The Role of Discovery in Context-Building Decision-Support Systems

Steven J. Gollery
Collaborative Agent Design Research Center (CADRC)
California Polytechnic State University (Cal Poly)
San Luis Obispo, CA

Introduction

There are many sources of on-line information available to those responsible for making decisions in complex situations. However, most of that information is either intended for human use or is available only in custom or proprietary formats. Both of these conditions reduce the ability of computer software to perform automatic reasoning about this information. As a result, the usefulness of on-line information in intelligent decision-support systems is currently limited to the few sources that are implemented specifically for those systems. The vast mass of sources that do not fit that description are virtually invisible.

Current practices for integrating multiple information sources have proven too costly to implement and have produced inflexible systems. These practices, which are discussed further below, cannot cope with the magnitude or the diversity of the available information.

This paper describes the TEGRID project, which demonstrates the use of a *service oriented architecture* to enable a more flexible, loosely coupled system of interoperable information sources and consumers.

Disaster Management Requires Information from Diverse Sources

To determine how best to manage disaster or emergency response, the decision-maker requires access to information from many sources. These sources might include law enforcement agencies, hospitals, ambulance services, the weather service, city traffic control, the National Guard, and so on. Each of these sources is controlled by different organizations, some non-governmental, some local, some state, and some federal; some organizations are civilian, some military. A decision-support system that assists users in planning responses to ongoing emergencies must incorporate information from as many of these sources as possible.

One often-used approach to the problem of inter-system communication is to create an interface agreement (i.e., an exact definition of the format that will be used to communicate data and information). Each system then creates a translation between its own internal data or information model and the format defined by the interface agreement. While this approach is conceptually simple, it poses problems at both the technical and the cultural levels.

It can be very difficult for multiple organizations to reach agreement on an interface format. Such agreements take time and a great deal of focused effort. At the same time, it can be difficult to arrive at common understandings of the information represented by each system. A single model representing all the kinds of information available in all systems may be so extensive that it becomes impractical to implement.

By far the most problematic aspect of the interface agreement approach is its lack of flexibility and extensibility. When any of the systems changes its internal model, this change may make the interface agreement obsolete. In that case, the agreement would have to be renegotiated and modifications would need to be made to all systems, with associated funding requirements. The level of effort required to change an interface once it has been implemented across multiple systems tends to create distributed systems that are brittle, static, and resistant to evolving quickly to meet the changing needs of their users.

Current approaches to inter-system communication too often result in tightly coupled systems. In extreme cases, the coupling becomes so tight that nothing can be changed in any individual system without requiring equivalent changes in other systems. Over time, the coupling of such systems tends to become tighter as more software is written based on assumptions about the exact format of communicated data and information.

Adding more information sources to such a system can also be difficult. As each new system is added, the communication among systems tends to become more complex. Each new revision of the interface agreement becomes more difficult to define, since any changes to the interface require changes to larger sections of the software. The result is a further slowing of growth and change.

This gradual reduction in the amount of change possible in a given period of time is especially injurious to decision-support systems. Decision-support systems, especially in the area of emergency management, should be implemented in a timely manner in order to provide assistance to their users as soon as possible. Furthermore, when users identify a requirement for change to the system, it must be feasible for developers to implement that change quickly so that new functionality is available when the decision-maker needs it.

These requirements, and the problems of the interface agreement approach, led us to consider a different type of architecture for our demonstration system. That architecture must provide (at least) two benefits over architectures that require tightly-coupled communications: first, the architecture must support the rapid addition of new sources of information; and second, the architecture must allow individual information sources to change their communication format while requiring little or no reworking of the systems consuming this information.

The proposed solution involves the loose coupling provided by *web services*, combined with the self-describing information model of the *semantic web*. Both of these concepts are discussed further in the following section.

Service-Oriented Architectures and Web Services

For our demonstration system, we explored the feasibility of replacing the tight coupling of earlier distributed systems with a more loosely-coupled architecture. We implemented a system based on web service standards that allowed us to use a discovery process to construct the system at run-time based on the information needs of the clients. The use of discovery enables each participant in the system to build an awareness of the context in which it is operating. This section defines these concepts, starting with the most basic: the service-oriented architecture.

In a Service-Oriented Architecture (SOA), each information source is considered to be a separate service, providing information to remote clients. Each service is developed and deployed by the organization that provides the information, eliminating the need for complex interface

agreements. Also, each service is remotely accessible, usually over the Internet. Services are generally not constrained to work with a single distributed system. This helps to keep the degree of coupling low, and allows the same service to be used in multiple contexts.

Web Services are a specialization of the more general Service-Oriented Architecture. The definition of Web Service has been hard to pin down, but the World Wide Web Consortium (W3C) has provided the following definition: "A Web service is a software application identified by a URI, whose interfaces and bindings are capable of being defined, described, and discovered as XML artifacts. A Web service supports direct interactions with other software agents using XML-based messages exchanged via Internet-based protocols." (see web site at: (http://www.w3.org/TR/wsa-reqs#IDAIO2IB)) Current implementations of web service standards specialize this farther: most web services use HTTP (Hyper-Text Transfer Protocol) to exchange messages defined in SOAP (Service Oriented Architecture Protocol). The SOAP standard defines an XML language and a set of rules for serializing and de-serializing objects and data, regardless of programming language, operating system, or hardware platform.

The description of web services is handled by WSDL (the Web Services Description Language), while discovery is provided by UDDI (Universal Description, Discovery, and Integration). UDDI defines an XML language for accessing a repository to register and locate web services according to the attributes of the service. The repository may be one of the public registries operated by Microsoft, IBM, or SAP. However, for most uses it is likely that participants will use a community registry accessible only to authorized partners. This paper discusses some of the limitations of WSDL and UDDI, below.

Why Discovery Matters in a Decision-Support System

As discussed earlier, in order to provide the broadest possible support for human decision-making, a decision-support system needs to provide all the information that is: (a) currently available; and, (b) currently relevant. This generally requires bringing together information from multiple sources.

Existing decision-support systems generally require that all potential sources of information be identified when the system is being designed and implemented. This requirement stems from the need to build concrete knowledge about each information source into the decision-support system. The information may include the location of the source, its access protocol, the data or information format, and the type of security used, among others.

The requirement that all sources of information be known prior to deployment prevents the decision-support system from taking advantage of information whose source is not identified (and may not even exist) while the system is being built. Most significantly, this requirement prevents the system from solving the 'transient need' problem. This problem arises when an unexpected situation requires a kind of information that is not present among the sources known to the system. For example, the information may be useful to a decision maker for a short period of time, but irrelevant after that point. Systems that can only use information sources that are known to the designers of the system tend to lack the flexibility required to deal with changes to the information environment and to the needs of the decision-maker.

A system built around web services solves the 'unknown information source' problem by adding the ability to discover services at run-time. If the service consumer can understand that a particular service provides information that the consumer can use, in a form that the consumer can process, then the consumer can use information from a previously-unknown source.

Some Limitations of Current Web Service Standards for Discovery

There are two important gaps in the ability of standard web services to discover and use unknown services. First, the standard discovery protocol only allows consumers to search for very specific types of services. In many cases, this amounts to a key-word search, which may result in missed opportunities and mistaken connections. Second, the standard method of defining service operations and parameters limits a would-be consumer to those services that implement operations using the names and classes that the consumer expects.

The difficulty is that the current web service standard for defining services and operations (Web Service Definition Language, or WSDL) does not include any information about the intent and meaning of an operation or its parameters. The best that a client can do is to search for operations based on the model that a given service implements, where the models are publicly defined either by the consumer, the providers, or by an industry standards body. The client will miss services that provide the same functionality but use different models.

For example: in the TEGRID demonstration project, we have created a definition of the operations and parameters for a 'publish-and-subscribe' service that allows clients to ask for information to be sent to them on topics of interest, and to send information regarding those topics. This service definition is then placed in the TEGRID service registry as a model of a service that the designers of publish-and-subscribe services may examine and implement. The SubscriptionManager service is one such service. In a real service-oriented system, there might be several services implementing the same model. Information about the SubscriptionManager is then entered into the service registry, including the fact that the SubscriptionManager implements the publish-and-subscribe model.

Participants in the TEGRID system that are interested in either sending or receiving information must implement the consumer side of the publish-and-subscribe model. That is, these participants must be able to discover those services that adhere to this specific model. In other words, they must be able to invoke operations using exact names and constructing parameters using class definitions that must be built-in to the client. Finally, the consumer must be able to receive values of a given class and map those values to objects within the consumer's own information model.

If another service were to provide the same functionality, but used different operation names and different classes for the input and output values, consumers searching the registry for publish-and-subscribe services would miss this equivalent service completely. The problem is that the registry does not describe the *purpose* (or *intent*) of a publish-and-subscribe service. As a result, consumers cannot find services based on a description of their own requirements, but only based on keywords and model definitions. The same problem also prevents consumers from locating new kinds of services (i.e., consumers are unable to describe their functional needs in a way that allows the registry to match those needs to specific service providers).

In short, using current web service standards, the 'discovery' process is limited in practical terms to discovering service locations. Adding entirely new types of services after deployment is only possible in limited circumstances. This in turn limits the ability of the client to draw information from all available relevant sources, and moves the effort of defining usable service types back from post-deployment to the development phase. Web service standards increase the flexibility of distributed systems, but they do not take us as far in that direction as we would like.

These limitations do not mean that web services as they are defined today provide no value. Many useful systems can be, will be, and are being built based on current standards. Service models are being defined by consortiums and standards bodies for specific vertical markets. These models will enable wide-spread interoperability. Service producers and consumers written for those models have the potential to provide unprecedented levels of system-to-system communication throughout an industry, resulting in significant increases in productivity.

Additionally, WSDL (the language used for defining service models) is designed to enable automated generation of client and server interfaces. The effort of incorporating a new service model defined in WSDL into a consumer is therefore generally very small, so that modifying a consumer to access a new type of information server can be done quickly and at little cost. Many (perhaps most) web service consumers will never have a need to dynamically locate sources of entirely unknown kinds of information. For these consumers, the limitations of the current methods of locating web services are simply irrelevant.

In order to provide full support for decision makers in areas where time is critical, on the other hand, these limitations do matter. The goal is to be able to give decision makers access to the information they need, when they need it, even when this involves types of information that were not anticipated by the designers of the system. At the CADRC, we are engaged in ongoing research exploring the use of semantic information to extend web services. We hope that such an approach will eventually allow systems to become progressively more flexible and responsive to the needs of their users. Some background on and explanation of the semantic approach is given in the next section.

Semantic Web Services

Several years ago, Tim Berners-Lee (the originator of the World Wide Web) began to discuss his vision of the future of the web. The web as it currently exists consists mainly of human-readable information. The markup in web pages is dedicated almost entirely to presentation instructions. This means that the only machine-processable content of most web pages is concerned with how the page should look.

Berners-Lee envisioned a web whose contents would include a new kind of markup that would enable software to reason about the *meaning* of the contents of a page. This would enable very intelligent automated processing of information on the web. Essentially, it would turn the contents of the web into an enormous knowledge base. Berners-Lee calls this vision of the future the *Semantic Web*. Semantic and logic languages are being developed to support this vision. The most influential of these languages is called RDF (Resource Definition Framework). RDF is deceptively simple, but has very deep underpinnings in the constructs and theory of formal logic.

Another language related to the Semantic Web is DAML-OIL (DARPA Agent Markup Language – Ontology Inference Layer). DAML-OIL builds on RDF, adding more complex

object-oriented concepts. DAML-OIL was submitted to the World Wide Web Consortium's Semantic Web working group, as the basis for the standardized OWL language, which is currently under development. It is likely that when OWL emerges from committee, it will bear a strong resemblance to DAML-OIL, although there may be significant differences.

Although web services and the semantic web are being defined by two very different communities, there is a growing realization of the potential synergy between the two. The basic idea here is that web services are a powerful means to deliver semantic information, while enhancing web service standards with semantic information will increase their flexibility and their effectiveness. The combination of Web Services with the Semantic Web is referred to (not surprisingly) as *Semantic Web Services*.

How Does Semantic Information Improve Service Discovery?

As discussed above, the current methods of discovering and invoking services and operations do not deal with the meaning of either the operations themselves or the classes of the parameters and return values (the input to and output from the operations). This limitation can be described succinctly as the *lack of semantic information*.

The premise of the research project currently in progress at the CADRC is as follows: If service definitions were to be expanded to include a formal description of the purpose of each operation, as well as an ontology that relates the classes of objects being passed in and out to other classes and concepts in the domain of knowledge, it would become possible for a would-be consumer to make a more intelligent determination of the suitability of a given operation to the consumer's own needs.

Semantic information concerning the service's domain of knowledge may also allow the prospective consumer to map operation parameters and return values to the consumer's own representation. This mapping is critical to the possibility of accessing services with unknown operation models. In other words, the consumer needs to be able to determine what information must be sent each operation, and what to do with the information received from the service. Without the ability to create mappings to and from its own information model at run-time, a consumer is again restricted to exactly those services that implement a service model that is built into the consumer during design and development.

In a previous pilot project, CADRC developers defined several ontologies using DAML (DARPA Agent Markup Language) and demonstrated the ability of a client program to automatically merge ontologies from multiple services under controlled conditions. This demonstration project also showed that users could extend the information model of a program at execution time, and that inference rules written for a specific ontology could also operate on instances of classes that the client had received from service provider, even though the classes were not known to the developers that wrote the rules. A future project will extend this investigation to include DAML-S (DAML Services), a vocabulary with semantics for defining the capabilities of services. We hope to learn whether the use of DAML-S to define the semantics of a service can enable a consumer to discover and utilize services without the need for each service to implement a specific interface.

The goal of adding semantic-level service descriptions is to enable consumers to locate services based on each service's purpose, rather than the names of the service's operations or the types of its parameters. This description of purpose is stated in a formal language that can be interpreted by the client. Automated reasoning can then determine the relationships between each operation and the definition of services needed by the client. Semantic description will allow service discovery to become more flexible, and will eventually lead to systems that can evolve as more services become available and the needs of the users of client program change over time.

The TEGRID Demonstration System

To demonstrate the use of web services and discovery in a decision-support system, the CADRC implemented a system within the context of emergency management of the rolling power outages experienced in many parts of California during the summer of 2001. This system is called TEGRID (i.e., 'Taming the Electric Grid').

Based on knowledge acquisition performed under the auspices of the National Institute for Urban Search and Rescue (NIUSR), and with the cooperation of the Los Angeles Sheriff's Department in the Fall of 2001, we identified several distinct entities that would be involved in planning and executing responses to power outage situations. Among these: the local sheriff stations (LSS); rapid response teams (RRT); the power supply organization (PSO); the traffic control organization (TCO); and the emergency operations bureau (EOB). The remainder of this section describes the responsibilities and actions of each of these major participants in the TEGRID demonstration system.

In the demonstration scenario, the Emergency Operations Bureau (EOB) is responsible for coordinating responses to the announcement of power outages. This coordination potentially includes a wide variety of decisions and communications. For the purposes of the scenario, we implemented only the assignment of Rapid Response Teams to provide support for local sheriff stations at priority locations.

Since the focus of this demonstration was on constructing the system through discovery and loosely-coupled communication through web services, we simulated the existence of several information sources:

1. A database at each local sheriff station that contains current officer assignments, equipment manifests and status, and priority infrastructure and intersections.

2. Lists of Rapid Response Teams (RRTs) and their primary and alternative assignments, maintained by the Emergency Operations Bureau (EOB).

3. Current power supply information, along with alerts about planned and current power outages, maintained and disseminated by a Power Supply Organization (PSO).

4. Traffic information, especially alternative route planning, supplied by the Traffic Control Organization (TCO).

5. Incident reports, fed into the system from 911 emergency lines and other sources.

Most of these information sources do exist, but are not (currently) available as web services. With the possible exception of the alternative route information, creating web services for these systems would be straightforward, given the cooperation of the agencies and organizations involved.

In addition, we implemented two services that are not part of the problem statement but are essential to the operation of the system. The first service is the Web Services Kiosk (WSK). Currently, this is an implementation of the Universal Description, Discovery, and Integration (UDDI) standard, but over time we expect this to evolve into an expanded service that will provide advanced semantic-based discovery services and will be the key to future semantic web services. The second infrastructure service is the Subscription Manager. This is our implementation of a web service that provides the ability for entities in the system to register their interests in information on specific topics, and to publish information that may be of interest to other participants in the system. The information is published as XML documents so that subscribers are not dependent on any static definition of the contents.

The TEGRID Demonstration Scenario: Initial Discovery Phase

The primary visible participants in the demonstration are the Emergency Operations Bureau (EOB) and the two Local Sheriff Stations, Lomita and East Los Angeles. Each of them starts with only one piece of information: the location of the Web Services Kiosk (WSK). The process of discovery is similar regardless of the order in which the three primary participants are started, but for the purposes of this discussion, we will assume that the EOB is started first.

The EOB queries the WSK for the location of a service that can provide publish-and-subscribe functionality. This functionality is defined, as described above, through the use a publicly accessible service model. The WSK finds the only match, which is the Subscription Manager, and returns the Subscription Manager's location (its URI) to the EOB. The EOB also searches the WSK for a service that can monitor power supply levels and send alerts when a power outage is about to begin. The WSK returns the URI of the Power Supply Organization service.

Using the Subscription Manager, the EOB subscribes to notifications about the creation of any Local Sheriff Station (LSS). The EOB also publishes a description of itself, in case there are other elements of the system that have subscribed to the creation of EOB entities. Additionally, the EOB subscribes to notifications of power outages.

Now we start one of the LSS clients – Lomita, for instance. Lomita goes through the same process of using its knowledge of the WSK to discover the Subscription Manager, and registers its interest in receiving messages regarding the creation of EOBs. Lomita also publishes information about itself. Since there is already a subscriber for notifications of LSS creation, the Subscription Manager passes along Lomita's information to the EOB. The EOB adds the Lomita information to its knowledge base, and replies with its own information. Now the EOB and the Lomita station know each other's web location, which means that they can communicate directly with each other, and the EOB has information about Lomita, particularly the RRTs assigned there, and the station resources. When the East Los Angeles station is started, the same process occurs, resulting in East Los Angeles and the EOB learning each other's web location, and the EOB learning all the information that East Los Angeles has provided.

As each participant has entered the system, several agents resident within each one have also subscribed to information on various topics. We will see the effects of these subscriptions in a later phase.

At this point, we have demonstrated the use of two different kinds of discovery. First, there is registry-based discovery, which allowed all the participants to locate the Subscription Manager, and the EOB to find the Power Supply service. Each participant has now created its own awareness of the context in which it is operating, and has exchanged information with other entities in the system. TEGRID has constructed itself in an ad hoc manner, based on available information sources, and has established a loosely-coupled communication system. This is the end of the discovery phase.

The TEGRID Demonstration Scenario: Operational Phase

The operational phase begins when the Power Supply Organization (PSO) determines that a rolling power blackout is imminent (i.e., is planned to begin in fifteen minutes). The PSO publishes that information to all subscribers, which in this case includes only the EOB.

On receiving the imminent power outage alert, the EOB immediately broadcasts the alert to all Local Sheriff Stations. The EOB then uses its Station Monitor Agent to determine which LSSs will experience the outage within their jurisdiction, using the information provided by each station at the time of discovery. These LSSs receive a second alert at a higher priority level. In this scenario, the only station directly affected is East Los Angeles. The EOB also alerts the RRTs assigned to assist the affected stations, so that these RRTs will begin to prepare for deployment. When an LSS receives a power outage alert, it assumes a state of readiness consistent with whether the outage is within its jurisdiction or not.

The second stage in the operational phase occurs when the power outage actually occurs. This is a repeat of the previous step, except that all elements move to a higher state of readiness. The third stage of the operational phase begins with a report of a traffic accident within East Los Angeles' jurisdiction, sent into the system by the Incident Report information source. The East Los Angeles Sheriff Station determines that it does not have sufficient resources to cover the traffic accident due to the power outage. East Los Angeles therefore requests additional resources from the EOB.

The EOB service receives the request and uses its own Scheduling Agent to assign an RRT and other equipment to the traffic accident. In addition, the EOB creates an Incident Agent to monitor further messages relating to the accident.

The fourth (and final) stage of the operational phase begins with the RRT finding that the route to the traffic accident is blocked by traffic due to the fact that many signals are not functioning because there is no power in the area. The RRT requests assistance from the EOB in finding an alternative route. The EOB Incident Agent queries the Web Service Kiosk for a service that implements an alternative traffic route model. The WSK responds with the address of the Traffic Control Service. The Incident Agent sends the request for assistance to the Traffic Control Service, which utilizes its own Routing Agent to determine an alternative route to the traffic accident. The Traffic Control Service sends this route as a reply to the EOB Incident Agent.

At this point, the Incident Agent displays the alternative route to the user of the EOB client system. This allows a human being to examine this route and accept it, alter it, or ask for another

route. Keeping a human in the loop is vital here and at many other places in the system, because the human being will almost always be aware of information that is not available through the system. With the user's acceptance, the alternative route is sent to the RRT, which also accepts it. This concludes the demonstration scenario.

Aspects of the TEGRID Demonstration

The TEGRID demonstration system shows the ability of a system to configure itself based on the available participants and their intents and interests. It also demonstrates the ability of loosely-coupled systems to exchange object-oriented information when the communications protocol is chosen for this purpose. We demonstrated further that participants in a distributed system can extend their awareness of the information available to them, and do not need to rely on pre-defined knowledge of information sources.

On the negative side, we also proved to ourselves the limitations of existing web service standards due to the lack of semantic information. Elements of TEGRID were not required to know exactly what servers they would be using, but we could not eliminate totally the necessity for each element to have knowledge of the interfaces provided by different kinds of services.

Future Directions

As indicated previously, future work will be focused in the area of adding semantic information to the descriptions of services and to the ontology of a service. We will be examining the suitability of the new DAML-Services language for this purpose, by defining representations of the information needs of each participant, and determining whether DAML-S can be used to locate services without an exact match.

We will also be exploring further the use of DAML as an ontology representation language, and creating tools and techniques that will eventually allow systems built around the concept of the semantic web to merge disparate ontologies into a single information model.

Other researchers working in similar areas can be found through articles listed below in the reference section.

References

(It should be noted that there are a large number of resources on these topics, most of which are on the World Wide Web. The following is a short list of starting points, and should not be taken as definitive or complete.)

Special Theme Issue: The Semantic Web, ERCIM News, No. 51, October 2002, http://www.ercim.org/publication/Ercim_News/enw51/

Berners-Lee, Tim, James Hendler and Ora Lassila, The Semantic Web; Scientific American, May 2001

Graham, Steve, et al., Building Web Services with Java: Making Sense of XML, SOAP, WSDL and UDDI; Sams Publishing, Indianapolis, IN, 2001

McIlraith, Sheila A., Tran Cao Son and Honglei Zeng, Semantic Web Services; IEEE Intelligent Systems, March/April 2001 (pp. 46-5)

Oellermann, William L., Architecting Web Services; Apress, Berkeley, CA, 2001

Pallos, Michael S., Service-Oriented Architecture: A Primer; eAI Journal, December 2001, http://www.eaijournal.com/PDF/SOAPallos.pdf

Appendix A: Semantic Web Services and Network-Centric Systems

A recently announced goal of the Defense Information Systems Agency (DISA) is a move to what has been dubbed 'net-centricity' or 'network-centric'. This goal is directly related to the enterprise architecture known as the 'Global Information Grid' (GIG). As John Osterholz, the director of architecture and interoperability in the Department of Defense's CIO office recently stated: "We believe ultimately that the key to managing data overload is making commanders responsible for pulling the data that they need into their decision space rather than having some galactic, on-line genius decide what they need". (Osterholz's statement was reported on the following web site: http://www.gcn.com/21_29/management/20098-1.html.)

This vision has a deep correlation with the goals of the project described in this paper. In order for each commander (or any other human or software agent) to pull the data (information) that suits his, her, or its current needs, most existing systems would require the user to identify an exact set of information sources. But in a distributed system of the size envisioned by the Department of Defense, it is unlikely that any individual would be able to be aware of all the potential information sources within this network-centric system.

Clearly, a form of automated discovery is necessary for the success of net-centricity. There may be a temptation to repeat past practices and create a (very large) number of standard service models, so that service providers can search registries for service providers implementing those models. If this occurs, commanders using specific client software programs will be restricted to accessing information providers whose models are built into the software. This will be problematic if the commander is in a fluid situation that might benefit from the input of unplanned types of information.

A far more flexible solution will include the addition of semantic descriptions of service capabilities to the discovery system. Client software will then be able to describe the information needs of the user in order to locate appropriate information sources. As the user's needs change, he or she will be able to direct the client software to seek out new services to provide new kinds of information. The result will be the right information delivered to the right person at the right time.

Web services in general also address two other problems that continue to obstruct progress in moving toward transformation of the Department of Defense's information technology systems: interoperability; and ,legacy systems. Web services help to solve both problems by putting a web-based interface on legacy systems. This interface is by definition interoperable with web service clients, which may include other systems that could use the information contained in the legacy system. Adding a web service interface to a legacy system is far more cost-effective than

re-engineering or replacing the system, and in the case of client/server systems, constructing a web service interface can generally be performed over a short period of time, contrasting with the major effort involved in developing replacement systems. Finally, in the case where multiple legacy systems are planned to be replaced by a single new system in the future, the web service interface can serve as a portal that allows clients to access information from the legacy system now, while forwarding requests to the new system as it comes on-line, without rewriting the web service client.

Appendix B: Organizations and Standards

Resource Description Framework (RDF)
http://www.w3.org/RDF/

DAML Organization
www.daml.org
The source for information on the DARPA Agent Markup Language

Semantic Web Working Group of the World Wide Web Consortium
http://www.w3.org/2001/sw/

Web Services Working Group of World Wide Web Consortium
http://www.w3.org/2002/ws/

UDDI
http://www.uddi.org/

WSDL
http://www.w3.org/TR/wsdl