Semantically Meaningful Data Exchange in Loosely Coupled Environments

C. Bornhövd, A. P. Buchmann
Dept. of Computer Science, Darmstadt University of Technology
64283 Darmstadt, Germany

ABSTRACT

When the Internet is used as a global infrastructure for data exchange between autonomous participants, the question of what a data object really means becomes crucial. Unfortunately, many of the underlying assumptions about the meaning of a given data object are given only implicitly. Thus, for a semantically meaningful data exchange, we need to make these assumptions explicit by providing semantic metadata. To be of any use this metadata has to be based on commonly agreed upon vocabularies. In this paper we present a framework for a meaningful data exchange between previously unknown participants, and the integration of data from different providers. It is based on the use of shared vocabularies, or ontologies, as a common interpretation basis for data and metadata. We present a fairly simple representation for these vocabularies which uses Java as a description language.

Keywords: semantic metadata, ontology, data integration

1. INTRODUCTION

The Internet may be seen as a global information carrier used by consumers and providers to exchange data from a wide range of topics. However, the combination of data from independent, perhaps previously unknown participants, is problematic because of heterogeneities in structure and meaning of the data. Thus, we need to map available data to a common representation model for further electronic processing. In many business-to-business applications the overhead of point-and-click interaction outweighs the cost of this mapping.

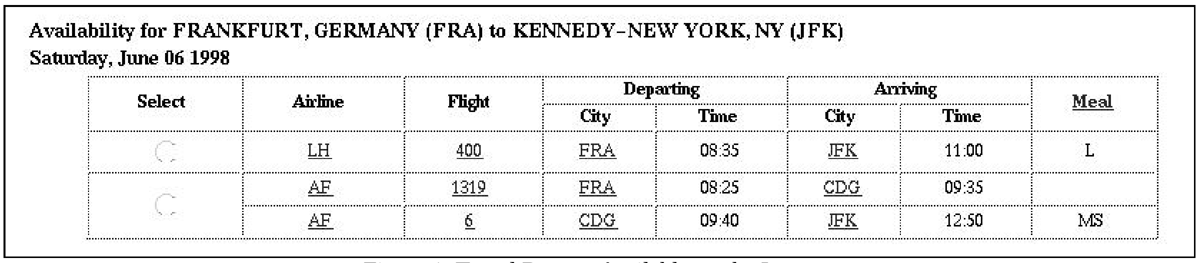
In addition, many sources offer data in semistructured form, such as SGML/XML, or HTML documents. They provide no explicitly specified schema on which meaningful data processing can be based. Even if data is made available via relational or object-oriented database systems many of the underlying modeling assumptions are only given implicitly, that is, they are in the minds of the designer, are specified in text documents not available externally, or are reflected in local applications. This context information [15] is lost when data is exchanged across institutional boundaries. Thus, to exchange and process data from independent participants in a semantically meaningful way, we need explicit information about its intended meaning. We use semantic metadata to represent this additional information.

For example, travel information is made available by a travel agency as HTML pages depicted in Figure 1. Given this data, it is not clear what terms exactly mean, what time format is used – is flight LH 400 to take off at 8:35 in the morning, or 20:35 in the evening, what do the abbreviations used for *Meal* mean, and so on. Without explicit information about these implicit assumptions the available data is of limited use.

To describe context information in an unambiguous way, we use domain-specific ontologies [18, 16]. An ontology provides an agreement about a shared conceptualization of a given subject domain [10, 11]. The concepts specified in the ontology provide a common vocabulary for which no further negotiation concerning their meaning is necessary. In addition, the ontology provides information about the representation of the data described on the basis of the model. In this way, the ontology can serve as a common basis for the interpretation of context information as metadata, and thus enable a semantically meaningful data exchange.

In an ideal situation, all participants that make use of data and metadata from a given domain should adhere to the corresponding ontology. In an imperfect real world, the model must be extensible to allow ontologies on the consumer side tailored to specific needs. Ontologies should follow existing description standards (such as SI standard units, or EDIFACT [13] for travel information) as much as possible to enhance acceptance by other participants. Domains for which no such standards exist require new concepts to be specified. Depending on the subject domain at hand, this can be done following a top-down approach as proposed in [5], or a bottom-up approach as introduced in [20]. By providing a way to add metadata and extend the ontology, we believe that we can claim a reasonable combination of rigor and flexibility that makes our model applicable in many real-life situations.

In the next section we present an ontology-based framework that supports semantically meaningful data exchange. In Section 3 we show how ontology concepts are represented using the Java programming language. Section 4 describes how domain-specific ontologies are structured and organized in MIBIA. Section 5 gives a short comparison of our system with existing prototypes. Finally, Section 6 provides conclusions.



Availability for FRANKFURT, GERMANY (FRA) to KENNEDY–NEW YORK, NY (JFK)
Saturday, June 06 1998

Select	Airline	Flight	Departing		Arriving		Meal
			City	Time	City	Time	
○	LH	400	FRA	08:35	JFK	11:00	L
○	AF	1319	FRA	08:25	CDG	09:35	
	AF	6	CDG	09:40	JFK	12:50	MS

Figure 1. Travel Data as Available on the Internet

2. THE MIBIA FRAMEWORK

We have implemented an ontology-based Java framework, called MIBIA (*MIX Based Integration Architecture*), that provides a platform for a semantically meaningful data exchange, and the integration of data from different providers. Implicit assumptions about the meaning of the data to be exchanged are made explicit by mapping it to a common representation model called MIX (*Metadata based Integration model for data X-change*) [3]. MIX is a self-describing data model since information about the structure and semantics of the data is given as part of the available data itself, thus allowing a flexible association of context information in the form of metadata.

MIX is based on the concept of a *semantic object*. A semantic object represents a data item together with its underlying *semantic context*, which consists of a variable set of meta-attributes (also represented as semantic objects) that explicitly describe implicit modeling assumptions.

In addition, each semantic object has a *concept label* associated with it that specifies the relationship between the object and the real world aspect it describes. These concept labels are taken from a commonly known ontology. Thus, the concept label and the semantic context of a semantic object help to describe the supposed meaning of the data. An example of how data is represented using MIX is given in Figure 3.

Our framework follows the classical mediator approach introduced in [22], and is shown in Figure 2. For the time being, components such as federation manager, and wrapper and ontology servers are colocated with the application processing the data. However, it is feasible to view these as generic Internet services that may be located remotely in the future.

The bottom layer of the architecture consists of autonomous data sources that provide data from a common subject domain, without sacrificing their autonomy. The available data sources can be of different types, as long as a wrapping component that maps local data to the common representation model is provided. The current prototype provides mapping components for relational databases and XML documents.

Data wrappers map local data structures and terms of a source to concepts specified in the common ontology, and add context information. This makes explicit the relationships between data of a source and the real world aspects being modeled. Thus, heterogeneities in the organization and the terms used are resolved, and differences in the underlying semantics are made explicit as far as possible. Because the meaning of the data is usually known only locally, it is preferable if the mappings are specified by the institution owning the data. If the interpretation is not provided at the source, the receiver can cast his or her own interpretation as metadata for future use.

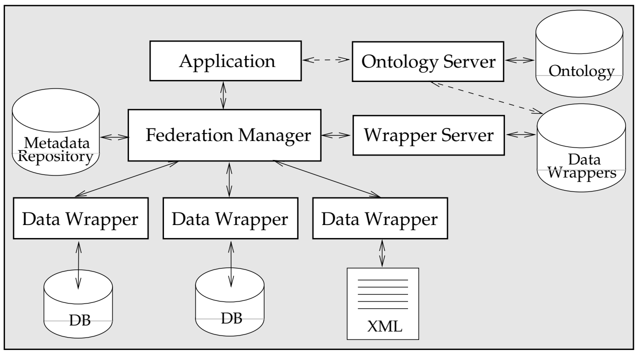


Figure 2. The MIBIA Framework

In MIBIA, data wrappers are implemented as Java classes that are registered with the federation manager and can be loaded by it to transfer data from the corresponding source. Since data wrappers are instantiated on the same machine as the federation manager, semantic metadata is added on the machine of the federation manager and does not need to be transferred from the machine on which the data resides.

By dynamically loading wrapper classes, it is possible for the federation manager to keep only those classes in memory it actually needs to answer a data request. Wrapper classes not available can be loaded from a designated wrapper server and are cached at the federation manager. This allows for the flexible management and use of a huge number of data sources that may change frequently.

A ***wrapper server*** manages the wrapper classes for a set of data sources. Because the federation manager can download wrappers from wrapper servers, the storage and management burden of the federation manager is lowered. Keeping all wrapper classes for a given source at the same wrapper server simplifies their use by different federation managers, and eases wrapper maintenance because there is only a single copy of each wrapper class.

The ***federation manager*** manages the available data sources by keeping a ***metadata repository***. This repository includes information about the concepts for which a given source provides data, as well as access information for this data, i.e., information about the required wrappers. When it receives a data request, the federation manager uses this information to load the appropriate wrapper classes and route requests to them. To lower access costs, the federation manager caches all recently used wrapper classes in main memory.

The wrapper classes return semantic objects represented on a common description basis, and with additional context information to the federation manager. However, these objects may still be, dependent on the respective source, represented with relation to different semantic contexts, i.e., on the basis of different units of measure, derivation formulas, coding conventions, or naming schemas. Therefore, based on the explicit description of the underlying context the federation manager tries to resolve these semantic heterogeneities by converting the data to a common context using appropriate conversion functions. This common context can be specified by the receiver of the data; see [3] for details of the integration process.

The federation manager then returns the unified semantic objects in the form of Java objects. The application views the available data on the level of concepts from a domain-specific ontology without being aware of their local organization.

An ***ontology server*** stores and manages the domain-specific vocabulary, or *ontology*, underlying the federation. An ontology describes a single domain and provides a way to describe the concepts in that ontology independent of any application or data source. Concepts are given as precompiled Java classes (see Section 3) that can be downloaded and used by wrappers and applications.

The common vocabulary provides the extensible description basis to which data providers and consumers refer. On this basis an explicit description of the meaning of the data is given by mapping the local data to appropriate concept classes. A mapping of data objects from different sources to the same ontology concepts makes clear that they represent objects of the same class of real world phenomena. In contrast, semantic differences are made explicit by mapping data objects to different concepts whose semantic discrepancies are described in the ontology.

If no corresponding concepts exist in the vocabulary, new concepts have to be introduced by the corresponding data source or the consumer of the data, and must be registered with the ontology server. To ensure the consistency of the ontology, new concepts have to be introduced by extending or specializing existing concepts in a predefined way to avoid ambiguous specifications or homonyms. The consistency of a given concept specification is ensured through the language constructs of Java.

To request data, an ***application*** builds an SQL-like query using concepts of the common ontology, and passes it to the federation manager. The federation manager returns the requested information in the form of semantic objects, i.e., objects augmented with metadata giving additional information about their meaning. The application can then use these objects without any further processing.

Wrappers and application programs are connected to the ontology at compile time by using a specific pre-compiler. This tool analyzes import statements of the given Java classes and loads the necessary concept classes into the respective local directory path.

3. REPRESENTING ONTOLOGIES IN JAVA

The use of ontologies as a common vocabulary for metadata, i.e., as “meta-metadata”, requires an agreed upon description basis on which ontologies can be specified. If no such description basis exists we would need additional information for the interpretation of the available ontology. Therefore, all participants should refer to the same specification language as a common description basis for the specification of ontology concepts and their relationships. Such a common specification language determines fundamental language constructs for which no further negotiation concerning their syntax and semantics is necessary.

Most specification languages used in AI may be broadly classified as entity-relationship- and frame-slot-based languages. *Entity-relationship-based* languages depend on entities and their attributes to specify concepts. Semantic dependencies between different concepts are described by introducing relationships between entities. In the case of *frame-slot-based* languages, concepts and the relationships between them are represented as so called *frames*. Different aspects of a concept are specified as *slots* of the corresponding frame. Even if the underlying language constructs are relatively simple, ontologies specified using languages such as KL-ONE, LOOM, KIF, or CLASSIC become quite complex, and in general are difficult to understand and use by an inexpert user.

Specification languages in AI, which could be used in MIBIA, contain many features that are needed for inferencing support. Because in MIBIA, determining what concept a given piece of data represents is most important and support of inferencing is of minor relevance, such languages are needlessly complex and we can use a different, simpler specification language.

For this reason, we use Java as the specification language for ontologies. Because of its object-oriented language concepts Java provides a description basis well suited for concepts and their relationships.

Representing Ontology Concepts

In our common representation model, MIX, we distinguish between simple and complex semantic objects. *Simple* semantic objects represent atomic data items, such as simple number values or character strings, with its underlying *semantic context*. In contrast, *complex* semantic objects can be understood as heterogeneous collections of semantic objects, each of which describes exactly one attribute of the represented phenomenon. These subobjects are grouped under a corresponding ontology concept. The semantic context of complex semantic objects is determined by the sum of the context information of their subobjects.

The attributes of a complex semantic object are divided into those used, similar to a set of key attributes in the relational model, to identify an object of a given concept, and additional attributes that might not be given for all objects of the concept. Identifying attributes make it possible to determine if two semantic objects represent the same real world phenomenon.

```
< OneWayFlight, {
 < FlightSegment, {
  < FlightNumber, 400 >,
  < AirlineIdentifier, "LH",          {<AirlineCode, "TwoLetterCode">} >,
  < DepartureDate, "Jun 06 1998",{<DateFormat, "Mon DD YYYY">} >,
  < DepartureTime, "08:35",           {<TimeFormat, "HH:MM">} >,
  < DepartureAirport, "FRA",          {<AirportCode, "ThreeLetterCode">} >,
  < ArrivalAirport, "JFK",            {<AirportCode, "ThreeLetterCode">} >,
  < ArrivalTime, "11:00",             {<TimeFormat, "HH:MM"> } >,
  < Service, "L",                     {<ServiceCode, "OneLetterCode">} >
                                                                 } > } >
```

Figure 3. Representation of Flight Data in MIX

For example, the first flight described in Figure 1 may be represented as a complex semantic object of the ontology concept *OneWayFlight* as depicted in Figure 3. Each flight is identified by its constituting flight segments. In turn, flight segments are identified by flight number, airline, and departure date. Additional properties, such as departure time, or meal services are not required to identify a flight segment and might not be given for all flight segments. The semantic objects on the right side of Figure 3 provide the semantic metadata, or context, needed to interpret the values of the concepts on the left side. A formal and more detailed presentation of the MIX model is given in [3].

In our Java representation concepts are defined by classes, as shown in Figure 4. Each class contains an informal description (in English), and a formal computer interpretable specification of the concept and its semantic relationships with other concepts, i.e., generalization/specialization (is-a), and aggregation (is-part-of). In addition, concept specifications may contain concept-specific functions, e.g., comparison operators, or conversion functions.

For classes used by simple semantic objects, the formal specification consists of the set of concepts inherited from, a slot to keep the actual data value, and a list of semantic objects representing the semantic context of an object of this concept. Since the aspects to be specified in the context are not determined by the concept, the context information may vary between objects of the same concept.

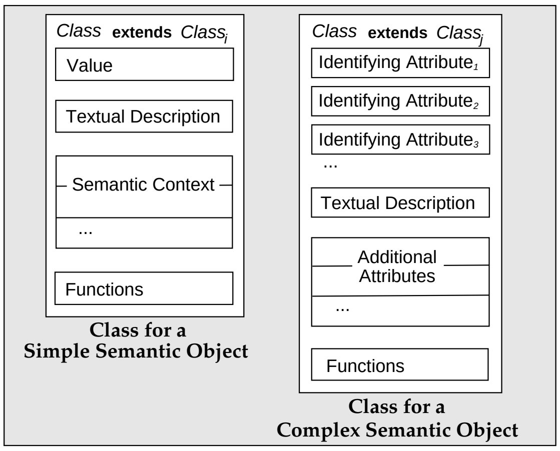


Figure 4. Representing Ontology Concepts in Java

For concept classes used by complex semantic objects, the formal specification consists of the set of concepts inherited from, slots for identifying attributes, and a list of non-identifying attributes. The concepts of these attributes can be understood as being in an is-part-of relationship with the concept of the complex semantic object, i.e., they describe properties or constituents. The non-identifying attributes in a complex semantic object may vary between different objects of the same concept. Figure 5 shows how the semantic object from Figure 3 is represented in Java.

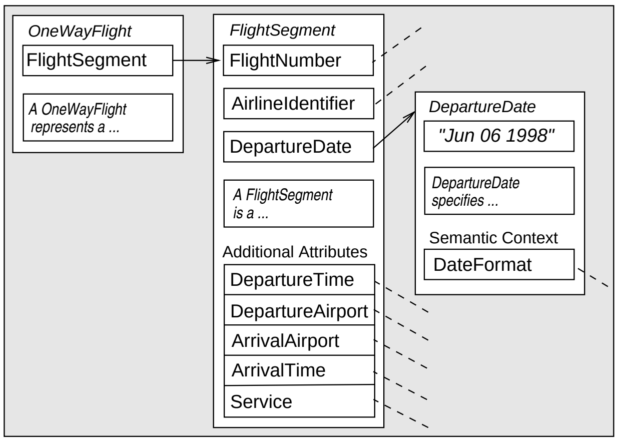


Figure 5. Java Representation of a *OneWayFlight*

Even if this approach does not provide such a mathematically handy representation, and is not as powerful in supporting inference mechanism as languages like KL-ONE, LOOM, KIF, or CLASSIC, it provides a less complex and easier to understand representation of ontologies better suited to our scope of application – providing a common interpretation basis for semantically meaningful data exchange and integration.

In particular, using Java classes to represent ontology concepts has two main advantages. First, by mapping local constructs to the corresponding concepts we specify how local data is mapped to Java objects which can be used by an application program, according to the semantics of the concept associated, without any additional translation, thus, avoiding any impedance mismatch between programming language and ontology specification language.

Second, concepts can be made available as precompiled Java classes via an ontology server. Using Java as the representation language for concepts allows their shipping, as well as that of the corresponding semantic objects, between different platforms without requiring any further transformations.

4. ORGANIZATION OF ONTOLOGIES

When conceptualizing a given subject domain, we can profit from subdividing the modeling task in separate portions. Ontologies in the MIBIA framework support this by being composed from separate modules, or theories, which consist of closely related concepts. An ontology concept is understood here to be an abstraction of a set of real world phenomena which from the perspective of the model are uniform. In this way, ontologies in MIBIA consist of sets of specific theories that provide sets of concepts belonging closely together, such as metric systems, classification schemata, or nomenclatures. In contrast, an ontology always provides a self-contained and consistent conceptualization of a given domain from the application's point of view.

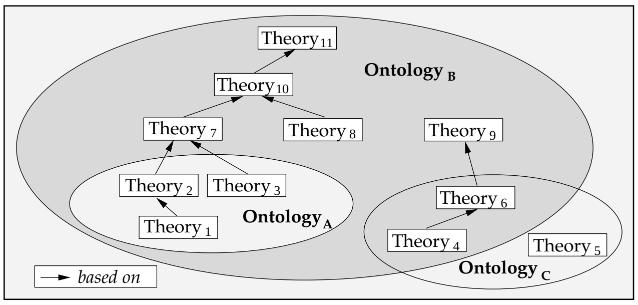


Figure 6. Modularizing Ontologies

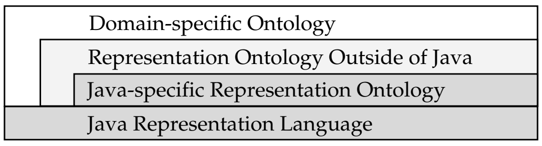
Theories can be built on the basis of concepts from more fundamental theories, as depicted in Figure 6. If a given theory T_{sub} imports a more basic theory T_{super}, then all concepts specified in T_{super} are also available in T_{sub}, and their respective ontologies are in a corresponding relationship. For example, because B includes all theories from A, A and B are in a subset/superset relationship. That means, the subject domain of ontology B includes all aspects formalized in A. In contrast, C and B are not in a subset/superset relationship because they provide only an intersection of common theories. However, this conceptual intersection indicates that the domains modeled by B and C, are semantically related.

In addition to being a more natural and easier way to comprehend the organization of ontologies, the modularization of ontologies and the use of an inclusion mechanism to compose them from different theories provides the basis for an easier extensibility of ontologies. In addition, it supports the efficient exchange and use of ontologies because we can assemble a minimal ontological basis, including only theories necessary for the cooperation task at hand.

In the MIBIA framework, theories are organized as Java packages. Packages support a modular organization of ontologies as packages or sub-packages. Concepts (classes) from more fundamental theories (packages) can be referred to in a theory (package) with a Java import statement. Java packages also provide a hierarchical name space. Thus, naming conflicts between ontologies or theories specified in different packages are impossible.

Representation and Domain-specific Ontologies

According to the semantics of theories or concepts, we may distinguish between representation ontologies and domain-specific ontologies [9]. A representation ontology is domain-independent – that is, it contains only concepts like *numeric value* or *character string*, for the representation of (domain-specific) concepts. An example of a representation ontology in this sense is provided by the Frame ontology developed in the Ontolingua project [5]. In contrast, domain-specific ontologies refer to a concrete subject domain, and provide a consistent conceptualization of this subject area.



Domain-specific Ontology
Representation Ontology Outside of Java
Java-specific Representation Ontology
Java Representation Language

Figure 7. Ontologies Based on Java

The distinction between representation and domain-specific ontologies results in a clean separation of facts from a predefinition of their representation in the ontology. Thus, the specification of a common representation ontology enables the exchange and reuse of concepts from different ontologies developed independently.

The dependencies between representation and domain-specific ontologies are illustrated in Figure 7 based on Java as the underlying representation language. The language constructs in Java, such as *class, extends, static variables* provide the description basis for the specification of concepts in Java. They are extended by a set of more powerful representation concepts in the form of language specific classes as given in the java.lang package, e.g., *Integer, Float*, or *String*. Based on this, language independent representation concepts, such as generic container classes can be specified; these can be used to specify domain-specific concepts like *OneWayFlight* in Figure 3.

Integrating Ontologies from Different Domains
To use data from different domains in combination we have to correlate different ontologies that may overlap, i.e., describe the same real world aspects, but use different concepts to model the same phenomena. This poses the question of how to resolve such differences on the ontological level, and how to integrate ontologies from different domains.

The basic requirement for integrating data from overlapping subject areas is to expose and explicitly describe the semantic relationships between the corresponding ontologies. In the literature several possible approaches for doing this have been suggested. For example, in [21] the following approaches have been discussed.

1. Concepts and their relationships common to the different ontologies are collected and given to a committee to resolve any differences by defining mutually-agreed-upon terms. These are documented and the results distributed to all participants, who are expected to adjust their usage to conform to the common terms [17, 19].
2. All participants initially assume that all concepts are used in a unique way. In the case of obvious discrepancies, the corresponding terms are explicitly separated by using source- or domain-specific prefixes [12]. Over time, the process of exchanging data – encouraged by the availability of the joint ontology – will result in a convergence concerning a common understanding of concepts, although at no time complete coherence can be guaranteed. This approach, requires a relatively stable set of concepts.
3. All participants assume that all concept names refer to different real world aspects until they are explicitly shown to refer to the same real world aspect. These relationships are then explicitly codified as equivalence rules, which form an explicit knowledge base that can be managed by representatives of the different subject areas. This approach, a similar form of which is found in [16], is in the center of discussion in [21].

All of these approaches depend on the existence of a central controlling instance or require participants to adjust to the resulting ontology. This may sacrifice their autonomy, and may need to be repeated for each participant entering the federation. Thus, these approaches seem to be unsuitable for a cooperation of autonomous, perhaps previously unknown participants in a worldwide and loosely coupled environment like the Internet.

In MIBIA the following approach for integrating conceptually overlapping ontologies is used. It is similar to the one followed by the Ontolingua project [5] concerned in developing a suitable infrastructure for the common use of knowledge bases on independent systems.

As a basis for the formalization of each subject area, a common pool of ontologies, organized as a graph of theories, is available to all participants. Organizing theories and ontologies as Java packages, such a common concept pool may be organized as a globally accessible directory service. The underlying conceptualization of a data source or application can be mapped independently of others to the corresponding concepts.

If the ontology pool does not contain a needed concept, it has to be enlarged. That is, a new concept has to be introduced by supplementing or specializing an existing theory.

Accordingly, an extension of the existing ontology pool by ontologies from new domains can be performed by successively adding the corresponding concepts. Thereby, the integration of new ontologies into the existing ontology graph is guaranteed through the reuse of concepts and theories in existing ontologies as much as possible. Multiple ontologies may share the same theories without affecting each other. Because of such sharing, there is little risk of introducing multiple concepts for the same real-world phenomenon. The semantic relationships between affected ontologies are made explicit as follows: Common or closely-related theories in two ontologies indicate semantic similarities in both conceptualizations. For example, synonymous concepts can be resolved to a single concept, which is then used by all ontologies. Homonymous concepts can be differentiated through different names, or by using ontology-specific prefixes. Hyponyms and hypernyms are made explicit through the specification of corresponding generalization and specialization relationships between concepts in the pool.

Any differences in the view or intended use of a real world phenomenon in different ontologies are reflected in the existence of multiple concepts for that phenomenon, whereat it is assumed that each concept has a different meaning. However, it is possible to compare concepts and to discover semantic relationships between ontologies because all ontologies are represented on the basis of a common representation ontology.

The main difference of our approach to those discussed above is that semantic relationships between different ontologies, such as synonym-, hyponym-, or hypernym-relationships between their concepts, are explicitly described in the integrated ontology pool. Here the modularization of ontologies is important, as it supports the reuse of concepts, makes it easier to create new ontologies, and reduces the risk of overlap. In contrast, the approach described in (3) specifies these dependencies through the introduction of an additional description level.

The extension of the common concept pool by independent participants makes it possible to independently introduce new concepts and makes the existence of a central controlling instance superfluous for many application domains. The MIBIA prototype shows that it provides a suitable solution for domains where common, comparatively stable conceptualizations can be reached, such as travel services, an application domain which motivated the current research and on which the prototype has been tested.

5. RELATED RESEARCH

Due to space limitations we discuss here only three approaches closely related to MIBIA.

Carnot [4] supports the development of applications operating on heterogeneous data sources in an enterprise[1]. It uses the Cyc ontology [14], which provides knowledge from different subject areas, to make explicit the semantics of the available data. Logical integration of a data source can be performed independently from others by specifying a bidirectional translation between local structures and concepts from Cyc. Carnot allows uniform access to data through the global view given by the source-independent ontology of Cyc.

MIBIA does not provide a homogeneous view of the data right away, but provides a framework for correctly interpreting data. Structural/schematic heterogeneities and differences in terminology are resolved by appropriate wrappers. However, differences in describing same real world aspects by using different concepts, e.g., different units of measure, scaling factors, or name schemas, are resolved at query processing time. Thus, MIBIA allows to resolve semantic heterogeneities dependent on the respective receiver of the data. Carnot uses equivalence equations to specify the relationships between local structures and Cyc concepts that leave open what physical representation type the resulting data items correspond to. This makes their uniform processing more difficult. In contrast, concepts in MIBIA are represented as Java classes and data objects are given as class instances. Finally, Cyc provides a monolithic model for data in an enterprise. The situation of a much larger set of autonomous sources from varying domains as they are typical for the Internet requires extensibility of the common vocabulary. This is supported by a modular approach as described in Section 4. Thus, in the MIBIA framework we begin with a set of interrelated ontologies from different domains.

OBSERVER [16] provides a framework for query processing in distributed, heterogeneous systems. The content of each source is described on an intensional level using existing ontologies. These descriptions function as an abstraction from the specific representation of a source to the source-independent view of the ontology. Data objects are seen as instances of entity types represented in CLASSIC. OBSERVER allows the use of different ontologies for the same domain to support the reuse of existing ontologies and the scaling of the system. Since ontologies are not integrated, semantic relationships, e.g., synonyms, hypernyms, and hyponyms, have to be specified and managed by a separate component, and additional translation steps between different ontologies are needed during query processing.

In contrast, MIBIA uses a unified set of ontologies from different domains. Synonyms are avoided by modularizing ontologies and reusing theories in different ontologies. Hypernyms and hyponyms are made explicit by specializing or generalizing concepts from other ontologies. Thus, the semantic relationships between different ontologies are specified in the integrated ontology model, which makes their management easier.

The **COIN** project [2] deals with the development of technologies to support uniform access to information from heterogeneous sources. The resolution of semantic heterogeneities is a major concern. Integration follows the *context interchange approach* [7], which does not require static resolution of semantic heterogeneities, but resolves them during query processing by comparing their contexts made explicit as metadata. To allow a meaningful comparison of this metadata it has to be described using a common vocabulary. Thus, COIN is based on the use of ontologies from different domains which have to be integrated.

The main differences between MIBIA and COIN are the ways in which ontologies and metadata are represented. In COIN, a logic-based representation in the form of horn clauses is used for ontology concepts and their relationships, and the semantic context of a source. Context information is given on the intensional level. In contrast, concepts in MIBIA are represented as Java classes, and the representation of semantic metadata is on the extensional level, i.e., as part of the data objects. Thus, MIBIA supports the integration of semistructured sources that provide no explicitly specified schema to which metadata could refer. In addition, metadata can be used directly, by the application, without any further processing since it is accessible as additional attributes of the data.

6. CONCLUSION

To allow a semantically meaningful data exchange between independent participants we need commonly agreed upon vocabularies to describe data and metadata. In this paper we have shown how ontologies are used in the MIBIA project to provide such a common interpretation basis.

In contrast to most approaches discussed in the literature that represent context information on an intensional level, we provide a more flexible approach to represent additional information on the extensional level which is well suited for the description of semistructured data. In addition, most

[1]The InfoSleuth project [1, 6] investigates the use of Carnot technologies in a dynamically changing environment, such as the Internet.

prototypes for the integration of data from autonomous sources use logic-based specification languages for the description of ontologies. Instead, we use Java to specify ontology concepts and their relationships, thus avoiding any impedance mismatch between programming language and ontology specification language, and allowing the shipping of ontology concepts between different platforms without requiring any further transformations.

Our approach is not claimed to be applicable for every subject area. However, it provides a fairly simple and pragmatic solution for many application domains, like travel information, or brokering of products and services. The present research was motivated by a request from a major travel agency to help them extract data from the web and prepare it for further automatic processing. Our prototype has been proven feasible in this context.

Current work concerns the development of tools to support the creation of wrappers for the mapping of different source types to the common representation model, and for the management of ontologies in MIBIA. We evaluate the use of MIX and MIBIA technologies in the WE-trade project, an EU sponsored international trade center to promote East-West business-to-business e-commerce in which *multilingual* product and service offers are brokered.

Acknowledgments: We gratefully acknowledge the contributions of Ronald Bourret for his suggestions and helpful comments regarding an earlier version of this paper, and Michael Watzke for his help in implementing the prototype.

REFERENCES

[1] Bayardo, R.J.; Bohrer, W.; Brice, R.; et al.: *The InfoSleuth Project*, SIGMOD Int'l Conf. on Management of Data, Tucson, Arizona, USA, 1997

[2] Bressan, S.; Goh, C.H.; Fynn, K; et al.: *The Context Interchange Mediator Prototype*, SIGMOD Int'l Conf. on Management of Data, Tucson, Arizona, USA, 1997

[3] Bornhövd, C.: *Semantic Metadata for the Integration of Web-based Data for Electronic Commerce*, Int'l Workshop on E-Commerce and Web-based Information Systems, Santa Clara, CA, 1999

[4] Collet, C.; Huhns, M.; Shen, W.: *Resource Integration Using a Large Knowledge Base in Carnot*, IEEE Computer, Vol. 24, No.12, Dec. 1991

[5] Farquhar, A.; Fikes, R.; Rice, J.: *The Ontolingua Server: A Tool for Collaborative Ontology Construction*, 10th Knowledge Acquisition for Knowledge-Based Systems Workshop, Alberta, Canada, 1996

[6] Fowler, J.; Nodine, M.; Perry, B.; Bargmeyer, B.: *Agent-Based Semantic Interoperability in InfoSleuth*, SIGMOD Record, Vol. 28, No.1, March 1999

[7] Goh, C.; Madnick, S.; Siegel, M.: *Context Interchange: Overcoming the Challenges of Large-Scale Interoperable Database Systems in a Dynamic Environment*, In: Proceedings of the 3rd International Conference on Information and Knowledge Management, Gaithersburg, Maryland, 1994

[8] Gruber, T.; Olsen, G.: *An Ontology for Engineering Mathematics*, 4th Int'l Conf. on Principles of Knowledge Representation and Reasoning, Bonn, Germany, 1994

[9] Gruber, T.: *A Translation Approach to Portable Ontology Specifications*, Technical Report KSL 90-53, Knowledge Systems Laboratory, Department of Computer Science, Stanford University, California, 1993

[10] Gruber, T.: *Toward Principles for the Design of Ontologies Used for Knowledge Sharing*, In: Int'l Journal of Human-Computer Studies, Vol. 43, No.5/6, 1995

[11] Guarino, N.: *Formal Ontology and Information Systems*, Int'l Conf. on Formal Ontology in Information Systems, Trento, Italy, 1998

[12] Humphreys, B.L.; Lindberg, D.A.B.: *The Unified Medical Language Project: A Distributed Experiment in Improving Access to Biomedical Information*, MEDINFO'92, North-Holland, 1992

[13] International Organization for Standardization / International Electrotechnical Commission: *Electronic Data Interchange for Administration, Commerce and Transport (EDIFACT)*, International Organization for Standardization / International Electrotechnical Commission, 1988

[14] Lenat, D.; Guha, R.: *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*, Addison-Wesley, Reading, Mass., 1990

[15] Madnick, S.E.: *From VLDB to VMLDB (Very MANY Large Data Bases): Dealing with Large-Scale Semantic Heterogeneity*, 21st Conf. on Very Large Data Bases, Zurich, Swizerland, 1995

[16] Mena, E.; Illarramendi, A.; Kashyap, V.; Sheth, A.: *OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies*, Distributed and Parallel Databases, Vol. 8, No.2, April 2000

[17] Morris, K.C.; Mitchell, M.; Dabrowski, C.; Fong, E.: *Database Management Systems in Engineering*, Encyclopedia of Software Engineering, John Wiley and Sons, 1994, S.282-308

[18] Moulton, A.; Madnick, S.E.; Siegel, M.D.: *Context Mediation on Wall Street*, In: Proceedings of the 3rd IFCIS International Conference on Cooperative Information Systems, New York, 1998

[19] Porter, J.H.; Henshaw, D.L.; Stafford S.G.: *Research Metadata in Long-Term Ecological Research (LTER)*, 2nd IEEE Metadata Conf., Silver Springs, Maryland, 1997

[20] van der Vet, P.E.; Mars, N.J.I.: *Bottom-Up Construction of Ontologies*, IEEE Transactions on Knowledge and Data Engineering, Vol. 10, No.4, 1998

[21] Wiederhold, G.: *Interoperation, Mediation, and Ontologies*, Int'l Symposium on Fifth Generation Computer Systems, Workshop on Heterogeneous Cooperative Knowledge-Bases, Tokyo, Japan, 1994

[22] Wiederhold, G.: *Mediation in Information Systems*, ACM Computing Surveys, Vol. 27, No.2, 1995