Ontologies and Semantics for Seamless Connectivity[‡]

Michael Uschold
Boeing, Phantom Works
P.O. Box 3707, m/s 7L-40
Seattle, WA 98124-2207, USA
michael.f.uschold@boeing.com

Michael Gruninger
Institute for Systems Research
University of Maryland
College Park, MD 20742, USA
gruning@nist.gov

Abstract

The goal of having networks of seamlessly connected people, software agents and IT systems remains elusive. Early integration efforts focused on connectivity at the physical and syntactic layers. Great strides were made; there are many commercial tools available, for example to assist with enterprise application integration. It is now recognized that physical and syntactic connectivity is not adequate. A variety of research systems have been developed addressing some of the semantic issues. In this paper, we argue that ontologies in particular and semantics-based technologies in general will play a key role in achieving seamless connectivity. We give a detailed introduction to ontologies, summarize the current state of the art for applying ontologies to achieve semantic connectivity and highlight some key challenges.

1 Introduction

One of today's most pervasive and bedeviling IT challenges is to get "*the right information to the right people at the right time*". Achieving this 'holy grail' requires seamless connections among people, software agents and various kinds of IT systems. Such connections are required to support the emergence of vibrant communities that can exchange and effectively use of the full range of data, information, knowledge and wisdom.

Early connectivity efforts have focused on the physical and syntactic layers. For example, Enterprise Application Integration approaches have used such technologies as: ODBC data gateways, message-oriented middleware, composite integration technology, software adapters, message transport and routing services [Pollock, 2002]. While this has been a major achievement, the fact that connectivity was *only* achieved at the physical level is problematic. Streams of data were successfully transmitted between systems, however there was no *meaning* associated with the data. This is analogous to successful delivery of an encrypted message, appearing to the recipient as mere scratchings on the page. With no meaning coming across, it is necessary to hardcode what to do with each data item from each application. Such hard-coding is usually done on a point to point basis. All this results in brittle systems that are limited in flexibility and expensive to maintain. Most commercial systems remain limited in this way.

Web services are a new way to package up IT functionality, and are hailed by many as the 'next big thing' in computing. The major breakthrough is to loosen up the tight physical coupling between systems. This allows for fewer stovepipe solutions and consequent increase in flexibility and reduced maintenance costs. However, Web services still are focused on physical connectivity, and do not address the semantics of the exchanged data.

Today's Web services technology assumes that all systems are semantically homogenous – i.e. that they will all use the same vocabulary. There is a long history of failed projects which attempt to integrate many systems into a single logical unit using the same vocabulary. There will always be sufficiently large groups for which global agreements are infeasible. It has been recognized that without addressing the reality of semantic heterogeneity, full seamless connectivity between systems will not be achieved. "It's the semantics, not the plumbing", says Michael Brodie[1]. Jeffrey Pollock says: "Semantics-based technologies will be an essential part of all interoperability solutions in the very near future" [Pollock 02].

XML documents were originally hailed as being much clearer, semantically, than alternatives. However the tags are only understandable by humans [Cover 1998]. There has been a recent surge of activity related to the Semantic Web[2] [Berners-Lee *et al.* 2001]. The idea is to make Web content more accessible to machines by using semantic markup. However, the problems of semantic integration remain the same: people use terms differently and mapping and translation must take place across

[‡] Certain software tools are identified in this paper in order to explain our research. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the software tools identified are necessarily the best available for the purpose.

[1] Invited Talk, ISWC, October 2003, Sannibel, Florida.
[2] See: http://www.w3.org/2001/sw/

different communities. Semantic integration has attracted a lot of attention in the research community in recent years, but there has been relatively little commercial impact to date.

It is becoming increasingly clear that the emerging technology concerned with the development and application of *ontologies* will play a central role in overcoming the problems of semantic heterogeneity [Bradshaw *et al* 2004; Wache *et al.* 2001]. An ontology is used by an agent, application, or other information resource to declare what terms the agent uses, and what the terms mean. By making this information publicly available, it becomes possible for high fidelity semantic communication to take place -- agents can communicate and share meaning with other agents, and agents can understand the meaning of applications, databases and other information resources on the Web. In this paper, we will consider the broad area of semantics-based technologies, and focus on ontologies in particular.

2 Overview of Ontologies

To meet a variety of needs in information modeling, software development and integration as well as knowledge management and reuse, various groups within industry, academia, and government have been developing and deploying sharable and reusable models known as ontologies[3].

2.1 What is an ontology?

The most commonly quoted definition of an ontology is *"a formal, explicit specification of a shared conceptualization"* [Gruber 1993]. A *conceptualization*, in this context, refers to an abstract model of how people think about things in the world, usually restricted to a particular subject area. An *explicit specification* means that the concepts and relationships in the abstract model are given explicit names and definitions. The name is a term, and the definition is a specification of the meaning of the concept or relation. A definition says how a term necessarily relates to other terms. *Formal* means that the meaning specification is encoded in a language whose formal properties are well understood—in practice, this usually means logic-based languages that have emerged from the knowledge representation community within the field of Artificial Intelligence. Formality is an important way to remove ambiguity that is prevalent in natural language and other informal notations; it also opens the door for automated inference to derive new information from the meaning specifications. *Shared* means that the main purpose of an ontology is generally to be used and reused across different applications and communities.

[3] Much of the material from this section is drawn from [Bradshaw *et al* 04]; additional material on the emerging field of ontologies may be found in [Staab & Studer, 04].

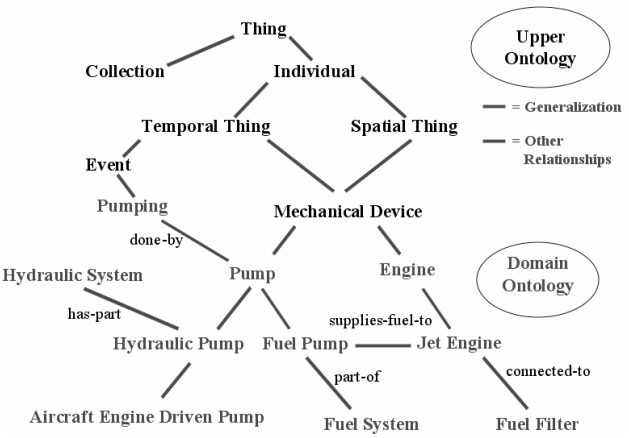

Figure 1: An Example Ontology

There has been much discussion on what exactly counts as an 'ontology'; however there is a common core that runs through virtually all approaches:

- a vocabulary of terms that refer to the things of interest in a given domain;
- some specification of meaning for the terms, [ideally] grounded in some form of logic.

Ontologies represent many different kinds of things in a given subject area (e.g. wing, physical object, wire). These things are represented in the ontology as *classes* (sometimes called *concepts*) and are typically arranged in a lattice or taxonomy of classes and subclasses. Each class is typically associated with various *properties* (also called *slots* or *roles*) describing its features and attributes as well as various restrictions on them (sometimes called *facets* or *role restrictions*). An ontology together with a set of concrete *instances* (also called *individuals*) of the class constitutes a *knowledge base*. The lattice or taxonomy of classes is a primary the focus of most ontologies (Figure 1).

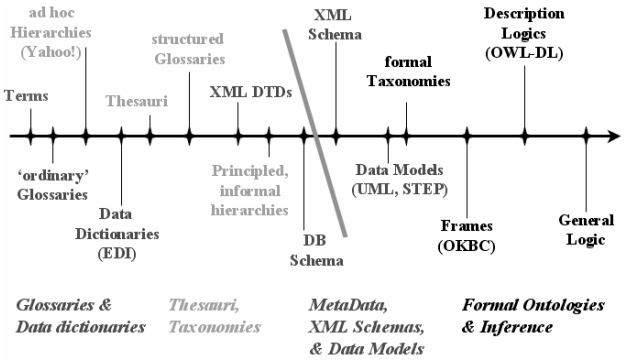

Figure 2: Kinds of Ontologies

What distinguishes different approaches to ontologies is the manner of specifying the meaning of terms. This gives rise to a kind of continuum of kinds of ontologies. At one extreme, we have very lightweight ones that may consist of terms only, with little or no specification of the

meaning of the term. At the other end of the spectrum, we have rigorously formalized logical theories, which comprise the ontologies (Figure 2). As we move along the continuum, the amount of meaning specified and the degree of formality increases (thus reducing ambiguity); there is also increasing support for automated reasoning.[4]

2.2 Ontologies vs. Knowledge Representation and Inference

Ontologies are a key part of a broader range of semantics-based technologies which include the areas of knowledge representation (KR) and automated inference that arose within the Artificial Intelligence community. Many different representation formalisms have been explored, and reasoning engines developed. A key result is the proven existence of a tradeoff between representational power of a language (i.e. the ability to represent/express many different kinds of knowledge) and the efficiency of the reasoning engines.

In one sense, ontologies are a sub-area within knowledge representation. Traditional KR languages are used for representing formal ontologies and standard inference engines are used for reasoning over ontologies. Also, a good knowledge base will frequently have an ontology as its backbone. A major difference is that, unlike for KR in general, a key focus of ontologies is on knowledge *sharing*.

Interpreted more broadly, 'ontologies' go outside the bounds of KR. Many so-called 'lightweight-ontologies' have little relevance to KR. For example, various thesauri and the Yahoo! taxonomy are often called ontologies; they are used in similar ways as some formal ontologies are.

The ontologies community may also be viewed as a major customer for KR – to the extent that this is true, the use of ontologies is really applied KR.

2.3 Ontologies vs. Database Schema

There are many interesting relationships between database schema and formal ontologies. We will consider the following issues: language expressivity, systems that implement the languages and usage scenarios.

There is much overlap in expressivity, including: objects, properties, aggregation, generalization, set-valued properties, and constraints. For example, entities in an ER model correspond to concepts or classes in ontologies, and attributes and relations in an ER model correspond to relations or properties in most ontology languages. For both, there is a vocabulary of terms with natural language definitions. Such definitions are in separate data dictionaries for DB schema, and are inline comments in ontologies. Arguably, there is little or no obvious *essential* difference between a language used for building DB schema and one for building ontologies. They are similar beasts. There are many specific differences in expressivity, which vary in importance. Many of the differences are attributable to the historically different ways that DB schema and ontologies have been used.

Ontologies have a range of purposes including interoperability, search, and software specification (see section 3). One or more parties commit to using the terms from the ontology with their declared meaning. The primary use of most DB schema is to structure a set of instances for querying a single database. This difference impacts heavily on the role of constraints.

For ontologies, constraints are called axioms. Their main purpose is to express machine-readable *meaning* to support accurate automated reasoning. This reasoning can also be used to ensure integrity of instances in a knowledge base. For databases, the primary purpose of constraints is to ensure the *integrity* of the data (*i.e.* instances). These 'integrity constraints' can also be used to optimize queries and help humans infer the meaning of the terms. Cardinality and delete constraints are important kinds of integrity constraints which have highly DB-specific uses that are outside the scope of most or all ontology systems. For example, cardinality constraints are used for getting the foreign key in the right direction and to ensure that extra tables are built for many to many relationships. Such constraints do express meaning, but this may be of secondary importance. The main roles for cardinality constraints in ontologies is to express meaning, and ensure consistency (either of the ontology, or of instances).

There are some key similarities and differences in systems that implement DB schema languages vs. ontology languages. For both, there are processing engines that can be used to perform reasoning. SQL engines are highly specialized and tuned for answering queries, reasoning with views and ensuring data integrity. They can handle rich and expressive logical expressions, as can ontology engines. An important difference is that reasoning over ontologies normally is done by general logic-based theorem provers, specific to the language. Although ontology inference may be used for queries and to ensure integrity of instances, these are optional. The fundamental role of a reasoning engine is to derive new information via automated inference. Inference can also be used to ensure the logical consistency of the ontology itself. Note that deriving new information and checking consistency can take place with or without instances. Classically, such mixing of types with instances does not take place with DB schema and data. This is mainly due to much greater scale and performance requirements for database systems. Another key difference is support for taxonomic reasoning: it is fundamental for nearly all ontology applications, but it is not supported by most DBMS.

[4] We will generally use the term 'ontology' in the narrow sense, restricted to formal logic-based models.

The different roles of DB schema and ontologies also affect design and other pragmatic issues. For example, there is much effort on normalization for DB schema, with no similarly pervasive analogous step for ontologies.

Enforcement of DB integrity constraints is expensive; therefore many constraints identified during modeling are left unstated, resulting in loss of captured meaning. This makes semantic integration difficult—if you don't know what things mean, how can you relate them to anything else? Capturing as much data meaning as possible in machine readable format is important for interoperability (e.g., for schema/ontology matching). Hence, a good step toward designing database schemas for interoperability would be more widespread decoupling of constraint definition from constraint enforcement (as is done in the ontology community).

3 How do ontologies help?

The promise of ontologies is "*a shared and common understanding of a domain that can be communicated between people and application systems*" [Fensel 2001]. We identify four main categories of ontology application scenarios. These are illustrated in figure 3 and discussed below [Jasper and Uschold 1999].

Neutral Authoring: Given the plethora of non-interoperable tools and formats, a given company or organization can benefit greatly by developing their own neutral ontology for authoring, and then developing translators from this ontology to the terminology required by the various target systems. Although this entails a very time-consuming effort in developing the neutral ontology such that it has adequate coverage in the subject matter of interest, the benefits of this approach include knowledge reuse, improved maintainability, and long-term knowledge. To ensure no loss in translation, the neutral ontology must include only those features that are supported in *all* of the target systems. The tradeoff here is loss of functionality of some of the tools; certain special features may not be usable.

Enterprise modeling is a prime example of the neutral authoring application of ontologies, particularly within enterprises that must integrate multiple software applications using a semantically uniform core.

Common Access to Information: In any given area where legacy software systems are required to interoperate, it will always be necessary to translate between various different formats and representations that evolved independently. While it is safe to assume there will not be global ontologies and formats agreed by one and all, it is nevertheless possible to create an ontology to be used as a neutral interchange format for translating among various formats. This avoids the need to create and maintain $O(N^2)$ translators and it makes it easier for new systems and formats to be introduced into an existing environment. In practical terms, this can result in dramatic savings in maintenance costs -- it has been estimated that 95% of the costs of enterprise integration projects [that become operational] is maintenance [Pollock 02].

This scenario is similar to neutral authoring, in that a neutral ontology is created in the domain of the systems that must interoperate. The key difference is that the translation is bidirectional—interoperability between any two systems happens by first translating from the source system format into the neutral format, and from there into the target system format. While it is inevitable for some loss of translation to occur between systems with differing functionality, it is desirable to limit this by ensuring that there is no loss of meaning in the translation from the source format to the neutral ontology. This impacts on how the neutral ontology is designed, since the neutral ontology must cover all of the concepts that occur in *any* of the target systems.

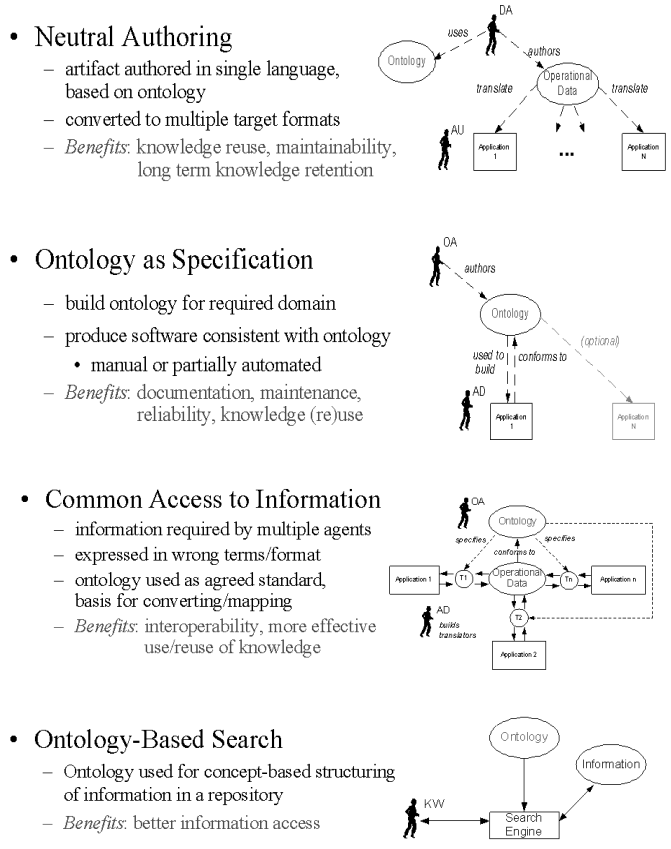

Figure 3: Ontology Application Scenarios
DA: data author, OA: ontology author, AD: application developer; AU: application user; KW: knowledge worker

The application of ontologies for common access to information is most widely seen in the development of standards. The Standard for the Exchange of Product Data (STEP / ISO 10303) and the Process Specification Language (PSL / ISO 18629) are excellent examples.

STEP supports interoperability among manufacturing product software (such as CAD systems and process planning software) throughout the entire product lifecycle. STEP provides standard data definitions for geometry (wire frame, surfaces and solid models), product identification, product structure, configuration and change management, materials, finite element analysis data, drafting, visual presentation, tolerances, kinematics, electrical properties and process plans. STEP is currently being implemented in the aerospace, automotive, shipbuilding, building design and electronics industries.

The Process Specification Language [Gruninger & Menzel 2003] has been designed to facilitate correct and complete exchange of process information among manufacturing and business software systems. Included in these applications are scheduling, process modeling, process planning, production planning, simulation, project management, workflow, and business process reengineering. Written in the Knowledge Interchange Format, the PSL Ontology is modularly organized into PSL-Core and a partially ordered set of extensions to this core theory.

Note that sanctioning by official standards bodies is not necessary. Any exchange community can adopt a neutral ontology as a *de facto* standard within the community and enjoy the same benefits.

Ontology-based Specification: There is a growing interest in the idea of "Ontology-Driven Software Engineering"[5] in which an ontology of a given domain is created and used as a basis for specification and development of some software. The idea is to create an ontology that characterizes and specifies the things that the software system must address, and then use this ontology as a (partial) set of requirements for building the software. For example, a database schema defines the vocabulary and specifies various integrity constraints that serve a set of requirements for the DB administrator, who must create an implemented DB faithful to the schema.

As in the neutral authoring scenario, the same ontology can be used as the basis for multiple systems. The difference between the two is the relationship between the ontology and the target system. For neutral authoring, there is a translator built to convert specifications in the neutral format into the required target system formats, and beyond this, there need not be any special relationship between the target software systems and the neutral format. In the case of ontology-based specification, the ontology is used as the *basis* for software development. For example, the development of an entire suite of product design software (including viewing and presentation tools, data bases, and even marketing and accounting tools that are used to track product sales) could be driven from the same ontology. This would ensure easier interoperability among software systems whose relationships are typically only implicit.

The benefits of ontology-based specification are best seen when there is a formal link between the ontology and the software. This is the approach of the so called "Model-Driven Architecture" created and promoted by the OMG[6], as well as ontology software which automatically creates Java classes and Java Documents from an ontology[7]. When the ontology changes, the code is automatically updated. A large variety of applications may use the accessor functions from the ontology. Not only does this ensure greater interoperation, but it also offers significant cost reduction for software evolution and maintenance. A suite of software tools all based on a single core ontology are semantically integrated for free, eliminating the need to develop translators.

Ontology-based Search: To facilitate search, an ontology is used as a structuring device for an information repository (e.g., documents, web pages, names of experts); this supports the organization and classification of repositories of information at a higher level of abstraction than is commonly used today. They can be used as a sophisticated indexing mechanism into such repositories. A very familiar example of this use of ontologies is the Yahoo! subject taxonomy, which can be thought of as a lightweight ontology that is well suited to its use for browsing the Web. However, close examination of this taxonomy reveals that it is somewhat 'semantically challenged' – e.g. it is not a strict taxonomy; the links have many different meanings. Although this is adequate for human users, it makes it impossible for a machine to do automated reasoning with predictable results.

Using ontologies to structure information repositories also entails the use of semantic indexing techniques, or adding semantic annotations to the documents themselves. If different repositories are indexed to different ontologies, then a semantically integrated information access system could deploy mappings between different ontologies and retrieve answers from multiple repositories. See [Barrett et al 2002] for an example of this technique applied to databases.

More sophisticated applications of ontology-based search require more formal ontologies. OWL-S is an OWL[8] Ontology for describing web services, created by a

[5] E.g., see: www.bpiresearch.com/WP_BPMOntology.pdf

[6] www.omg.org/mda/

[7] E.g., see: www.ontologyworks.com/PR-patent.htm

[8] See: www.w3.org/2004/01/sws-pressrelease.

coalition of researchers through the support of the DARPA Agent Markup Language program. OWL-S supplies Web service providers with a core set of markup language constructs for describing the properties and capabilities of their web services. Such service descriptions facilitate automated advertising and discovery of Web services by intelligent agents [Sycara *et al*, 04].

4 Issues, Challenges, Progress

The holy grail is the achievement of fully automatic semantic interoperability among independently developed and heterogeneous agents. Unfortunately, in its full generality, the problem amounts to cryptography, and is intractable. To make progress, individual researchers and practitioners will have to initially make many assumptions, and then relax them one by one as technology progresses. Examples of important assumptions that can be made are:

1. that all parties use a single language for representing ontologies;
2. that all members in a given exchange community use:
    a. a single shared ontology, *or*
    b. a single shared upper ontology, with distinct domain ontologies, *or*
    c. a shared interlingua ontology to map individual ontologies to and from;
3. that semantic mapping among ontologies will be human-assisted, rather than fully automated;
4. that mapping will be done between lightweight ontologies, with a limited role for automated reasoning;
5. that there is adequate infrastructure support for community repositories of both ontologies and inter-ontology mappings.

Analyzing these various assumptions sets out a research agenda. Recent efforts of the Semantic Web community have focused on making the first assumption viable on a large scale; for example, some ontology tool vendors are already supporting OWL, sanctioned by the W3C as the standard ontology language for the Web.

The second set of assumptions represents points along two theoretical extremes for semantic integration. In the first case there is *complete global agreement* on terms and their meaning; issues of semantic integration do not arise because there is a single shared ontology, which need not even be explicit. At the other extreme, there is *complete semantic anarchy* with no agreements. There are many good reasons why we should never expect to achieve the former, since software vendors will retain their proprietary formats; on the other hand, the costs of semantic heterogeneity to companies and society are prohibitive. The pragmatic approach will be to reach agreements and use common standards or a shared upper level ontology whenever possible. It will also be necessary to create mappings between multiple, and possibly conflicting, ontologies; interoperability can be achieved by executing these mappings to translate data and messages between formats based on different ontologies.

Creating mappings is a labor-intensive and error-prone activity, even for humans. Many current mapping tools are semi-automated, helping humans in an interactive manner. For applications where accuracy is of the utmost importance, this approach is likely to continue for some time. Some mapping is done in a fully automated manner, using some combination of linguistic and learning techniques [Doan 2002]. However, the results are much more error prone, and the ontologies that these techniques are used on tend to be less formal. Users need to choose a mapping tool that gives the most labor-saving automation, provided that it gives an acceptable level of accuracy. Researchers will strive to improve accuracy of fully automated techniques, augmenting them with interactive approaches which involve humans.

Another key element of a research strategy concerns the balance between using semantically lightweight representations versus semantically rich representations with formal axiomatizations. The tradeoff here is between computational cost and flexibility and powerful reasoning capabilities. We believe that there will always be a wide range of solutions which use approaches ranging from very lightweight to richer formal representations.

Note that there is an important relationship between the amount of agreement there is within an exchange community and the role of rich semantics and automated reasoning. Many current applications have no explicit ontologies; rather everyone assumes that they know what is meant by certain terms from context. For example, Web-based applications such as travel and bookseller agents automatically access [non-semantic] Web pages looking for good deals. However, this will only work to a point. We claim that:

The **less** the following things are true:

- *there is widespread agreement about the meaning of a term, and the syntax for expressing it, and*
- *all the software is built by humans who correctly embed the agreed meaning of the term, and*
- *all the databases and Web pages and applications use the term in the agreed way,*

then the **more** necessary it is to:

- *have an explicit formal declarative semantics of the term that the machine can process to interpret the meaning of that term.*

To a large extent it will depend on the type of the problem. Problems requiring flexibility and powerful reasoning capabilities will be driving the advance of the semantic integration technology development.

Many semantic mapping, integration and/or interoperability projects take place more or less in a vacuum. In order for vibrant communities of semantically connected of agents, applications etc to emerge on a large scale, there needs to be some general infrastructure in place where one can easily register, access and use various things such as: ontologies, mappings between ontologies, mapping languages, and translation engines. At a minimum, it should be possible to issue requests such as: "Given this message, encoded using ontology A, please return a translated message encoded using ontology B. And please use this particular mapping and that particular translation engine." Such infrastructure is currently lacking, although there is some good early work in this area [Melnik *et al* 2003] .

Finally, we suggest that there be challenge problems with associated metrics so that ontology mapping can be evaluated in a meaningful way. Testbed environments must be created to compare different approaches. The first Information Interpretation and Integration Conference in August 2004[9] is a step in this direction.

Acknowledgements

Natasha Noy gave feedback on an earlier draft. Phil Bernstein, AnHai Doan, and several others helped clarify the differences between DB schema and ontologies.

References

[Barrett, et al, 2002] T. Barrett, D. Jones, J. Yuan, J. Sawaya, M. Uschold, T. Adams & D. Folger; RDF Representation of Metadata for Semantic Integration of Corporate Information Resource. In Proceedings of Real World RDF and Semantic Web Applications Workshop held in conjunction with WWW-2002.

[Berners-Lee et al. 2001] Berners-Lee, T., Hendler, J. and Lassila, O. (2001) "The Semantic Web," Scientific American, May 2001.

[Bradshaw et al 2004] Semantic Integration. In Bradshaw, J. M., Boy, G., Durfee, E., Gruninger, M., Hexmoor, H., Suri, N., Tambe, M., Uschold, M., & Vitek, J. (Ed.). (2004). Software Agents for the Warfighter. ITAC Consortium Report. Cambridge, MA: AAAI Press/The MIT Press (to appear)

[Cover 1998] Cover, R. "XML and Semantic Transparency," The XML Cover Pages

[Doan 2002] AnHai Doan, Jayant Madhavan, Pedro Domingos, and Alon Halvey. Learning to map between Ontologies on the Semantic Web, International World Wide Web Conference (WWW) 2002, Hawaii, USA

[Fensel 2000] Fensel, D. Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce. Springer-Verlag.

[Gruber 1993] Gruber, T. . *A translation approach to portable ontology specifications*. Knowledge Acquisition 5:199-220; 1993

[Gruninger & Menzel 2003] Gruninger, M. and Menzel, C. *Process Specification Language: Principles and Applications*, AI Magazine, 24:63-74.

[Jasper & Uschold 1999] Jasper, R. and Uschold, M. 1999. *A Framework for Understanding and Classifying Ontology Applications*. In Twelfth Workshop on Knowledge Acquisition Modeling and Management KAW'99

[Melnik et al 2003] Melnik, S., Rahm, E., Bernstein, P. *Developing Metadata-Intensive Applications with Rondo*, Jnl of Web Semantics, Volume 1, Issue 1,

[Pollock, 2002] Pollock, J. *Integration's Dirty Little Secret: It's a Matter of Semantics"* Whitepaper; Modulant, The Interoperability Company; February 2002

[Pollock, 2004] Pollock, J. Adaptive Information: Improving Business through Semantic Interoperability, Grid Computing, and Enterprise Integration. John Wiley & Sons, 2004; (to appear)

[Staab & Studer 2004] S. Staab, R. Studer (eds). *Handbook on Ontologies*; Springer Series on Handbooks in Information Systems

[Sycara et al, 04] K. Sycara, M. Paolucci, Ankolekar & N. Srinivasan; *Automated Discovery, interaction and composition of Semantic Web services*; Journal of Web Semantics 1:1

[Wache *et al.* 2001] Wache, H., Voegele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H. and Huebner, S. 2001. Ontology-Based Integration of Information - A Survey of Existing Approaches. In *Proceedings of the IJCAI-01 Workshop on Ontologies and Information Sharing,* Seattle, WA, August 4-5 (pp 108-118).

[9] See: www.atl.external.lmco.com/projects/ontology/i3con.html