Agents as Web Services

Michael N. Huhns • *University of South Carolina* • *huhns@sc.edu*

A recent minor encounter with my local medical system produced the following flurry of mail: I received bills from the doctor, the medical lab, the hospital, the specialist, and another bill from the hospital, then a refund from the doctor, a form to verify a procedure (which was disallowed, in spite of prior assurances from the doctor), and another bill as reimbursement for the procedure that was no longer covered by insurance. My insurance provider has strict rules about covered procedures, co-payments, and deductibles in various categories, but these rules are difficult to understand and not easily discovered by my local healthcare providers.

What does this have to do with Web services? Well, the organizations participating in my medical encounter might have implemented their capabilities as online Web services. By invoking each other's functionalities, the Web services could have determined the covered procedures, the insurance payments, and the correct amount to be paid to each healthcare provider. I would have received a single statement and could have then authorized automatic debits from my bank account, avoiding the blizzard of paperwork that I have just weathered.

Web Services

Given the potential this example illustrates, Web services are the hottest trend in information technology. Hardly a computer magazine drops through my mail slot today that doesn't feature them. And why not? Web services are XML-based, work through firewalls, are lightweight, and are supported by all software companies. They are a key component of Microsoft's .NET initiative and are deemed essential to the business directions charted by IBM, Sun, and SAP.

Web services are also central to the envisioned Semantic Web, which is what the World Wide Web is evolving into. But the Semantic Web is a friendly environment for software agents, which will add capabilities and functionality to the Web. How will agents and Web services relate?

The Semantic Web

The Web was designed for humans. It is based on a simple concept: information consists of pages of text and graphics that contain links. Each link leads to another page of information, with all of the pages intended for viewing by a person. The constructs used to describe and encode a page — the Hypertext Markup Language (html) — describe the page's appearance, but not its contents. By contrast, software agents don't care about the appearance, but only the contents.

Some agents, however, use the Web as it is now. Take the shopbot, an agent that visits online retailer catalogs and returns the prices asked for items users might want to buy. Shopbots operate by a form of *screen-scraping*, in which they download catalog pages and search first for the name of an item of interest and then for the nearest set of characters that has a dollar sign, which presumably is the item's price. The shopbots also might submit the same forms that a human would likely submit and then parse the returned pages that merchants expect that humans are viewing. The Semantic Web[1] will make the Web more accessible to agents by making use of semantic constructs, such as ontologies represented in well-established languages, so that agents can *understand* what is on a page.

Current Standards for Web Services

Web services are currently based on the triad of functionalities depicted in Figure 1 (next page). The architecture for Web services rests on principles and standards for connection, communication, description, and discovery:

- The eXtensible Modeling Language (XML) provides the common language service providers

 1089-7801/02/$17.00 ©2002 IEEE

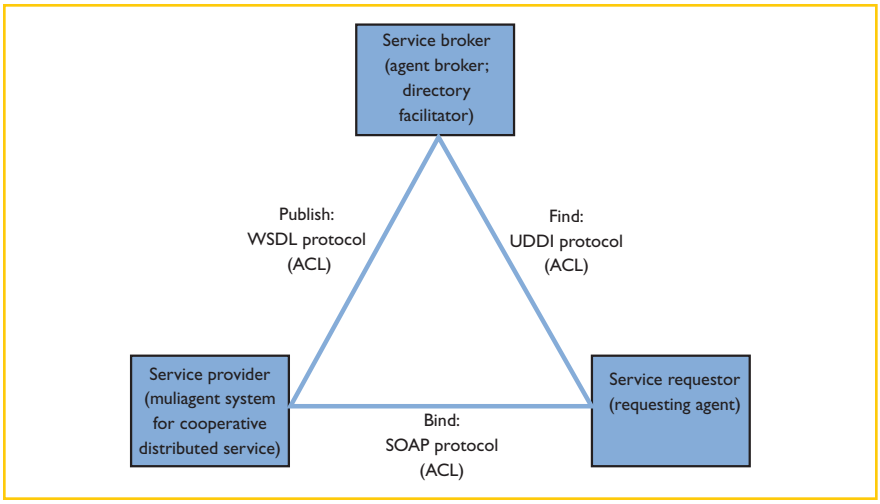

Figure 1. Web services rely on the functionalities of publish, find, and bind. The equivalent agent-based functionalities are shown in parentheses, and all interactions are via an agent-communication language (ACL).

and requestors need to connect and exchange information.

- The Simple Object Access Protocol (SOAP) provides the common protocol systems need to communicate with each other so that they can request services, such as to schedule appointments, order parts, and deliver information.[2]
- The Web Services Description Language (WSDL) describes the services in a machine-readable form, where the names of functions, their required parameters, and their results can be specified.
- Finally, Universal Description, Discovery, and Integration (UDDI) gives clients — users and businesses — a way to find needed services by specifying a registry or "yellow pages" of services.

Besides standards for XML, SOAP, WSDL, and UDDI, Web service developers and users need to agree broadly on the semantics of specific domains. The Resource Description Framework (RDF),[3,4] the DARPA Agent Modeling Language (DAML), and, more generally, ontologies[5] support such broad agreement.

Directory Services

A directory service lets components and participants locate each other, where the components and participants might be applications, agents, Web service providers, Web service requestors, people, objects, or procedures. The two general types of directories, determined by how entries are found in the directory, are name servers or *white pages*, which list entries by name, and *yellow pages*, which list entries by their characteristics and capabilities.

Implementing a basic directory is a simple database-like mechanism that lets participants insert descriptions of the services they offer and query for services offered by other participants. A more advanced directory might be more active than others, in that it might provide not only a search service, but also a brokering or facilitating service. For example, a participant might request a brokerage service to recruit one or more agents that can answer a query. The brokerage service would use knowledge about the requirements and capabilities of registered service providers to determine the appropriate providers to which to forward a query. It would then send the query to those providers, return their answers to the original requestor, and learn about the properties of the responses it passes on. For example, the brokerage service might determine that advertised results from provider X are incomplete and so seek out a substitute for provider X.

UDDI is itself a Web service that is based on XML and SOAP. It provides both white-pages and yellow-pages services, but not a brokering or facilitating service.

Agents Versus Web Services

Typical agent architectures have many of the same features as Web services. Agent architectures provide yellow- and white-pages directories, where agents advertise their distinct functionalities and other agents search to locate the agents so they can request those functionalities. However, agents extend Web services in several important ways:

- A Web service knows only about itself, but not about its users, clients, or customers. Agents are often self-aware at a metalevel, and through learning and model building gain awareness of other agents and their capabilities as interactions among the agents occur. Without such awareness, a Web service could not capitalize on new capabilities in its environment or customize its service to a client, such as by providing improved services to repeat customers.
- Web services, unlike agents, are not designed to use and reconcile ontologies. If a service's client and provider happen to use different ontologies, the result of invoking the Web service would be incomprehensible to the client.
- Agents are inherently communicative, whereas Web services are passive until invoked. Agents can provide alerts and updates when new information becomes available. Current standards and protocols make no provision for even subscribing to a service to receive periodic updates.
- A Web service, as currently defined and used, is not autonomous. Autonomy is a characteristic of agents, and it is also a characteristic of many envisioned Internet-based applications. Among agents, autonomy generally refers to social autonomy, where an agent is aware of its colleagues and is sociable,

but nevertheless exercises its independence in certain circumstances. Autonomy is in natural tension with coordination or with the higher-level notion of a commitment. To be coordinated with other agents or to keep its commitments, an agent must relinquish some of its autonomy. However, an agent that is sociable and responsible can still be autonomous. It would attempt to coordinate with others where appropriate and to keep its commitments as much as possible, but it would exercise its autonomy in entering into those commitments in the first place.

- Agents are cooperative, and by forming teams and coalitions can provide higher-level and more comprehensive services. Current standards for Web services do not provide for composing functionalities.

Composing Cooperative Services

Imagine that a merchant would like to empower a customer to track the shipping of a sold item. Currently, the best the merchant can do is to point the customer to the shipper's Web site, which the customer can check for delivery status. If the merchant could compose its own production notification system with the shipper's Web services, the result would be a customized delivery notification service by which the customer — or the customer's agents — could find a purchase's status in real time.

As Web uses (and thus Web interactions) become more complex, having one server provide a total solution will become increasingly difficult, as will having one client integrate solutions from many servers. Web services currently involve a single client accessing a single server, but soon applications will demand federated servers with multiple clients sharing results. Cooperative peer-to-peer solutions will have to be managed, an area in which agents have excelled. In doing so, agents can balance cooperation with their owner's interests.

Conclusion

Web services are extremely flexible. Most advantageously, a developer of Web services need not know who or what will use the services being provided. They can be used to tie together a single company's internal information systems or the interoperational systems of virtual enterprises. But how Web services tie the systems together — likely by using agents — is a current topic of research. Someday, the agent-based Web services might even make sense out of the bills and notices I receive from the healthcare industry.

Acknowledgements

The National Science Foundation supported this work under grant number IIS-0083362.

References

1. T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American*, vol. 284, no. 5, May 2001, pp. 34–43.
2. D. Box et al., "Simple Object Access Protocol (SOAP) 1.1," 2000, www.w3.org/TR/SOAP.
3. S. Decker et al., "The Semantic Web: The Roles of XML and RDF," *IEEE Internet Computing*, vol. 4, no. 5, Sept.-Oct. 2000, pp. 63–74.
4. S. Decker, P. Mitra, and S. Melnik, "Framework for the Semantic Web: An RDF Tutorial," *IEEE Internet Computing*, vol. 4, no. 6, Nov.-Dec. 2000, pp. 68–73.
5. J. Heflin and J. A. Hendler, "Dynamic Ontologies on the Web," *Proc. Am. Assoc. for Artificial Intelligence Conf.* (AAAI), AAAI Press, Menlo Park, Calif., 2000, pp. 443–449.

Michael N. Huhns is a professor of computer science and engineering at the University of South Carolina, where he also directs the Center for Information Technology.

IEEE Internet Computing

Editorial: *IEEE Internet Computing* targets the technical and scientific Internet user communities as well as designers and developers of Internet-based applications and enabling technologies. Instructions to authors are at http://computer.org/internet/edguide.htm. Articles are peer reviewed for technical merit and copy edited for clarity, style, and space. Unless otherwise stated, bylined articles and departments, as well as product and service descriptions, reflect the author's or firm's opinion; inclusion in this publication does not necessarily constitute endorsement by the IEEE or the IEEE Computer Society.

Circulation: *IEEE Internet Computing* (ISSN 1089-7801) is published bimonthly by the IEEE Computer Society. IEEE headquarters: 3 Park Avenue, 17th Floor, New York, NY 10016-5997. IEEE Computer Society headquarters: 1730 Massachusetts Ave., Washington, DC 20036-1903. IEEE Computer Society Publications Office: 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720; (714) 821-8380; fax (714) 821-4010. Subscription rates: IEEE Computer Society members get the lowest rates and choice of media option — US$37/30/48 for print/electronic/combination. For information on other prices or to order, go to http://computer.org/subscribe. Back issues: $10 for members, $20 for nonmembers. Also available on microfiche.

Postmaster: Send undelivered copies and address changes to *IEEE Internet Computing*, IEEE Service Center, 445 Hoes Ln., Piscataway, NJ 08855-1331. Periodicals postage paid at New York, N.Y., and at additional mailing offices. Canadian GST #125634188. Canada Post International Publications Mail Product (Canadian Distribution) Sales Agreement #1008870. Printed in USA.