling distributed systems based on modal logic is well assessed in literature,[11] research on the application of this paradigm in the field of service integration in a Web environment is relatively new.[12] Still, clients of distributed services in non-trivial interoperability scenarios do assume modality, and this results, in practice, in implementing policies for the transfer of data from the provider into the client implementation, taking somehow into account the fact that these data reflect one of a set of possible situations. Efforts to let these modal aspects emerge from the shadow of implementations and come out in transparent models are under way. The syntax underlying the recent proposals for ontology sharing and meta-data exchanges is generally capable of representing "higher-order" constructs such as metaclasses. This would allow modal operators to be syntactically represented. In general, however, rich-description logic-based languages like OWL[13] are conceived for a single agent that accesses a unique (albeit open) situation. Meta-properties such as *trust* are usually framed in meta-level characterizations attached with some suitable syntactic glue to "first-order" business models.[14] As for formal semantics concerns, recent studies on formalizing nonfunctional properties of Web services

aim at capturing some modal aspect in the sense outlined here, including temporality and trust.[15]

Note that, in a modal framework, the client's understanding of the provider's meaning of the service description's predicates (e.g., XML elements) can be thought of as a *propositional attitude* (relational mental state connecting a person to a proposition). For instance, a client of services provided by Italian Railways could believe that 'Train' is intended by the provider as a subclass of Artifact, that is, a Physical Object, and thus, if a Train is present at the time t, then each part of it exists at the time t as well.[16] This is not granted once and forever by any transcendental reason; it is just the specific belief of a specific client of Italian Railways' services. In effect, Italian Railways could have used 'Train' to denote the run of a train, thus as a subclass of Event, which if it occurs at t, is not necessarily fully present at t.[17] How are clients provided with means for making reasonably sure that their own beliefs regarding providers are well-founded? Certainly, having services described in a data language in which basic first-order constraints can be expressed is very helpful, but how can "Train as Object" be distinguished from "Train as Event" without means for expressing inclusion dependencies (e.g., "is-a" relations) of the form $Train(x) \rightarrow Object(x)$? In fact, the more logic constraints can be expressed in a standard way, the more clients can check providers' models as valid, and load them in their knowledge base with, at least, some formal assurance. Moreover, formal constraints can help semantic understanding because unclear predicates can be intended in the light of more basic predicates to which they are possibly bound. Logic constraints, however, are not enough: good logic would be quite useless if predicates and axioms were badly chosen. For instance, who could tell that the distinction of Objects from Events is crucial because it is the root of two fundamentally different ways of "being in time" and that it is crucial to put this distinction on top of a dependency inclusion hierarchy? Where is a consistent set of axioms of this kind of basic meaning maintained, and how, if it is given at all, can such semantic foundations be shared? This is where ontologies come into play.

Ontologies

Following Quine, we consider an ontology as any intensional account of what exists in a certain domain, where "to exist" means "to be the value of a bound variable".[18] As such, in a framework like that outlined earlier, any service description is in fact an ontology. Actually, service-description languages such as WSDL allow using the expressiveness of XML schemas to describe atomic and composite data types that can be considered as structures of logic predicates and constraints, which altogether allow specifying a sort of intensional description. However, any provider is entitled to set up its own ontology, and there is no reason to assume consistency or any immanent unifying heuristics, such as, for instance, the similarity of predicate symbols (e.g., XML tags) based on similarities to natural languages. Interpreting someone else's ontology is a conjecture unless a standard set of predicates is adopted, shared, understood, and consistently interpreted by every node in the distributed environment. The availability and acceptance of semantic standards is an enabling condition for widespread adoption of service-oriented infrastructures.[19] For this reason, a large community of people in both research and industry are currently working on this. A large number of industry "content standards" that aim at abstracting the common conceptualization that underlies the industry sector into well-established data dictionaries are already available for use in electronic exchanges within specific communities. Standard business-data dictionaries are often used as semantic backbones to implement hub-and-spoke service integration infrastructures.[20]

It is not the purpose of this article to provide guidance to ontology adoption and development in the context of distributed environments such as service infrastructures; this complex and crucial topic requires extensive treatment. However, the promulgation of Semantic Web standards such as RDF (Resource Description Framework) and OWL has driven important developments in this field, and applications of semantics-oriented XML schemas in Web services are currently undergoing study and experimentation.[21] We do not discuss results of this research here, but it is important to remark that the adoption of these languages, even by means of partial implementations, facilitates the development and the maintenance of broad ontologies in a distributed way. For instance, defining inclusion dependencies in terms of sufficient and necessary membership conditions, as OWL permits, would relieve the burden of developing the conceptualizations as complete structures of primitive predicates,

thus enhancing the coverage of the business semantics. For instance, provided that $CarService\ (x) \rightarrow Service\ (x)$,[22] a concept like the following:

$$TrainWithServices\ (x) \equiv Train\ (x) \wedge \exists y\ Service(y) \wedge hasService(x,y)$$

would allow inferring this inclusion:

$$Train\ (x) \wedge CarService\ (y) \wedge hasService(x,y) \rightarrow TrainWithServices\ (x);$$

hence eliminating the following primitive from the ontology:

$$TrainWithCarService\ (x) \rightarrow TrainWithServices(x).$$

Adopting a solid top-level conceptualization, that is, a small set of basic concepts valid for any domain (e.g., SUMO[23] or DOLCE[24]), mitigates the risk of introducing obscure predicates and constraints that could hardly be interpreted by most clients. For instance, the introduction of a concept like the following:

$$TenMinutesWagon\ (x) \equiv Wagon\ (x) \wedge Duration\ (x,10)$$

could be avoided if just *Wagon* and *Duration* were bound to a top-level ontology in which:

$Duration\ (x,y) \rightarrow Event(x)$
(*time duration is defined for events*)

$Object(x) \rightarrow \neg\, Event(x)$
(*objects are not events*)

$Wagon\ (x) \rightarrow Object(x)$
(*wagons are objects*)

because most of automatic reasoners[25] would be able to detect that in every model:

$\{x \mid \exists y\ Wagon\ (x) \wedge Duration\ (x,y)\} = \varnothing$
(*there aren't temporal wagons*)

In conclusion, as ontologists have pointed out, the reason for investing in the "ontological level" is that by sharing a sound conceptualization "the potential misunderstandings and inconsistencies due to conflicting intended models are reduced."[26] On the other hand, working out sound and complete ontologies is very far from trivial, and acquiring, evaluating, and adopting available ontological resources is not an easy task. Ontological relativism, in addition, disallows the faith in a natural, smooth, and global convergence to a small set of universally accepted ontologies and suggests that, in many business domains, we will probably keep going with many heterogeneous, yet coexisting, conceptual models.

MODELS FOR SEMANTIC INTEROPERABILITY

SOA technologies and standards provide designers with a variety of solutions for semantic interoperability in the sense outlined in the previous sections. In this section, we illustrate the main features of a number of possible approaches to semantic interoperability. The selection of a specific approach depends upon the business environment in which the service infrastructure is deployed. These possible approaches are based on a classification criterion that captures what is essential from the perspective of semantic integration and that addresses the main issues in managing semantics in service-oriented distributed systems.

The different models for semantic interoperability are classified based on two fundamental dimensions: 1) choosing one of two possible ways to set up integration mappings, one in which each service schema is mapped to any other (any-to-any) and another in which each one is mapped to a single schema (any-to-one), and 2) choosing whether the integration logic is executed in a single, distinguished node (centralized) or the execution is distributed among multiple, functionally equivalent nodes (decentralized). By using this classification criterion, we obtain the following four interoperability models.

Any-to-any centralized model

In the *any-to-any centralized* model illustrated in *Figure 3*, services are interpreted and mapped to others without resorting to ontologies. The semantic integration function is provided by a special component, the *integrator*. The integrator creates a model for each participating service based on its own interpretation of the service description. The figure shows that the integrator holds a model for each service (inner ellipses), based on its interpretation of the respective service description. On the other side, providers maintain their own models, realized through their implementations. Mappings, however, are applied directly between pairs of inner

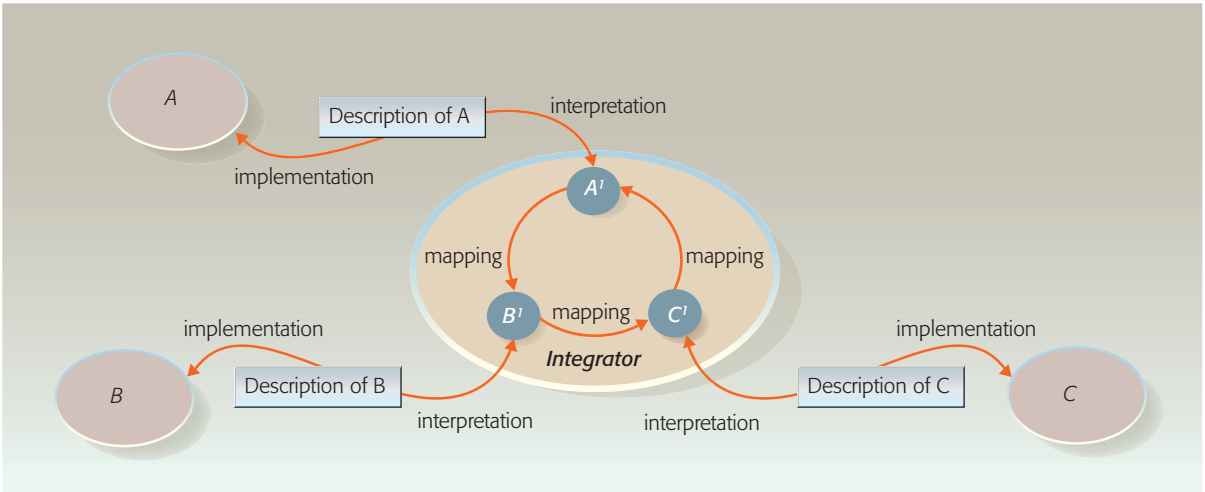

Figure 3
An illustration of the any-to-any centralized model for semantic interoperability

models. Therefore, designers of such applications determine the integration logic by freely mapping input/output data from any service in the system to any other, based on their understanding of what the descriptions made available by service providers actually mean. The any-to-any centralized model is used in the specification of business processes, supported by standardized languages such as BPEL,[27] which are used in mature production environments. We now look at some practical implications of implementing such an integration model.

In any-to-any centralized scenarios, service providers are usually noncommital with respect to integration issues; their contribution to the infrastructure is limited to service interfaces, and they do not make any commitments or accept any constraints to integrate their services in any superseding application. Providers are not involved by designers in making the integration semantically consistent, which means that integrators can freely use the service interfaces without the need for any preliminary or subsequent negotiation with providers. Services are usually atomic, independent, and self-contained; they are not affected by what happens outside their host. This is simultaneously the strength and the limitation of this approach. Even if service providers were willing to cooperate in the integration effort, they would be prevented from doing so by the requirement to provide the full specification of their mapping to any service that the designers plan to include. In practice, to be able to contribute to the integration process, each service provider should receive the integration plan, analyze it, and specify the mappings of its data languages to those of other services, which would require bilateral agreements between all pairs of participants, a cumbersome process indeed.

In fact, in "choreography" applications, that is composing and choreographing business-aligned services, it is not the case that independent service providers can contribute to any shared semantics to any extent. Choreography designers are the only "semantic authorities," and the overall system behavior is the product of their understanding (actually, beliefs) about services, based on available descriptions. In this context, misinterpreting a single service operation call could cause errors in data exchanges that would propagate and eventually affect the entire choreography application. Moreover, unlike the use of libraries in standard software engineering, this kind of error could result in disrupting systems that are in effect supplied by third parties. Also, because semantic errors can be detected only at runtime, and setting up test environments for service infrastructures is not trivial, there is the concrete possibility that such errors might occur at any time during the lifetime of the system with unpredictable consequences. Thus, it is crucial to reduce the risk of semantic misunderstanding as much as possible. As we have seen, the entities that a service provider deals with are not limited in any way; they may include events,

properties, abstractions, and whatever else designers have envisioned as part of the system domain. This envisioning (i.e., ontological commitment) is neither given by nature, nor is it driven by something inherent to the business itself; rather, it is a design choice. Now, either choreography designers can safely assume that the services they are going to integrate in fact share the same set of ontological commitments, or they must find a way to capture the intended meaning of service descriptions and set up a complex web of conceptual mappings.

In "closed" environments, such as large enterprises, service providers may be required to adopt the very same ontology. Of course, this would solve all the problems in advance, and this is in fact the best option whenever available. In this case, however, the team supporting the service infrastructure would actually be involved in a process of ontological standardization, whose impact should be carefully analyzed and managed. Integrating legacy applications and databases, for instance, would raise the nontrivial issue of harmonizing the conceptualizations they convey with the corporate ontology. If the adoption of a standardized ontology is out of the reach, it is still possible that clients and providers belong to tightly bounded organization units, and choreographers have access to the service implementation (for instance, by having access to internal documentation, which presumably contains much more information than the exposed service interface). This gives them the possibility of checking the provider's interpretation of the public interface and to make sure that operations and data are correctly interpreted within the choreography application. This is the ideal scenario for developing service integration with any-to-any centralized models. Unfortunately, this is not the only scenario designers usually face.

What can help choreographers interpret the service specification when there are no standardized ontologies and the only information available is in the published interface? We recall that the measure of semantic consistency in the use of a service is related to how the client's interpretation matches the provider's; a model in the provider's set of possible worlds should be a model for the client as well. In the light of what we have discussed so far, the only heuristic available comes from the description language, with the provision that that language is capable of expressing logic constraints (e.g., inclusion dependencies). A language with constraints would drive the client's interpretation through a grid of descriptive predicates, so as to allow a sort of holistic understanding of each single predicate based on the entire ontology. Unfortunately, the standard data language for SOAs, that is, XML, does not standardize logic modeling constraints, and this is a serious drawback when developing choreography applications in loosely coupled environments. Once again, this reveals the value of semantics-oriented XML extensions like RDFS (RDF Schema) or OWL, but industrial support for these standards is just developing, and their adoption is still limited.

In conclusion, any-to-any centralized integration models, like those used with choreography languages, are well-suited for closed environments such as enterprise integration infrastructures. Adopting this model for integration of nontrivial services in open environments, on the other hand, would require additional constraints and would pose some risk, which has to be estimated and managed.

Any-to-one centralized model

As shown in ***Figure 4***, in the *any-to-one centralized* model service, input/output data are mapped to a single ontology managed by a specialized application or service (the component Integrator in Figure 4). As in Figure 3, inner ellipses show that the integrator holds a model for each service, based on its interpretation of the available descriptions, while providers realize their own models through implementations. Mappings, though, end up at a unique ontology maintained by the integrator.

The ontology (sometimes called "business information model") is an all-inclusive data language that is a semantic superset of the union of all data languages exported by the service providers. This is the model used, for instance, in a variety of content-aware hub-and-spoke systems, such as message brokers in message-driven service architectures, which are capable of translating and routing messages based on their content,[28] and the more general Enterprise Service Bus.[29]

Generally, providers are not involved in developing the logic of integrator nodes; thus, designers of the integrator are in charge of setting up the semantic layer. The semantic consistency of the system

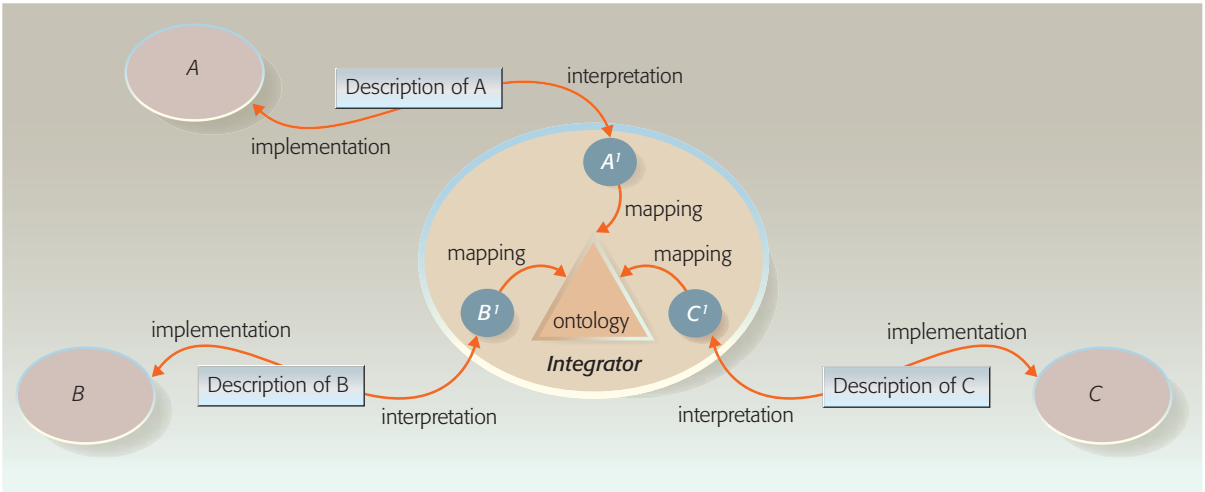

Figure 4
An illustration of the any-to-one centralized model for semantic interoperability

depends upon the designers' understanding and beliefs, and thus remarks about any-to-any centralized models apply here also. In addition, designers are requested to map service descriptions with respect to a single, fixed ontological backbone, which requires additional skills when compared with the task of mapping services case by case, based on local knowledge and ad hoc transformations. Hence, the adoption of business information models as a unified semantic foundation for service-oriented infrastructures results in additional cost, at least in terms of skills requested to roll out the integration logic. Is this cost justified?

It is often remarked that the number of mappings required for the entire system is significantly reduced in any-to-one models, decreasing (in the limit) from $N \times (N-1)$ to N, where N is the number of services involved. However, the reduction in the number of mappings is not the striking difference here. The real difference is in the *existence* of a business model. In fact, whereas any-to-any applications let semantics scatter in a web of pair-wise mappings, any-to-one models make semantics a tangible and manageable object that consists of an ontology of the business domain plus a set of mappings that translate providers' service descriptions in terms of the business model.

We have briefly discussed the impact of adopting ontologies in the previous section. In any-to-one interoperability models, the *completeness* of the ontology is the main issue to address; that is, how do we make sure, in advance, that the business model adopted will contain anything that service definitions encompass? If there are no limitations to the number and variety of the services to be integrated, of course, this is not generally possible, and in fact these systems must include some suitable means to extend the ontology to accommodate new business entities. Extensibility of the business model is therefore key to the success of this kind of integration pattern. *Property-centric* ontology languages such as those based on RDF Schema allow this feature to be implemented properly.

Integration of data in this model can be implemented in one of three ways.[7] In the local-as-view (LAV) approach, mappings proceed from source schemas to the global view (i.e., the ontology) by associating single data-alphabet elements exported by services to views (queries) over the business model. In the global-as-view (GAV) approach the reverse holds; that is, queries on the data alphabet of sources are mapped to single elements of the ontology. A third approach, GLAV (global-local-as-view), includes elements of both modes and is the typical pattern in data-exchange systems.[30]

For instance, suppose that the service provider Italian Railways exports the predicate *Ticket* (*train, from, to, class, price*) to a system whose business model contains *Route* (*ID, start, end*) and *Fare* (*route, means, amount*) (see Figure 2; notice

however that the model type depicted here could be either any-to-any or any-to-one, depending on whether the broker uses ontologies or not). A LAV mapping could be:

$$\textit{Ticket}\ (\_, f, t, \_, p) \rightsquigarrow \exists r\ \textit{Route}\ (r, f, t) \wedge \textit{Fare}\ (r, \text{'train'}, p).$$

Notice, in this example, that some of the attributes that characterize *Ticket* are missing in the translation (namely, those referring to the concept of 'train'), while the value of the *Route ID* should be injected in the data transfer. In fact, the LAV approach is valuable if the integration system is based on a stable and complete ontology; otherwise, much information provided by the source could be lost. On the other hand, services are allowed flexibility in entering or leaving the infrastructure; having a new service enter the system would result in enriching the mapping set without changing the business model, provided that for each item in the service data language a suitable mapping on the global schema can be established.

In data integration systems, answering queries formulated on the global schema by using LAV mappings is notoriously difficult because the global view is characterized by potentially complex mappings to the available data sources, and inferring how to use that characterization in order to answer global queries may be nontrivial in many cases. Using a business information model to support data exchange algorithms that translate messages coming from a mapped service to messages directed to other mapped services could reveal the same kind of complexity; consider, for instance, the problem of deciding whether to route to Italian Railways, as described in Figure 2, a request message originating at Grand Tour Agency with content like:

$$\textit{GetRoute}\ (x, \text{'Palermo'}, \text{'Roma'}) \wedge \textit{Fare}\ (x, \text{'land transport'}, y)$$

For a broker this decision would require determining that 'train' is a kind of 'land transport', and therefore, Italian Railways is eligible as a data source. Depending on the expressiveness of the ontology language, a decision like this could involve complex algorithms, and the brokering system could therefore be affected by scalability problems. Of course, simple one-to-one correspondences of local and global properties could be computed at a reasonable cost, but a poor mapping language would probably require the business model to be continuously expanded and adapted to data languages exported by services, with a severe decline of flexibility.

The GAV approach in a service integration system would work the other way around: concepts of the business model would be mapped to views of the source schemas. With reference to Figure 2, provided that the broker's schema coincide with the ontology, a GAV mapping with the agency's schema would look like:

$$\textit{Itinerary}\ (x) \wedge \text{In}\ (x,a) \wedge \text{In}\ (x,b) \rightsquigarrow \exists r\ \textit{Route}\ (r, a, b)$$

Algorithms such as those needed to calculate content-based routing would be much simpler if integration systems were limited to this kind of mapping, because it requires specifying the exact way to get (put) instances of global concepts out of (into) the available sources. We notice, however, that GAV mappings are not well-suited for the smooth integration of unforeseen data sources into a given global business model in dynamic environments, as typically requested in service infrastructures. In fact, in this approach, the business model comes first, and sources (i.e., services) are generally regarded as a means to provide it with a suitable extension. Mixing GAV with LAV results in the so-called GLAV approach, where views on the local sources can be mapped to views on the ontology. This kind of mapping is the most powerful and general one, but, of course, requires at least the complexity associated with the LAV approach.

In summary, any-to-one centralized integration models are powerful and well-suited for managing complex and dynamic environments, such as Web services in business-to-business scenarios. Nevertheless, this approach requires careful studying of the trade-offs between expressiveness of ontology languages, generality of schema mappings, and computability of integration algorithms. Solid any-to-one centralized solutions should be based on high-quality ontologies and expressive mapping languages, but the resulting ability to efficiently perform complex reasoning tasks is tied to limits on the scalability of the integration system.

Any-to-any decentralized model

As ***Figure 5*** illustrates, *any-to-any decentralized* systems are those where services manage their

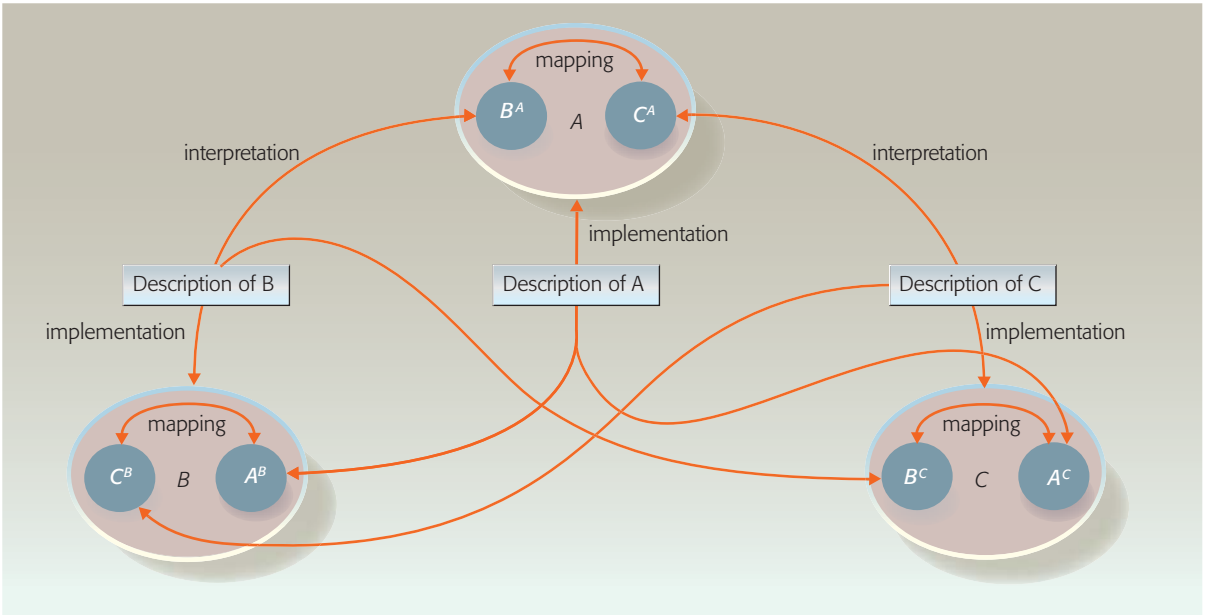

Figure 5
An illustration of the any-to-any decentralized model for semantic interoperability

integration with each other without resorting to any overall business model or integrator component. For this reason, we could refer to them as pure peer-to-peer systems. In this context, 'pure' means not only that each node in the network is potentially both provider and client, but also that there is no common information model or use of a centralized integration service. We notice that, whereas most peer-to-peer systems are very far from pure, basic Web Services infrastructures fall exactly in this category, because the only common service that providers are supposed to use is a very simple kind of publishing repository that is not involved in the runtime operation of services. Figure 5 depicts three peers, A, B, and C, each holding its own model of the other two (nested ellipses). Integration logic is distributed among peers and is accomplished by mapping models to one another.

In pure peer-to-peer systems, as in standard choreography applications, the integration consists in a web of pair-wise mapping statements, which potentially bind the state of each provider with the state of the other. As in any-to-any centralized models, providers do not maintain any global semantics, which in general makes the peer-to-peer approach robust and scalable. Unlike choreographies, however, mappings in peer-to-peer systems are not established by a centralized application; instead, they are distributed across the network, and algorithms that execute the integration logic are distributed as well. Thus, in a certain sense, any service can contribute to a sort of *emergent semantics*.[31] Nobody has the entire set of mappings under control; therefore, cyclic mapping may occur, causing the integration system to be susceptible to undetectable loops, if not properly designed.[32]

From a theoretical perspective, data integration in pure peer-to-peer systems has been shown to require modal logic (in the sense illustrated in the section "Modal semantics") to better suit a number of requirements such as modularity, generality, and tractability.[33] In fact, integrating a set of independent and isolated providers by using a first-order setting would require a supervisor agent to make sure, at least, that mapping axioms are consistent. But supervising the integration logic would detract from the peer-to-peer advantages, because it would require each provider to cooperate with the infrastructure on a global level, thus participating in a tight federation, with a resulting decrease in robustness, scalability, and flexibility. On the other hand, an integration framework capable of dealing with the concept of "possible world" would be able to handle inconsistencies, cycles, unreliable sources,

and all the other problems that may occur within an uncontrolled, self-organizing environment without imposing central authorities, yet making integration computationally tractable.

As we have seen, an intensional framework is one in which each service is modeled as an agent that contributes to the set of possible worlds accessible by clients. A mapping is not interpreted as material implications of FOL formulas,[34] but as the transfer of information coming from the source into the client's knowledge, according to a *propositional attitude* of the client toward the source. In practice, such an attitude is a policy that rules the way a client gathers information from providers, taking into account that they reflect a set of coexisting models. By formalizing data transfer policies, a system can deal with any kind of information provider, whether reliable and well-understood or uncertain and obscure, by introducing explicit rules for trust, preferences, temporal reasoning, and so on.[35]

Infrastructures of decentralized services also need a way to handle "vicious circles," which may be introduced by the uncontrolled distribution and proliferation of mappings in the system. This requires all peers adopting a rule that allows identifying transactions in the entire system, thus avoiding request messages that are reprocessed by the same nodes that created them. This can be easily implemented in special service-oriented environments, such as grids, where distributed transactions can be uniformly identified.[36] Technical problems apart, it is noticeable how intensional reasoning helps cope with mapping cycles in a sound theoretical framework, thus allowing integration algorithms to be designed in a decentralized way.[33]

To conclude, peer-to-peer semantic interoperability, more than an option, is the only model available when the possibility of sharing an ontology or resorting to centralized integration services does not exist. Even though service infrastructures are peer to peer by nature, comprehensive frameworks for such models are still at an experimental stage. In any case, designers of such kinds of infrastructures face inherent uncertainty due to the fact that semantics are distributed in systems that are strongly isolated from one another. Some hints on how this uncertainty can be modeled with the conceptual tools of modal logic have been provided here.

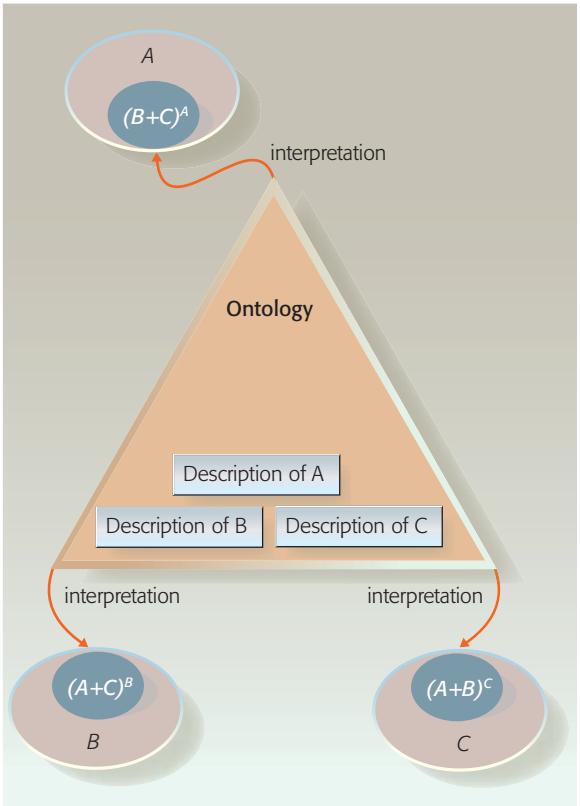

Figure 6
An illustration of the any-to-one decentralized model for semantic interoperability

Any-to-one decentralized model

As illustrated in ***Figure 6***, *any-to-one decentralized* systems are peer-to-peer infrastructures in which endpoints directly connect and exchange information with one another, but service descriptions are mapped to (or directly taken from) a common shared business model. Figure 6 shows how all the service descriptions actually belong to a shared ontology, and each service models the other ones with interpretations of ontology fragments that correspond to the union of the their schemas (nested ellipses).

Shared ontologies can be supplied by specific metadata services, or simply provided as network-addressable vocabularies (e.g., XML Schema documents). Actually, this is the most common way to address the semantic layer in Web-based peer-to-peer architectures, as viewed in the Semantic Web community. Proposals such as Web Services Modeling Framework, for instance, focus on this model, and the authors state that "[the use of ontologies]

ensures consistency of the textual representation of the concept exchanged and allows the same interpretation of concepts by all trading partners involved."[37] We observe that, as the quotation informs, ontologies allow, but do not ensure, the same interpretation of concepts, for the reasons we have outlined in the previous sections.

With respect to any-to-one centralized systems like those based on brokering services, decentralization requires providers to strictly cooperate with the infrastructure as a whole, and to adopt a *lingua franca* that may significantly differ from their inner legacy conceptualizations, and therefore involves a certain adaptation effort. On the other hand, whereas centralized integration entitles a single developer (or a few of them) to set up the overall semantics, the only semantic authorities of decentralized integration models are providers themselves. In fact, even if adopting well-established business models, the way each provider interprets the shared vocabulary remains in the shadow of service implementation, which is where semantics dwells. The distribution of *semantic responsibilities*, which is the distinguishing feature of peer-to-peer systems, is the crucial factor in these scenarios and should be carefully considered. We observe that, whereas centralized applications are based on explicit decisions about how service descriptions exposed by independent providers map to each other—or correspond to some reference ontology—decentralized models require semantic decisions to be made independently throughout the network. Nevertheless, as a rule, these decisions are not assessed by any global authority, and the process of converging to a uniform semantics can be influenced but not controlled. After all, in a completely decentralized environment where no authority is established, who could tell that a service provider misinterpreted a descriptive predicate of a shared ontology?

For this reason, coping with an inevitable *relativism* is the main issue when adopting shared conceptualizations in decentralized environments. As we outlined in the previous sections, having business models specified in a sound and rich ontology language and having them based on a suitable foundational layer reduces the risk of misinterpretations injected by agents when mapping their own conceptualizations to such models. A suitable coverage of primitive concepts with respect to the business domain (completeness) that entails the possibility to progressively enhance conceptual schemas (extensibility) are also key factors of success in the adoption of shared ontologies within service-oriented infrastructures. But still there is the risk of inaccuracy, misunderstandings, errors, approximations, lack of knowledge, or even malicious intent that need to be considered, especially in uncontrolled, loosely coupled environments. From the previous sections, we have learned that modeling these factors requires the adoption of frameworks capable of managing modal policies.

In conclusion, although the any-to-one decentralized integration is a viable solution for handling highly dynamic, peer-to-peer service infrastructures with a minimum level of centralization, it requires the nontrivial task of making all participants compliant with a common conceptualization. To deploy this model, the availability of well-founded and broad ontologies is necessary but not sufficient. In fact, it is crucial to manage specific policies that allow dealing with the uncertainty that is always associated with the semantics of data coming from external information sources.

CONCLUSION

On the way to information systems integration in an ever growing distributed world, developers come face to face with one of the earliest and most venerable disciplines: semantics. SOAs, armed with suitable descriptive languages and powerful reasoning algorithms, provide a solid and standardized basis that facilitates the design and implementation of semantics-enabled IT infrastructures. Nevertheless, at the current state of the art, although service-oriented infrastructures are rapidly becoming a reality, the adoption of specific frameworks to address the semantic layer is still at an early stage. Many kinds of difficulties hinder the acceptance of semantic-oriented artifacts, technologies, methodologies, practices, and standards. Above all, we believe that the lack of awareness of what semantics is, why it is important, and how it could be modeled constitute the most significant obstacles in its application to the semantic layer of service-oriented infrastructures.

In this paper, we have provided a broad overview of the issue of semantic interoperability in service-oriented infrastructures. We have given a specific definition of what a semantic characterization of

services is supposed to be, both in an extensional and intensional setting, and we have explained the role of ontologies. Then, we have analyzed four basic models for semantic interoperability that depend on the way mappings between service descriptions and ontologies are drawn and where the integration logic is evaluated. Selecting one of those four models depends upon nonfunctional constraints, such as organizational boundaries, availability of semantic standards, and so on. We have provided some advice on what designers should be focusing on when implementing the semantic layer in the range of the possible interoperability models.

In conclusion, semantics is not an easy discipline, as specialists in this field know very well. Generally, this is due to the arbitrary, problematical, and concealed nature of interpretation. As we might expect, this is reflected in the way semantics can be dealt with in complex distributed information systems, and this explains why semantic interoperability is so often neglected. Since the very beginning, and particularly with the expansion of the Web, developers and researchers in industries and universities have been tackling semantics based on the state of the art of our theories on meaning, and research has continuously improved methods, standards, technologies, and solutions. Nevertheless, there are no "killer solutions" or "silver bullets" available—by virtue of pure technological advances—to put in place a semantic layer. The quality of semantic interoperability in distributed systems largely depends on concrete theories of meaning that humans provide, exchange, understand, and reconcile, as well as concrete processes that humans execute in order to achieve semantic agreements. Understanding of semantics can help developers of service-oriented infrastructures to deliver better solutions. We hope this paper will help achieve this understanding.

CITED REFERENCES AND NOTES

1. An ontology is an account of what "there is" in a certain domain of discourse.
2. RosettaNet, http://www.rosettanet.org.
3. See, for instance, S. Colucci, T. Di Noia, E. Di Sciascio, F. M. Donini, and M. Mongiello,"Description Logics Approach: Logic Based Approach to Semantic Matching of Web Services," *Journal of Computing and Information Technology* **11**, No. 3, 217–223 (2003).
4. T. J. Taylor, *Mutual Misunderstanding: Communicational Scepticism and the Theorizing of Language and Interpretation*, Duke University Press, Durham, N.C. and Routledge, London (1992).
5. W. V. O. Quine, *Ontological Relativity and Other Essays*, Columbia University Press, New York (1969).
6. See the Semantic Web site at W3C: http://www.w3.org/2001/sw/. A useful discussion of this topic is in: M. Uschold,"Where are the Semantics in the Semantic Web?" *AI Magazine* **24**, No. 3, 25–36 (September 2003).
7. M. Lenzerini, "Data Integration: A Theoretical Perspective," *Proceedings of the 21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, PODS 2002, Madison, Wisconsin (2002), 233–246.
8. It is normally understood that symbols of the operational alphabet are mapped to access and manipulation methods in the service implementation.
9. F. L. G. Frege, "On Sense and Reference," *Translations from the Philosophical Writings of G. Frege*, Third Edition, P. Geach and M. Black, Editor and Translator, Blackwell, Oxford (1980).
10. See J. Hintikka, *Knowledge and Belief*, Cornell University Press, Ithaca, NY (1962) and S. Kripke, *Semantical Analysis of Model Logic,* Zeitschrift für Mathematische Logik und Grundlagen der Mathematik 9, 67–96 (1963).
11. R. Fagin, J. Y. Halpern, Y. Moses, and and M. Y. Vardi, *Reasoning about Knowledge*, MIT Press, Cambridge, MA (1993).
12. T. Finin and and A. Joshi, "Agents, Trust, and Information Access on the Semantic Web," *ACM SIGMOD Record*, ACM, New York (December 2002).
13. Web Ontology Language (OWL), http://www.w3.org/2004/OWL/.
14. Semantic Web Trust and Security Resource Guide, http://www.wiwiss.fu-berlin.de/suhl/bizer/SWTSGuide/.
15. J. O'Sullivan, D. Edmond, and A. H. M. ter Hofstede, "Formal Description of Nonfunctional Service Properties," Technical Report, Queensland University of Technology (February 9, 2005), http://www.citi.qut.edu.au/about/research_pubs/technical/non-functional.jsp.
16. We refer here to the notion of "continuant" as given in P. Simons, *Parts: A Study in Ontology*, Clarendon Press, Oxford (1987).
17. Once again we refer to the notion of ontological relativism in W. V. O. Quine, Reference 5.
18. W. V. O. Quine, *Word and Object*, The MIT Press, Cambridge, MA (1960).
19. J. Jacobs and A. Linden, "Semantic Web Technologies Take Middleware to Next Level," *Research Note T-17–5338,* Gartner, Inc. (August 20, 2002).
20. Siebel Business Integration Common Objects, Siebel Systems, Inc., http://www.siebel.com/.
21. W3C Semantic Web Services Interest Group, http://www.w3.org/2002/ws/swsig/.
22. Unless specified otherwise, variables are universally quantified.
23. Suggested Upper Merged Ontology (SUMO), IEEE Standard Upper Ontology Working Group, http://suo.ieee.org/.
24. Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE), WonderWeb Foundational Ontology Library, http://wonderweb.semanticweb.org/.
25. See, for instance, I. Horrocks, "FaCT and iFaCT," *Proceedings of the International Workshop on Description*

Logics (DL'99) (1999), pp. 133–135, and V. Haarslev and R. Moeller,"Description of the Racer System and Its Applications," *Proceedings of the International Workshop in Description Logics (DL2001)* (2001), pp. 132–142.

26. N. Guarino, "The Ontological Level," in R. Casati, B. Smith and G. White, Editors, *Philosophy and the Cognitive Sciences*, Hölder-Pichler-Tempsky, Vienna (1994), pp. 443–456.
27. Business Process Execution Language for Web Services, IBM Corporation, http://www.ibm.com/developerworks/library/ws-bpel/.
28. WebSphere Business Integrator Message Broker, IBM Corporation, http://www.ibm.com/software/integration/wbimessagebroker/.
29. *Patterns: Implementing an SOA Using an Enterprise Service Bus,* IBM RedBook SG24-6346, IBM Corporation (2004).
30. R. Fagin, P. G. Kolaitis, and L. Popa, "Data Exchange, Getting to the Core," *Proceedings of ACM Symposium of Principles of Database Systems,* ACM, New York (2003), pp. 90–101.
31. K. Aberer, T. Catarci, P. Cudré-Mauroux, T. Dillon, S. Grimm, M. Hacid, A. Illarramendi, M. Jarrar, V. Kashyap, M. Mecella, E. Mena, E. J. Neuhold, A. M. Ouksel, T. Risse, M. Scannapieco, F. Saltor, L. de Santis, S. Spaccapietra, S. Staab, R. Studer, and O. De Troyer, "Emergent Semantics Systems," *Proceedings of the First International Conference on Semantics of a Networked World (CSNW04)*, Paris, France (2004).
32. A. Halevy, Z. Ives, D. Suciu, and I. Tatarinov, "Schema Mediation in Peer Data Management Systems," *Proceedings of 19th International Conference on Data Engineering (ICDE)* 2003, pp. 505–516.
33. D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati, "Logical Foundations of Peer-To-Peer Data Integration," *Proceedings of the 23rd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 2004),* ACM, New York (2004), pp. 241–251.
34. These are if-then formulas, such that if the premise is true, then the consequence is true as well.
35. J. Carter and and A. A. Ghorbani, "Towards a Formalization of Trust," *Web Intelligence and Agent Systems* **2**, No. 3, 167–183, (March 2004).
36. Grid data integration based on epistemic logic is illustrated in D. Calvanese et al.,"Hyper: A Framework for Peer-to-Peer Data Integration on Grids," *Proceedings of the International Conference on Semantics of a Networked World: Semantics for Grid Databases (ICSNW 2004)*, pp. 144–157.
37. D. Fensel and C. Bussler, *The Web Service Modeling Framework,* http://informatik.uibk.ac.at/users/c70385/wese/wsmf.bis2002.pdf, (extended abstract).

Accepted for publication June 27, 2005.
Published online October 27, 2005.

Guido Vetere
IBM IGS Italy, Via Sciangi 53, 00144, Rome, Italy (gvetere@it.ibm.com). Mr. Vetere is Research Coordinator at the IBM Center for Advanced Studies, Rome, Italy. He obtained a degree in philosophy of language at Rome University, with thesis work in computational linguistics. After a postdoctoral appointment in machine learning and artificial intelligence at the IBM Rome Scientific Center, he participated in several research and development projects in expert systems, knowledge management, and object-oriented techniques, including OntoSeek, one of the first applications of linguistic ontologies in Web-based information retrieval. He is currently involved in a number of projects related to the semantic Web, data integration, and service-oriented architectures. He has been on program committees of international conferences related to Web Services and ontologies (ISWC 2003, WISE 2003, ODBASE 2004), and sits on the advisory boards of European research programs such as OntoWeb and WonderWeb. He has been collaborating with Prof. Lenzerini's team at the University of Rome on Hyper, a framework for peer-to-peer data integration on grids, an IBM Shared University Research program.

Maurizio Lenzerini
Università degli Studi di Roma "La Sapienza," Via Salaria 113, I-00198, Rome, Italy (lenzerini@dis.uniroma1.it). Professor Lenzerini, a full professor in computer science and engineering at Rome University (La Sapienza), joined the university in 1983 and is now leading a group involved in research in software engineering, databases, and artificial intelligence. He is the author of several textbooks on computer science, software engineering, and databases. His main research interests are in data modeling, information integration, knowledge representation and reasoning, object-oriented methodologies, and service-oriented architectures. He is currently involved in national and international research projects on ontology representation and reasoning, information integration, and e-service modeling and composition. He is the author of more than 250 publications in conference proceedings and journals, including *Journal of Computer and System Science*, *Information and Computation*, *Artificial Intelligence*, *Information Systems*, *IEEE Data and Knowledge Engineering*, *ACM-PODS*, *ACM-SIGMOD*, *LICS*, *IEEE-ICDE*, *VLDB*, *ICDT*, *IJCAI*, *AAAI*, *KR*, and *CoopIS*. He is the editor of several books, including the recent *Data Warehouse Quality*. He is a regular member of program committees of international conferences such as IJCAI, AAAI, EDBT, PODS, KR, CoopIS, ER, and ICDT. He has been an invited speaker at many conferences, has organized several international conferences and workshops, and is a member of the editorial board of various international journals. He is the editor of *Information Systems: An International Journal*, for the areas of data modeling, knowledge representation, and reasoning. He was Program Co-Chair of the 4th International Conference on Cooperative Information Systems (1999), of the International Conference on Conceptual Modeling (1999), and of the 9th International Conference on Database Theory (2003). ■