Context Building Information-Centric Decision-Support Systems

Steven J. Gollery
California Polytechnic State University (Cal Poly)
San Luis Obispo, California
Cal Poly State University
One Grand Avenue
San Luis Obispo, CA 93407, USA
E-mail: sgollery@cadrc.calpoly.edu

Abstract

As the volume of data and human-centered information available to decision-makers continues to increase at an ever-accelerating rate, the need to represent information in software-processable formats becomes more apparent. At the same time, the availability of information from diverse sources through the World Wide Web provides the opportunity to widen the scope of input to decision-support systems, if this information can be made accessible through automated means. Past approaches to information-centric interoperability have been based on the use of a shared static object model, but this becomes impractical when we consider the loosely-coupled decentralized nature of the Web.

This paper discusses the motivations driving a change from static to dynamic information models. It defines a representative use case, and describes a service-based architecture that allows for extending existing information sources to allow programmatic access. The proposed architecture uses existing and emerging Web Service specifications, enhanced by an ontology definition language, to create an environment that does not require information service providers to use static shared models, while allowing information consumers to learn ontologies from the services themselves. Clients such as decision-support systems can thus build their own information context at run-time based on models received from multiple sources.

Introduction

There is gathering momentum towards distributed systems that interoperate by exchanging ontology-based information instead of data expressed in standardized formats. The use of ontologies can provide a context that enhances the ability of software to reason about information received from outside sources.

In the past, approaches to inter-system communication have relied on agreements to use pre-defined formats for data representation. Each participant in the communication then implemented translation from the communication format to its own internal information model. While relatively simple to construct, this approach led to distributed systems that were brittle, static, and resistant to change.

It is the premise of this paper that, for large scale ontology-based systems to be practical, we must allow for dynamic ontology definitions instead of static, pre-defined standards. The need for ontology models that can change after deployment can be most clearly seen when we consider providing information on the World Wide Web as a set of web services augmented with ontologies. In that case, we need to allow client programs to discover the ontologies of services at run-time, enabling opportunistic access to remote information. As clients incorporate new ontologies into their own internal information models, the clients build context that enables them to reason on the information they receive from other systems.

The flexible information model of such systems allows them to evolve over time as new information needs and new information sources are found.

This paper describes an approach to this problem through the use of an ontology definition language for communicating information models. We discuss the use of this approach in a web-service environment, and define a specific application domain where these ideas may prove useful.

Organization of Paper

The paper begins with a discussion of the background of and motivation for the problem to be solved.

The next two sections consider two new concepts for the World Wide Web: the Semantic Web and Web Services. Following this is a section about synthesizing these two ideas to produce Semantic Web Services.

The "Initial Problem Statement" defines the application domain that we are considering as a concrete situation where we expect that our proposed solution will be useful. The following section describes the concepts and architecture of our proposal.

Background and Motivation

For over a decade, the Collaborative Agent Design Research Center (CADRC) at Cal Poly, San Luis Obispo (CA), has created decision support systems that use ontology-based object models to represent information in a form that allows the construction of agents with reasoning capabilities. This provides the basis for collaboration between the user and the system to solve problems while avoiding pre-planned, inflexible solutions.

The ability to develop systems that share a common object model also enables high-level, intelligent interoperability at the "information" level, as distinct from the more common data level inter-process communication.

All previous and existing CADRC systems use a static, pre-determined ontology. That is, the ontology is defined during system design, and becomes an integral part of the system itself. This approach ensures that all elements of the distributed system share exactly the same semantics (or "understanding") of the shared information, and in most cases is a very practical methodology.

However, the CADRC also sees that some types of systems may benefit from the capability for the system or the user to change or extend the ontology after deployment. Such a capability is called a *dynamic ontology* or an *extensible ontology*. Extending ontologies may be useful in many situations, such as :

- An individual user may have unique information needs. In this case, it would be reasonable to change the model for that user without affecting all other users. Such changes may be initiated by the user or by system designers.

- The functional and domain models of a system inevitably change over time. This can become a serious problem when systems from multiple organizations are required to interoperate. By basing architectures on dynamic ontologies, we expect to minimize the effect of change on existing fielded systems.

- Static ontologies require that designers of a client program identify all potential sources of information prior to deployment, so that the client ontology can include all the concepts and relationships from those sources. But some kinds of systems (particularly in the web service world described below) require the ability to incorporate new sources of information in an ad hoc fashion, which can only be supported if the client's ontology can change at run-time.

All these argue against the idea of systems whose ontologies or object models are fixed during design and development. For systems to accommodate change, we need to make the models fluid and plastic. They must be modifiable at run-time either due to accessing object models of other systems, or due to direct user interaction.

Context of the problem

The general situation that this paper considers may be described in the following terms : *There is at least one client program that consumes information from multiple systems. Each information server has its own information model and was developed independently of all other servers. Client programs cannot be limited to a pre-defined set of servers, but must be allowed to incorporate information from unexpected sources as the needs of the user change.*

One method of enabling communication among systems is to begin by producing an interface requirements document that includes a precise specification of the format for the data or information that will be transmitted from one system to another.

Such interface requirements agreements tend to suffer from two limitations: first, there must be one such agreement between each pair of participating systems; and second, the agreements are rigid and must be renegotiated each time the needs or capabilities of one of the systems changes. In the worst case, interface agreements that assume static data formats and requirements can reduce the ability of individual systems to evolve quickly to meet the changing needs of their users.

One frequently-seen alternative to pair-wise agreement is to create a new system that manages communication between all participants. In other words, instead of interacting directly, all information flows into a centralized server. This arrangement can reduce the number of interface agreements, since each information source must only communicate with the new server. However, this approach can also create a bottleneck, for precisely the same reason: all information must pass through the central server.

Pre-defined interface agreements also do not allow inclusion of new information servers into an existing community of clients and servers, unless the new servers first work through the time-consuming process of defining and implementing one or more interface agreements. This becomes a problem when access to new sources of information can mean the difference between success and failure in a situation where timely response is a requirement. This paper will describe an example of such a situation in a later section.

In order to meet all requirements, any solution must include a way for information consumers to discover the domain and functional models of each information provider at run-time, rather than requiring each provider to implement a single standard interface. Such a solution will allow the construction of services that do not assume use by specific clients, and client that do not assume that all information will come from pre-determined resources. Any acceptable solution also must minimize the amount of effort required for a service provider to join the system. While some effort is inevitable, there should be no major architectural redesign in order to make the service provider accessible to other system participants.

The next three sections describe emerging and planned capabilities for World Wide Web systems that will help to support the proposed solution of the above problem.

The Semantic Web

The first generation of the World Wide Web consisted of static web pages. Early in Web history, browsers added forms in order to provide limited communication from the user back to the web site. To make web pages more attractive to humans (and, in a very few cases, to convey information), animation and other types of automatic visual changes were added. Throughout the first generation, web sites and web developers were almost entirely concerned with defining the appearance of the page within a browser.

Second generation web development included dynamically-generated web pages. This was added for several reasons, among them: to deal with often-changing information while minimizing the amount of server-side rework; special user requirements, including customer relation management; differences in display technologies, such as palmtop vs. desktop; and so on. Changes to the Web to allow for dynamic content were almost entirely on the web server side: client programs such as browsers still received HTML markup very similar to the language they had processed in the first generation. From the client side, then, the content definition was unchanged.

In both first and second generations, most information was still intended for human consumption. Content was based on the human ability to construct meaning out of natural language. Programmatic access to web information was only possible when developers coded to the format of a given page layout. Typically, such programs would fail whenever the layout changed or the page moved.

Most web sites today are second generation sites: they produce some dynamically-generated content. But there are some sites that have taken the first steps toward a new generation of the web: what has been called (by Tim Berners-Lee and many others) the *Semantic Web*. [1]

The semantic web will build on the idea of dynamically-generated pages to provide machine-processable information. This will be made possible by the addition of "semantic markup" to web-based resources. In contrast to HTML markup, semantic markup will include definitions of concepts and relationships involved in the information being transmitted. This richer definition will allow client programs to process the contents instead of merely the markup.

In the relatively short history of the World Wide Web, then, we see a progression in the kinds of communication offered by web sites: (1) human-to-human communication (humans place web pages on a site, other humans read it); (2) machine-to-human communication (server-side software assembles information, formats it for the web, passes the result to humans to read); and, (3) machine-to-machine communication (software assembles information, formats it to make it machine-processable, passes the result to other software).

Client programs will receive information and process it into internal formats to allow representation and reasoning. Accessing many diverse sources of information will enable a client program to build a rich and complex information context that will include multiple relationships, many of them between pieces of information that come from different systems. The client program will be able, in some sense, to "learn" the domain of knowledge assumed by the systems with which it communicates.

It is the growing movement toward the semantic web that allows us to imagine that a future of distributed systems exchanging information in a decentralized and nearly world-wide environment might be practical.

Web Services

As the World Wide Web evolves to emphasize the exchange of information among software systems, there is a need for client programs to be able to find and connect to remote information providers in ways that are compatible with the nature of the Web itself. In particular, there can be no centralized control, organizations must be able to create services independently, and client programs should be able to access services on an ad hoc basis. To state this another way: creating and using web-based information services should follow the same pattern as creating and using web sites.

The emerging standard approach to this problem includes a number of specifications that together are known as "Web Services". The three fundamental methods in a Web Service system are: register; discover; and, bind. Each of these is discussed below.

Register: When an organization exposes services on its web site, the services are ready to use, but only by clients that know in advance where the service is and how to access it. This is analogous to human use of the web in the absence of search engines: we would only be able to get to web pages whose URL we know in advance and to pages that are directly or indirectly linked to those pages. In the web service world, this situation may be entirely acceptable: for instance, where the service is only intended for the use of a specific group of clients.

However, in other cases, a web service may be intended for more general use. We may see this in the future, for instance, where services are provided on a pay-per-use basis. To increase the number of potential customers, web service providers will *register* their services in a public registry. Registries provide the equivalent of a phone book for web services.

Discover: Once services have been registered, would-be service consumers can locate systems that provide relevant operations and information. Registry specifications include interfaces for various kinds of query, including industry type, name of service, key word descriptions, etc. The information in the registry includes descriptions of service endpoints: communication method and class definitions for input to and output from each operation.

The registry provides enough information for service consumers to discover web services dynamically and to learn how to access them. However, in the "Semantic Web Services" section (below), we discuss some important limitations of this capability.

Bind: Once a service consumer has located a relevant service in the registry, the consumer has the information it needs to use that service. Connecting to the service provider, passing correctly-formed input, and receiving the results, is called *binding* to the service.

One common misconception of the role of the registry is that it may appear to be a point of centralization. In reality, the registry is simply another service provider, no more "central" than any other service. There may be many registries available to any given service consumer, and the consumer's interaction with a registry is limited strictly to performing queries against the store of registration information. Communication between consumer and provider never takes place through the registry, and so the registry should never become a bottle-neck or limitation on system scalability.

Semantic Web Services

Although semantic web concepts and Web Service specifications were developed by substantially separate groups of researchers and organizations, there is now a growing understanding of the benefits of combining the two to form "Semantic Web Services". [3]

Semantic Web Services will combine the meaningful representation of information, relationships and context from the semantic web community with the dynamic discovery and binding of the Web Service specifications to create a system that will allow contextual discovery and use of web services.

In the areas of discovery and registration: for the type of registry queries defined by the Web Services specification are keyword-based. Although the registry does allow a service provider to state the (standard) taxonomy from which descriptive terms are drawn, taxonomies cannot represent the full range of relationships possible with an ontology.

If an organization registers its services, including a semantically-rich description of the service model, then service consumers can discover service providers based on reasoning about the context within which the service is defined. Moving to a semantics-based discovery process will further loosen the coupling between service consumers and service providers, allowing for greater flexibility across the system.

In the area of binding: when service operations are registered using current web service specifications, the parameter and return types are exactly defined. This is reasonable (even necessary) when the services are based on standard or well-known definitions: for instance, when an industry or organization has created interface specifications for all operations and object types in a given domain. However, the type definitions usually mean that a would-be consumer must include classes that can only be created at design time. This mitigates against the total flexibility of run-time discovery.

If, however, the service is defined using an ontology, it is possible for would-be consumers to learn the context in which the parameter and return classes are defined. This will allow the consumer to relate those classes to classes in its own information model, potentially allowing the consumer to construct and populate objects that can then be passed to the service, even though the classes themselves are not part of the consumer's internal representation.

The DARPA Agent Markup Language project is defining DAML-S, a DAML vocabulary for service definitions [4]. As the DAML-S specification matures, it should provide a standard way to define services based on an ontology. This will help to allow context-based service discovery.

In addition to defining the service model, ontology representation in the semantic web services world must have the ability to describe the domain model of the service. This is the area that this paper and the associated project explore.

Initial Problem Statement

For the purposes of this paper and the associated project, we consider the following (highly simplified) situation:

An emergency management planner has access to information from multiple organizations. These organizations may belong to different branches of government, and may include private-sector and non-profit entities. Because of this diversity, we assume that there will be no single information model across all organizations. We also assume that individual organizations may change the kinds of information they provide independently to all others. Additionally, new sources of information may become available over time. These new sources must be incorporated into the planner's assessments as early as possible.

All information providers have an on-line web presence that allows qualified users to have access to information. However, these web sites provide the "second generation" web experience described above: live, dynamic information, but intended only for human processing.

The emergency management planner would like a decision-support system to assist in monitoring an emergency situation, in locating and requesting available resources, and in responding to immediate needs. The current state of the information sources is inadequate due to the lack of automated access. Some means must be found to enable the decision-support system itself to be responsible for locating, receiving, and reasoning about the information in the sources, while improving the quality of assistance provided to the user.

To be practical, any solution must limit the scope of the changes required to enable existing information sources to participate in the new system. Large-scale changes may: (1) face organizational opposition; (2) cost too much time and money; and, (3) disrupt current users of the human-centered interfaces that already exist. A solution that can be implemented quickly without necessitating changes to each system's internal information model is more likely to be successful.

Decision-support for emergency management is only one of many situations where opportunistic incorporation of unplanned information sources in a timely manner may be important. Other examples include: stock market analysis; supply chain management; resource allocation; corporate knowledge management; and military command and control. In each case, an environment that allows information consumer programs to automatically locate and access information will substantially increase both the quality and the timeliness of the decision making process.

Proposed Solution

To solve the problem described above, this paper proposes the creation of semantic web services to allow programmatic access to the information available on multiple web sites.

These services will allow the decision-support system to discover the ontology of each service's knowledge model. The decision-support system will then be able to extend its own model to include the new concepts and relationships, and to reason about the new ontologies and their instances. The system will therefore provide more intelligent assistance to the user, and possibly behave more intelligently as it adds concepts from more ontologies.

Our proposal is that, instead of requiring agreement on a common data interchange format, participants in this system will agree to describe their models using a common ontology definition language. This serves two purposes. First, we eliminate the time and effort needed to create a standard for data interchange, since each organization is free to define its own ontology without obtaining a consensus from all participants. Second, changes to information models do not necessarily require changes to information consumers. Consumer problems implement a language processor, not a processor for a specific format. In a sense, the consumer "learns" about the data model for the other systems by parsing the definition language.

Since the interchange model is not built into any of the systems, this also enables inclusion of unexpected systems into the consumer's processing. All that is required is that at some point, the developers of the newly-included system must have implemented some way for the system to communicate its object model to a remote consumer using the ontology definition language, and that the services are advertised through an accessible service registry.

We hope to show that this approach can allow information providers to participate in this environment with little change to their existing systems, and with only minor dependencies on other organizations. Further, we expect that, due to the flexibility of ontology-based web services, decision-support system will be able to utilize information from previously unknown sources, and to survive changes to the models presented by existing sources.

Demonstration Project Concept and Implementation

In order to provide for extending an ontology, we need to consider several cases: (1) an information producer extends one or more ontologies that are already known to the consumer; (2) the user adds new attributes and relationships, possibly even new classes; and, (3) a consumer needs to work with several ontologies that do not share a common upper ontology.

From a system design point of view (and given that the ontology is defined as instances of classes in the ontology definition language, not static classes), the second case seems relatively simple: if the program presents the user with a comprehensible representation of the ontology, it will be simple to construct instances of ontology classes that represent the new attributes, relationships, or concepts. The difficulty lies, not in the logical processing, but in designing the user interface. For a user who is not trained in ontology modeling, a graphical or tabular representation of the model may be daunting or even incomprehensible. Finding an acceptable user interface will require experimentation. We expect that one possibility is to allow a (constrained) natural language interface, where a user would identify the class or classes to change, along with what kind of change to make, and the program would perform the modification.

The third case – that of merging seemingly-unrelated ontologies – seems by far the most difficult. Conceptually, this is a larger-scale version of the second case, but here the user requires much more support. We expect that tools to enable this most general use will be created only over a period of time. Early tools are likely to be aimed at the experienced ontology designer rather than the end-user. Such tools will gradually become more intelligent as we create new ways to reason about multiple ontologies. Additionally, successive iterations of the user interface will result in more intuitive usage patterns. Eventually, it may be possible for domain experts (instead of ontology experts) to perform merges.

The first situation, where the information provider extends a known ontology, turns out to be the simplest: there is no need for a user interface, and the provider's definition can be readily added to the known definition while the client is executing . We expect that this will be a common situation: client and server will be dealing with the same domain of knowledge, so as more ontologies are publicly specified, it is likely that service providers and consumers will utilize "standard" ontologies wherever possible. Since this is the most straight-forward possibility, we have begun implementation of our project assuming that this is the case.

In our proof-of-concept, we have three service providers, one consumer, and another client program that allows simulation of changing conditions. All participants use a common "upper ontology" that provides very general concepts from the domain of knowledge. Each of the service providers extends the upper ontology for its own needs. The user has the opportunity to extend the ontology in simple ways (for example, creating new relationships), but we have not yet addressed the important issue of intelligent tool support. We expect that explorations of this problem will be a major focus of future work.

The consumer discovers the service providers at run-time. From each provider, the consumer learns about the extensions that system adds to the upper ontology. The consumer adds those extensions to its own model, and is then able to perform simple reasoning on the extensions, utilizing the semantics of the upper ontology model. The second client program is then used to alter the state of objects contained by one of the service providers, and the consumer is able to infer from those changes that the state of objects in another provider will also to change.

At the time of this writing (April, 2002), the system architecture has been designed and developed, and early versions of all ontologies are in place. The service consumer program can discover the ontologies and perform basic reasoning on them. Within the next two to three months we expect to have a graphical depiction of some of the information involved, as well as to support a limited form of user-initiated ontology modification. Following this phase (which we consider the baseline), ontologies and client capabilities will be enhanced to form a more complete and realistic domain of knowledge.

Conclusion

This paper has defined the need to provide architectures for flexible interoperability based on information models rather than fixed, pre-defined data formats. It proposes an approach based on a combination of concepts from emerging Web Service specifications with ideas from the Semantic Web. The paper also described the proof of concept system that is in progress at the CADRC at this time.

As more organizations present services via the World Wide Web, we expect to see a transition from the human-centered web of today to a distributed system that primarily consists of software accessing machine-processable information. By adding the ability for service consumer programs to learn the information models of service providers, we enable these consumers to act as our surrogates in locating and reasoning about relevant information.

References

1. T. Berners-Lee, J. Hendler, O. Lassila. "The Semantc Web". in *Scientific American*, May 2001.

2. ---, "Planet Internet", in *Technology Review*, March 2002.

3. S. McIlraith, T. Son, H. Zeng. "Semantic Web Services". In *IEEE Intelligent Systems,* March/April 2001

4. G. Denker, J. Hobbs, D. Martin, S. Narayanan, R. Waldinger. "Accessing Information and Services on the DAML-Enabled Web". In proceedings of *Second International Workshop on the Semantic Web – SemWeb'2001*, May, 2001.

5. T. Sollazzo, S. Handschuh, S. Staab, M. Frank. "Semantic Web Service Architecture – Evolving Web Service Standards Toward the Semantic Web." In *Proceeding of the 15th International FLAIRS Conference,* 2002. AAAI Press.

Acknowledgements

The research described in this paper is jointly funded by the C3RP Program of the California Polytechnic State University, San Luis Obispo (Cal Poly), the Collaborative Agent Design Research Center at Cal Poly, and CDM Technologies, Inc.