

The Many Faces of Mapping and Translation for Semantic Web Services

Mark H. Burstein¹
BBN Technologies
burstein@bbn.com

Abstract

Semantic web services hold the promise of greatly increasing interoperability among software agents and web services by enabling content-based (as opposed to format-based) automated service discovery and interaction. However, as different services may well use different, only partly compatible ontologies to describe their capabilities, some amount of ontology mapping or translation will be required during the various stages of service discovery and utilization. In this paper, we reexamine some of the processing assumptions that were made in the development of semantic web service models like DAML-S in order to uncover the very different roles of semantic translation in the subprocesses of service discovery, service process model interpretation, task negotiation, service invocation and response interpretation. We present examples and arguments showing how several different styles of translation will be required based on the informational context and perspectives of agents in each of these stages.

1. Introduction

It was a clear goal of early envisioners of the semantic web [1] that semantic descriptions would one day be used to describe web services so that other software agents could use them without having any prior 'built-in' knowledge about how to invoke them, just as people can make effective use of web sites found using a search engine. The technology would ultimately enable individuals to use their own personal software agents for such things as comparative shopping, information discovery, or to weave together new services from available ones, much as a travel agent does. RDF and DAML/OWL were designed to support the development of many different inter-related ontologies, each published on the web to be shared within communities and across communities that have a strong need to

interact. Ultimately, though, increasing interoperability means having an ability to reach out to communities with services that were developed independently, using different ontologies. If the communities share the same syntactic language for defining ontological terms (e.g., RDF or OWL), then these communities can be bridged by reading the published ontologies of the other and, where necessary, defining correspondences or more complex mappings between related terms. It is the question of *how* to use such mappings that is the subject of this paper.

Semantic web services are web services that can be dynamically discovered, applied and composed, by reasoning from published representations of their *service descriptions*, expressed in a declarative semantic web description language, such as DAML [2] or its emerging standardization as OWL [3]. Once services are described by published semantic representations, using shared ontologies of concepts and relations, other software systems (agents) can reason about how to invoke the services dynamically by reading these descriptions at run time, composing messages with the prescribed content, and sending them using WSDL [4] message templates. This is in contrast to distributed object models where the invocation interfaces are loaded at compile-time, and a programmer is required to identify how the proper message content is copied from the local environment and passed to the remote service using the specified interface.

DAML-S [5,6,7] is a specific DAML ontology designed to provide a framework for just such descriptions of web services. DAML-S was designed to serve as the basis for representing descriptions of services from several perspectives, including the discovery and selection of an appropriate service (service profile), the declaration of the inputs, outputs and effects of the service (process model), of pre-defined patterns of service invocation (process model) and the formulation and interpretation of messages to that service by mapping (grounding) inputs and outputs to messages in a model such as WSDL.

¹ This is a slightly edited version of the paper that appears in WISE'03 (Rome, December, 2003)

The basic idea behind DAML-S is that a service developer can describe his or her service using the abstract procedural concepts provided by the DAML-S ontology, by combining it with other domain specific DAML ontologies developed by the communities served by the service. These domain ontologies will provide the terms used for different classes of service inputs, outputs, and effects, building on other basic ontologies providing support for describing common elements like representations of time and space, people, organizations, commercial transactions, etc. DAML, by extending RDF [8], is designed to allow ontologies developed by different people or groups to build on each other in this way.

There is, however, a catch. When the software agent that is going to invoke a service is developed by one organization, using internal data models described in terms of one ontology, and the just-discovered service to be invoked is developed by another organization using a different ontology, the ontologies may use different terms for the same or similar notions. In such cases, translation is required, even though both software systems seek to support interoperability by using DAML or OWL ontologies to describe themselves and the data they can reason about. To make ontologies interoperable, *ontology mappings* are required, so that the terms in the different ontologies are brought into correspondence. Sometimes these mappings can only be partial, as ontologies developed for different purposes may not have needed to define all of the same concepts, or may have incompatible definitions. The descriptions of such mappings are sometimes called *articulations* [9], or *bridging axioms* [10] and can include simple correspondences between terms (uni or bi-directional), rules ‘defining’ terms in one ontology in terms of some set of terms in another, and even functional mappings, such as for translating units of measure. Articulations are typically developed using semi-automatic tools that are integrated with ontology development tools or editors [9,11,12,13] although this is still an active area of research. For the purposes of this paper we will ignore the question of how these mappings are developed.

Much of the work on development of ontology mappings was initiated in support of distributed information systems (e.g., [14,15]), where the objective is to support access to heterogeneous information sources, and make the results of queries appear to be coming from a single integrated source. Because of the many potential incompatibility problems that can arise in relating ontologies developed independently [16], the approach commonly used was to develop a master ontology that makes all of the distinctions found in the original models of the data sources [17]. When this approach is successful, queries can be uniformly

expressed in terms of the master ontology, and interpreted locally by each data source. Results of database access can be translated back into that ontology. This largely eliminates one of the potential problems of translating queries. Because query translation is translation in the opposite direction from the translation of the query result data, and because the proper translation of a query may require knowledge of both the structure and content of a data source, it can be difficult for a requester or middle agent to complete the translation. This is an example of the kind of problem discussed below, namely, that the process of translating a service request may be very different than the process of translating the result of the request back to the requester, making it difficult or impossible to define a general purpose middle-ware “translation agent” for many domains.

This paper looks at the different processes involved in dynamic utilization of semantic web services with an eye toward the knowledge needed for effective translations between representations that may be needed to support those processes. We begin by reviewing the different aspects of semantic web service descriptions using the DAML-S model. We then look at some of the assumptions about the process of making semantic web services interoperate dynamically that were part of the rationale for that model. Finally, we look at the differing kinds of knowledge and reasoning required to translate requests and responses between semantic web services and the agents invoking them, and draw some conclusions about *where* translation may need to occur in each case.

2. Describing Semantic Web Services with DAML-S

DAML-S is, effectively, an “upper ontology” described in the DAML language, comprised of a set of abstract classes that can be specialized by service developers to describe their web services from several different perspectives. Its purpose was to form a common core set of terms (concepts, relations) on which service developers could build to describe their own services. DAML-S provides a framework within which specific services can be described and reasoned about, but does not address (or need to address) specific domain issues (e.g. product taxonomies, types of inputs, etc.) as these ontologies can developed, adopted and shared among service developers and consumers within the communities using them. These domain specific elements are intended to be inherited from other DAML ontologies found on the semantic web, or defined as

needed and published so they can be shared and interrelated with other ontologies.

DAML-S models services as consisting of three basic abstract components. The *service profile* part of DAML-S defines a general class of description of services intended for publication in a registry or matchmaker [18]. A DAML-S matchmaker [19] is analogous to and can be built as an extension of a yellow pages service like UDDI [20] that can interpret and compare semantic descriptions of services to find candidate services satisfying a particular need. The DAML-S ontology for service profiles provides an extensible set of properties for describing the purpose of a service, and other features that might be relevant discriminators between services from the perspective of potential users of that service. The DAML-S *service process model* describes the set of activities that a potential consumer of the service might request, their inputs, outputs, conditions of use, and possible effects. It also provides a composition language so that a service provider can describe ways that individual service processes are intended to be composed. A *service grounding* describes the relationship between processes and communications by specifying mappings between (e.g.) process inputs and outputs and portions of messages in a language such as WSDL.

The DAML-S notion of an *atomic process* can be viewed as an abstract representation of a WSDL operation, where the inputs and outputs of the process map onto parts of WSDL messages. One of the key distinctions between the models, and the reason to maintain both and the mapping between them is that WSDL message parts are defined locally to the WSDL service definition, with no means of providing their semantics relative to a set of broadly accepted terms. In contrast, DAML-S process inputs (for example) are typed by URI references to classes that are elements of shared semantic web ontologies. This key difference enables services defined using DAML-S to be interpreted by any agents that can reason *dynamically* about the relationship between the type of information required to formulate a service request and the specification, held internally to the agent, of the goal to be achieved. Once the *information* required to create a request has been identified, DAML-S provides a mapping to WSDL (the grounding) to enable the request to be turned into a SOAP or HTML format.

The basic use model envisioned for DAML-S consists of the following steps, not all of which are required in all cases. This model is shown in Figure 1

1. A service advertises itself by publishing a service profile – either by sending it directly to a matchmaker or as a web document that could be scraped by such a service.
2. A software system (we will call these agents) requiring a service to achieve some *explicitly represented purpose* (goal) poses a query to a matchmaker with a pattern that represents a class of services capable of achieving the goal. If the agent can do planning, it may decompose the goal and find different services to achieve parts of the overall purpose.
3. The matchmaker compares the query to its library of service profiles and returns the profiles of candidates that could produce the desired effects, including pointers to those services' process models and groundings (stored on the WWW).
4. The agent selects the best candidate by considering additional information in the profile (such as locale, quality of service, types of inputs required).
5. The agent maps its goal onto the set of inputs to the server process.
6. The agent uses the process grounding to map the inputs onto WSDL message(s). The messages are sent to the server.
7. The service determines that it can perform the request, and (usually on completion) sends a reply message.
8. The reply is parsed using a WSDL output message template.
9. The output message is mapped (again by a process grounding) to the set of output properties of the DAML-S process.
10. The agent reasons from the outputs as to whether its goal has been accomplished, and what following process should be invoked.

Viewed abstractly, this use model bears a resemblance to the web services model envisioned for UDDI-WSDL-SOAP, except that the whole process can be performed automatically (without human intervention) using services whose APIs that were not known to the agent prior to their being interpreted at execution time. The software agent is responsible for the interaction with the matchmaker, the interpretation of which candidate services are most appropriate, the determination of the information required to invoke each service, and the interpretation and response to messages returned by the service.

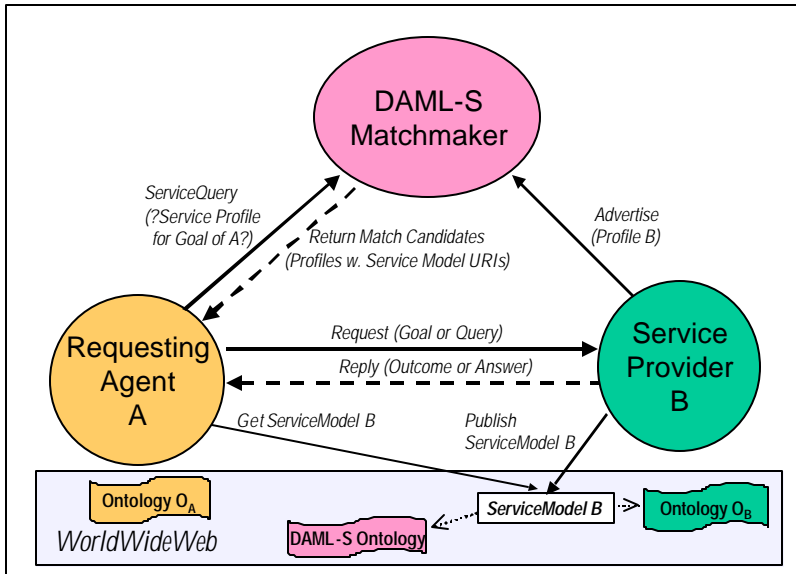


Figure 1: Basic DAML-S Semantic Web Service use model.

Variations on this execution model for DAML-S use have been implemented at BBN and at a number of universities (e.g., [21,22, 23]). For the most part, these models have all ignored translation issues, by assuming a shared set of ontologies. Yet, translation will clearly be required to achieve the broad interoperability envisioned for semantic web services, and we need an architecture for client and middle agents that supports it. For services to be discovered and utilized by software agents developed independently, they must both reason and communicate using the same ontology or else there must be some agent, be it the service itself, the consumer, or a middle agent, to translate the between them.

The question we seek to address in this paper is: where does translation play a role in this model of service discovery and invocation when different ontologies are used by different agents, and how does the *information* required to do the translation in each case impact who (which agents) can perform most readily perform that function? If a translator needs information that is local to a client or service provider, rather than just the ontologies used by the agent receiving the translation, then we have probably given the wrong agent the job of translation. We argue that, in many cases, we must consider, in addition to the ontologies of the sender and receiver and the mappings between them, the sources of domain-specific information required to formulate messages fully.

3. Translation for Service Invocation

Consider, as an example, the situation sketched in Figure 2. In this example, I want to use my personal

agent to help me buy a book on the web. My agent knows a number of things about me and my possessions, and it knows how to shop for me by contacting semantic web services. Assume for now that I have told it that I want a book on XML, and it has used a matchmaker to find a reputable service called Books4Sale. Books4Sale, has published a process model which declares that I can accomplish my goal of owning the book, provided that it is in its inventory. The process (we will assume it is an atomic process – the DAML-S term for a simple one-step request/response operation) has as inputs the book title, author, credit card and shipping information. It produces as output (if successful) an order confirmation number and shipping tracking number (not shown).

At first blush, it would seem that MyAgent can now achieve my goal by translating its representation of my goal (internally described using its local ontologies **mylife** and **mystuff**) into the form described by the inputs to the Books4Sale’s process description. The Books4Sale service description would be expressed using DAML-S and the ontologies of the book vendor’s business, although we have not used the correct syntax to simplify the figure. In fact, however, my goal, owning a book, corresponds to a stated *effect* of the service process description, that I own whatever book was requested, when the process has completed successfully. A classical AI planning system would expect just such a correspondence, and MyAgent would use such a planning process to reduce the goal to an action, namely to execute the process described by sending the expected message to that service. (For technical issues involved in this kind of planning, see [22, 24, 25].)

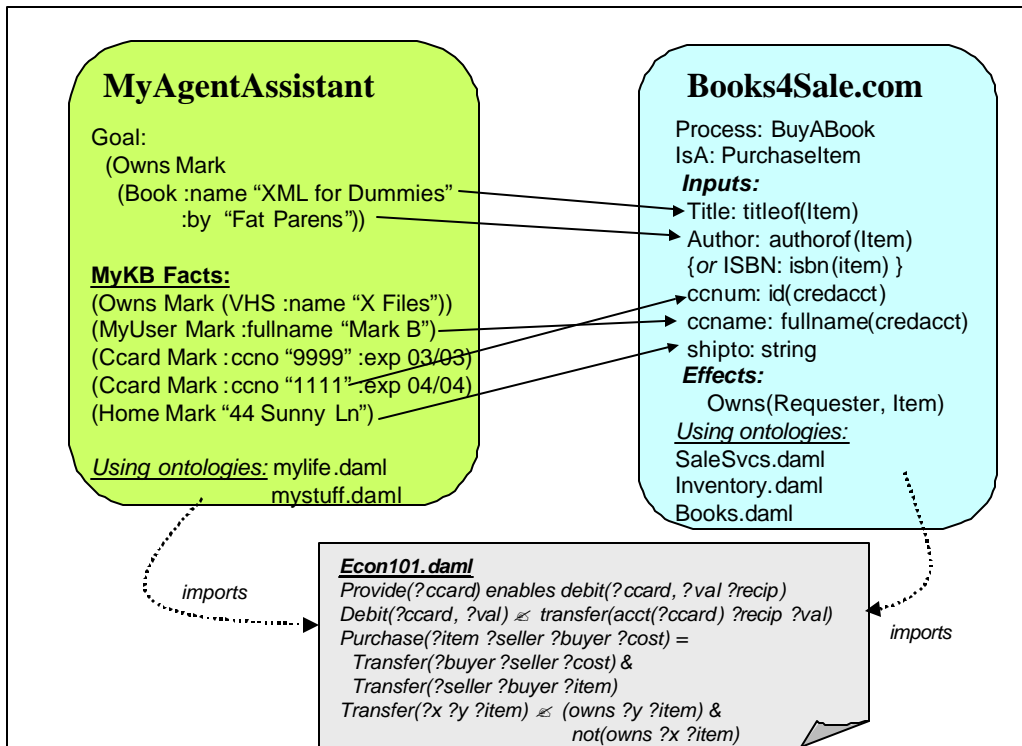


Figure 2: Translation of a Request.

DAML-S was designed to support this kind of reasoning by providing a standard way to associate effects with represented processes, so that potential service clients could treat service processes as *descriptions of planning operators*. MyAgent can thus reason that by invoking the BuyABook process with the required attributes of the desired book (title, author) specified among the other required process inputs, the process will succeed, because the item owned after the process completes corresponds to the book in my goal.

It is at this point in the agent's reasoning that translation is required. The BuyABook process requires an input called *title* whose value is constrained to be the *titleof* the item to be purchased (owned). To fill this input requirement, the agent needs to query its own knowledge base for a particular property of the concept for book in its ontology that can be translated to the property *titleof* on inventory items of the bookseller. Assuming that this articulation exists in a published model relating the two ontologies, then it can determine that it needs to retrieve the *name* property of the book sought and provide that value in its request message.

This almost trivial example shows how tightly intertwined the use of articulation or mapping knowledge can be with the reasoning required to formulate service requests. We would argue that examples like this show why, as a practical matter, service requests are not formed

and then translated, but should be created by the service requester by *plan formation reasoning interleaved in the presence of articulation rules*. Effectively, the planning process must decide which process to invoke (based on effects) and then plan (generate) the message by gathering the inputs required, translating constraints on what those inputs are into its own ontology along the way.

A second aspect of the example will help to reinforce this point. In addition to the title and author, the BuyABook process requires inputs which in its process model are called *ccnum* and *ccname*, the values of which are as the IDnumber of a credit card account and the Fullname associated with the account. As these elements of the service's process inputs do not correspond to any elements of the agent's goal (that I own a book), they must come from the agents general knowledge about me as the request originator, and one of the standard plans for gaining ownership – purchase by credit card. The agent's knowledge of web purchasing procedures could include the fact that one step in the process is providing the credit card information to the vendor. If translation is required here, it is required to discover which properties in the agent's internal representation of the user's credit cards can be mapped to the required properties of the vendor's concept of credit card, so that the necessary values can be retrieved.

By analogy, we can view dynamic service request formulation, given a set of required inputs to a service process model the agent has just read for the first time, as being like filling in the individual elements of a form, where the type of value to be supplied in each field on the form must be interpreted from the form designer's description, and translated into an internal query to the form filler's knowledge base. Some fields will ask for things that further specify the requester's goals, others may ask for personal information needed to further the process.

One further conclusion we can draw from this scenario is that because the requester's ontology may not have a prior representation of the kind of request required with precisely the right elements (as a class or schema or an interface in programming terms), the request could not have been generated first in the sender's ontology and then translated by an intermediary. If the sender cannot do the translation while reasoning from the published process model, then a plausible alternative approach would be that a middle agent interprets the model and translates the descriptions of what is *required* for each input, and provides this to the sender as a set of queries to be answered in order to complete the message. Then the middle agent could prepare the final message and send it, or provide the mapping of query answers into a message format, such as in WSDL. This is effectively the role that tax return preparation services provide for people. It is an open question how interactive this would need to be.

4. Translating Responses to Service Requests

Perhaps the most well understood kind of translation is that of translating responses to requests or queries. Here, the information to be provided is known by the service provider, and the issue is one of identifying the mapping of this information into the recipients ontology. Most of the work on heterogeneous information retrieval is focused on the problem of developing the necessary ontology mappings. In most cases, response translation can be done by any agent that has access to the source and target ontologies and the necessary articulation rules. This could be the sender, receiver or a middle agent. There are some circumstances where this is not the case, however.

In previous work [26] we described an approach to message translation between agents based on the idea of generating special purpose translation code that would translate specific classes of structured data sets between representations used by different agents. The approach assumed that the ontologies of both agents were available, along with axioms that related the two, or mapped them into a shared abstract ontology. For example, if agent A was to answer a query Q by agent B,

but the two used different ontologies, then our translation middle agent would develop a specific translator for the class of data sets D_A represented using ontology O_A that resulted from queries like Q, into a data sets of class D_B , represented in the target ontology O_B . This special purpose piece of code could then be stored in a middle agent, or provided to either the sender or receiver to make that particular class of message translate efficiently in the future.

The approach was sufficient to handle cases where the source and target ontologies were different in quite substantial ways. For example, target ontology might only be capable of representing a summarized representation of the data being provided (such as a count or histogram of similar items over time) or it might not represent related objects in as much detail, perhaps only caring about their names. Items that were represented as classes of objects in one system might be represented as instances in another (e.g., a particular book with a title and author would be an instance of BOOK to a potential purchaser, but would represent a type of book with an inventory quantity to a vendor).

The potential difficulty with this approach as it might apply to web services is that it relied on being provided a specific target representation for the data to be translated, rather than an ontology. The code that was generated mapped a particular schema in one ontology into a different schema represented in a second ontology. In contrast, services or agents may generate very different responses to a request, depending on their internal state. For example, a request to purchase something may result in a confirmation message or an "out of stock" message. Or a query might result in a response consisting of a heterogeneous list of object descriptions. Generating translations of such variable responses is not easily done using stored procedures, and so recent work described in [27] has used a first order reasoning system to perform the message translation. Somewhat similar techniques have also been used in [13,28], although the latter is uses a more syntactic kind of transformation rule.

The main issue for translation of descriptions (query results) is still the ability to represent and reason about the relationships between the ontologies, and whether that is sufficient to translate correctly. There are cases where access is required not only to the ontologies but to the content of the knowledge bases represented using those ontologies. Gio Wiederhold frequently tells the story that there are different administrative offices at Stanford University that have slightly different models for who qualifies for membership in the class Employee. One includes only salaried employees, while the other is broader, and includes (for example) Emeritus Professors. If the distinction between these classes could not be represented precisely, because neither knowledge base

contained the attribute that represents the distinction, and the translation rule could not enumerate the exceptions, how would one decide if an approximately correct translation rule adversely impacted the translation of data about a particular set of employee records or not? The only way to tell for sure would be to have access to the enumerated set of instances in the target and see if everyone whose information is being transmitted is in that set.

5. Translation Issues for Dynamic Service Discovery

The next case to consider is web service interactions with a semantic web service matchmaker or registry. Recall that a service matchmaker receives advertisements called *service profiles* from services that wish to be utilized, and client agents looking for services of that kind query the matchmaker with a general description of their goal *represented in the form of a partial or abstract profile* that can be matched against those in the registry. Candidate services are returned to the requester to be further filtered until a suitable service is identified.

The key issue here is the question of what ontologies are used to represent the service profiles. If a matchmaker is to cover a broad range of services, there are two likely possibilities. The first is that the registry has its own set of ontologies that helps to organize the services in a single framework. The other is that every service uses its own ontologies to describe itself, with only a small amount of common structuring provided by the abstract definition of a profile as provided by an ontology like DAML-S. Let's call the first approach the Yahoo approach and the second the Google approach.

With the Yahoo approach, each service developer must identify the concepts in the shared ontology they can use to best describe their service. Distinctions that cannot be made will be lost, unless there is a mechanism by which the ontology can be extended consistently. Using this model, an agent needing to find a service must translate its goal into a description of a type of service that can be represented by the matchmaker's ontology. If a DAML-S profile framework is used, then this means finding both a concept for the class of service desired, and potentially representing critical outputs or classes of acceptable outputs or effect.

One type of problem we have encountered in using this approach, even without translation, is that of circumscribing the range of acceptable outputs. For example, if one wants to find a particular kind of restaurant in Rome, such as Chinese food, is it better to ask for a service that will provide listings of Chinese Restaurants in Rome, or just restaurants in Rome, or

Chinese restaurants in Italy, etc. The point that is determining *how to translate* depends on both the ontology of the matchmaker and how it matches queries against candidate services.

With the Google approach, a somewhat different set of translation issues arises. Here, potentially every new service profile that is added to the matchmaker's repository uses somewhat different ontologies. The problem then arises as to how any agent querying the matchmaker could even begin to translate its query, since the matchmaker cannot even advertise what ontologies to use. In this case, it must become the responsibility of the matchmaker to do any translation that it can *each time it matches the query against a candidate service*. If it has no information about articulation rules relating the query's ontology to the service profile's ontology, then the query will only match if they happen to have used the same ontology, at least in some aspects.

These are, of course, extreme characterizations of the translation issues to be faced when using semantic representations for service profiles. It is still an open, empirical question as to the size and diversity of services that can be handled using a semantic matchmaker, and how to strike an appropriate balance between the diversity of ontologies used for matchmaking and the level of translation supported in the process. To date, the only work I am aware of that has begun to address the question of translation during matchmaking is described in [29].

Summary: The Impact of Knowledge Locality on the Locality of Translation Processes

This paper has reviewed some of the assumptions and motivations behind the development of systems for supporting semantic web services, and, in particular the approach taken with DAML-S. As a key incentive for semantic web services is increased and more dynamic interoperability among agents and services, it is important to look at the role that translation will play in mediating among services using the variety of ontologies that will be supported on the semantic web. Our focus was on how the process of developing translations depends critically on the particular function being supported. In particular, I have presented suggestive examples to argue that translation of requests, in the presence of a published process model can be very different than translation of service results back to the requestor, up to and including which agents or services might be capable of doing the translation. I have also suggested how different models of matchmaking for semantic web services will engender different requirements for who and what to translate about service profiles.

The question of how and where translation happens has an impact on the design of architectures to support semantic web services. These questions are among the subjects of discussion in the Semantic Web Services Initiative Architecture committee. I hope that this discussion has helped to make clear what are some of the issues involved.

Acknowledgements

I wish to thank my colleagues Drew McDermott of Yale University and Doug Smith at Kestrel Institute for many, many conversations about semantic translation for agents and semantic web services. I would like to thank as well the members of the DAML-S consortium and the SWSI for discussions on related topics. This writing of this paper was funded under a contract with Yale University to the DARPA DAML program, contract number F30602-00-0600, and BBN contract F30602-00-C-0178 to the same DARPA program.

10. References

- [1] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, 2001.
- [2] Hendler, J. and D. L. McGuinness, “The DARPA Agent Markup Language.” *IEEE Intelligent Systems*, 2000. 16(6): p. 67-73.
- [3] Dean, M., Connolly, D. van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneider, P., and Stein, L. “OWL Web Ontology Language 1.0 Reference” <http://www.w3.org/TR/owl-features/>, 2003.
- [4] Christensen, E. Cubera, F. Meredith, G. and Weerawarana, S. “Web Services Description Language (WSDL)” <http://www.w3.org/TR/owl-ref/> 2002.
- [5] DAML-S: <http://www.daml.org/services/>
- [6] The DAML Services Coalition (A. Ankolenkar, M. Burstein, J. Hobbs, O. Lassila, D. Martin, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne and K. Sycara), “DAML-S: Web Service Description for the Semantic Web”, *The First International Semantic Web Conference (ISWC)*, Sardinia (Italy), June, 2002.
- [7] DAML Services Coalition (alphabetically A. Ankolekar, M. Burstein, J. Hobbs, O. Lassila, D. Martin, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, H. Zeng), “DAML-S: Semantic Markup for Web Services”, in *Proceedings of the International Semantic Web Working Symposium (SWWS)*, July 30-August 1, 2001.
- [8] Lassila, O. and Swick, R. “Resource Description Framework (RDF) Model and Syntax Specification” <http://www.w3.org/TR/REC-rdf-syntax> 1999.
- [9] Mitra, P., G. Wiederhold, and M. Kersten. “A Graph-Oriented Model for Articulation of Ontology Interdependencies” In *Proceedings Conference on Extending Database Technology 2000 (EDBT'2000)*. 2000. Konstanz, Germany
- [10] McDermott, D. Burstein, M. and Smith, D. “Overcoming ontology mismatches in transactions with self-describing agents.” In *The Emerging Semantic Web: Selected Papers from the First Semantic Web Working Symposium*, pp.228- 244. 2002.
- [11] Noy, Natalya F. and Musen, Mark A. “Evaluating Ontology Mapping Tools: Requirements and Experience” SMI Report 2002
- [12] Noy, N. F. and M. A. Musen. “PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment”. In *Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*. 2000. Austin, TX
- [13] Mitra, P., Wiederhold, G. and Jannink, J. “Semi-automatic integration of knowledge sources.” In *Proc. Of the 2nd Int Conf. On Information FUSION'99*, 1999.
- [14] Sheth, A. P. and J. A. Larson, “Federated database systems for managing distributed, heterogeneous, and autonomous databases.” *ACM Computing Surveys*, 1990. 22(3): p. 183-236.
- [15] Siegel, M. and Madnick, S. “A metadata approach to solving semantic conflicts.” In *Proc. Of the 17th Int. Conf. On Very Large Data Bases*, pp. 133-145, 1991.
- [16] Klein, M. “Combining and relating ontologies: an analysis of problems and solutions.” In *IJCAI-2001 Workshop on Ontologies and Information Sharing*. 2001. Seattle, WA
- [17] Buneman, P, Davidson, S. and Kosky, A. “Theoretical aspects of schema merging,” In *Proc of EDBT '92*, pp152-167. Springer Verlag, March 1992.
- [18] K. Sycara, M. Klusch, S. Widoff, and J. Lu. Dynamic service matchmaking among agents in open information environments. *ACM SIGMOD Record (Special Issue on Semantic Interoperability in Global Information Systems)*, 28(1):47–53, 1999.
- [19] Paolucci, M., Kawamura, T., Payne, T., Sycara, K. Semantic Matching of Web Services Capabilities. In *Proceedings of the 1st International Semantic Web Conference (ISWC)*, 2002.
- [20] UDDI. The UDDI Technical White Paper. <http://www.uddi.org/>, 2000.
- [21] Marta Sabou, Debbie Richards, and Sander van Splunter “An experience report on using DAML-S,” In *The Proceedings of the Twelfth International World Wide Web Conference Workshop on EServices and the Semantic Web (ESSW '03)*. Budapest, 2003.
- [22] Mithun Sheshagiri, Marie desJardins, Tim Finin, A planner for composing services described in DAML-S,” In *AAMAS Workshop on Web Services and Agent-Based Engineering*, 2003.
- [23] Massimo Paolucci, Katia Sycara, and Takahiro Kawamura, “**Delivering Semantic Web Services**” *Tech Report CMU-RI-TR-02-28* Robotics Institute, Carnegie Mellon University, December, 2002.
- [24] McDermott, D. and Burstein, M. “Extending an estimated-regression planner for multi-agent planning.” *Proc. AAAI Workshop on Planning by and for Multi-Agent Systems* 2002.
- [25] McDermott, Drew “Reasoning about autonomous processes in an estimated-regression planner.” *Proc. Int'l. Conf. on Automated Planning and Scheduling* 2003
- [26] Burstein, M., McDermott, D., Smith, D. and Westfold, S. “Derivation of glue code for agent interoperation.” *J. Autonomous Agents and Multi-Agent Systems*,6:265-286. 2003.
- [27] Dou, D., McDermott, D. and Qi, P. “Ontology Translation on the Semantic Web.” To appear, *Proc. Int'l Conf. on Ontologies, Databases and Applications of Semantics* 2003
- [28] Chalupsky, H., “OntoMorph: A translation system for symbolic knowledge,” In *Principles of Knowledge Representation and Reasoning: Proceedings of the Seventh International Conference (KR2000)*, A. G. Cohn, F.

Giunchiglia, and B. Selman, Editors. 2000, Morgan Kaufmann Publishers: San Francisco, CA.

- [29] Mandell, D. J. and McIlraith, S. A. "A Bottom-Up Approach to Automating Web Service Discovery, Customization, and Semantic Translation," In *The Proceedings of the Twelfth International World Wide Web Conference Workshop on EServices and the Semantic Web (ESSW '03)*. Budapest, 2003.