My Agent Wants to Talk to Your Service: Personalizing Web Services through Agents

Harumi Kuno, Akhil Sahai
Software Technology Laboratory
HP Laboratories Palo Alto
HPL-2002-114
April 23rd , 2002*

E-mail: harumi_kuno@hp.com, akhil_sahai@hp.com

web services, agents, personalization

Web services are tailored for the support of dynamic business-to-business interactions. As web service technology evolves, we anticipate that the question of how to use customer information to personalize the relationship between web services and their customers will move beyond the current focus on a service's interactions with the human consumer, and on to the challenge of how a customer can assign a delegate that will programmatically interact with web services according to context when acting on the behalf of a customer. We recognize that agent technology could be used to meet these goals. We identify a number of outstanding issues that web service and agent platforms must evolve to address in order for the two paradigms to work together. In addition, we propose a personalization component that can be integrated with existing web service infrastructures.

* Internal Accession Date Only
Approved for External Publication

My Agent Wants to Talk to Your Service: Personalizing Web Services through Agents

Harumi Kuno
Hewlett-Packard Laboratories
1501 Page Mill Road, MS 1135
Palo Alto, CA 94304
(1) 650-857-3228
harumi_kuno@hp.com

Akhil Sahai
Hewlett-Packard Laboratories
1501 Page Mill Road, MS 1135
Palo Alto, CA 94304
(1) 650-857-2022
akhil_sahai@hp.com

ABSTRACT
Web services are tailored for the support of dynamic business-to-business interactions. As web service technology evolves, we anticipate that the question of how to use customer information to personalize the relationship between web services and their customers will move beyond the current focus on a service's interactions with the human consumer, and on to the challenge of how a customer can assign a delegate that will programmatically interact with web services according to context when acting on the behalf of a customer. We recognize that agent technology could be used to meet these goals. We identify a number of outstanding issues that web service and agent platforms must evolve to address in order for the two paradigms to work together. In addition, we propose a personalization component that can be integrated with existing web service infrastructures.

Keywords
Web services, agents, personalization

1. INTRODUCTION
Web services are distributed services that are accessible via the Internet through Uniform Resource Locators (URLs). Naturally, web services must address traditional requirements, such as scalability and reliability. In addition, specific characteristics distinguish web services from more traditional e-commerce applications:

1. *Describable*: Web service interfaces are described in terms of operations they support and messages they exchange.
2. *Discoverable*: One of the foremost requirements for a web service to be useful in a commercial scenario is that it be discoverable by consumers (humans or other web services). UDDI operator sites provide such capability for web services for registration and discovery.
3. *Message-based communication*: Web services communicate by exchanging documents through SOAP messaging.

The reason that web services have these characteristics is that they are intended to engage in dynamic business-to-business (B2B) interactions with services deployed on behalf of other enterprises. Some consider any application that offers a web-interface (i.e., that publishes data in the HTML format using the standard HTTP protocol) to be a "web service," but we believe that this interpretation, which ignores description and discovery functionality, misses the innovation of the web service vision.

As web service technology evolves, we anticipate that they will become increasingly sophisticated, and that the challenges web service community will face will also evolve to meet their new capabilities. One of the most important of these challenges is the question of what it means to personalize web services.

In this paper, we address the problem of web service personalization. We begin in Section 2 by providing a brief overview of current web service platforms. We then discuss in Section 3 how the goals of personalization apply to the web services paradigm. In particular, we identify a number of new issues that web service platforms must evolve to address. In Section 4 we propose an architecture that addresses these issues, which we compare to related work in Section 5. Finally, in Section 6, we summarize our conclusions

2. CURRENT WEB SERVICE PLATFORMS
The two primary currently emerging web services infrastructure standards are J2EE and .Net. J2EE was designed to provide for 3-tier service architectures, as opposed to 2-tier client-server architectures. HP's Application Server, BEA's WebLogic are examples of J2EE-compliant application servers that combine web server farms with EJB engines. The 3-tier architecture extends the client-server paradigm with a presentation layer, which could be static as is the case with a simple browser or could be more dynamic depending on the usage of JSPs, servlets, the Application server logic layer or business logic layer as it is usually termed is implemented in the form of Enterprise Java Beans (EJB)s. The business logic layer accesses data from databases through entity beans that in turn use JDBC for accessing the databases. They can also access legacy software through Java Messaging Service (JMS). The commercial implementations of J2EE provide a web service layer on top of the J2EE infrastructure. This layer enables exposing services through WSDL [3] interfaces, registering these services at UDDI [4] operator sites and communicating with clients and other web services through SOAP messaging.

The .Net my Services initiative on the other hand personalizes web services by smoothing users' interactions with other web services. .NET provides a hosted environment for users to create, register, discover and use other web services. In order to personalize user interaction with other web services, a set of

customization services will be run by the hosted environment. These customization services will register user devices (.Net Devices), maintain the payment information in its wallet (.Net Wallet), maintain repository of favorite web sites (.Net Favorite Web sites), maintain mails in its Inbox (.Net Inbox,), maintain user documents in (.Net Documents) etc. The interactions between web services and creation of WSDL interfaces for web services will be facilitated by easy to use tools like .Net Visual Studio.

3. PERSONALIZATION AND WEB SERVICES

Personalization describes the problem of how to use customer information to optimize a business's relationship with its customers. Traditionally, personalization technology has focused on a service's interactions with the customer. According to the Personalization Consortium [16], the goal of this technology has been to use information about a customer so as to better serve the customer by anticipating needs, make the interaction efficient and satisfying for both parties, and build a relationship that encourages the customer to return for subsequent purchases.

The web service platforms described in the previous section are beginning to offer technology that supports traditional forms of personalization. For example, Microsoft's .NET Passport authentication service enables users to enter and store profile information (e.g., e-mail address and password, mailing address, etc.) so that it does not have to be re-entered as they move from web site to web site. The .Net wallet service will store credit card information so that users do not need to re-enter it either. As an alternative to Microsoft, the Liberty Alliance Project [14], is an organization to create an open, federated, single sign-on identity solution for the digital economy via any device connected to the Internet. Similarly, as Mike Clark [7] notes, a number of value added service suppliers that provide rudimentary rating and enriched service search services are beginning to emerge. Sites such as www.salcentral.com, www.bizrate.com offers service reviews and ratings. However, although these and other web services are emerging, they are intended for human user, and do not provide explicit support for automation and delegation.

Web services, however, offer the potential for a customer to assign a programmatic delegate to act on his or her behalf in interactions with services. Thus, the web service environment extends the issues raised by personalization to include the question of how a delegate can make decisions on behalf of the customer when interacting with a service, as opposed to how to adapt the service so that it can better interact with the customer.

In this paper, we do not address how a customer might store their personal profile information for use by an agent, or how a delegate should authenticate itself as representing a user. Instead, we focus on how a delegate could interact programmatically with web services.

4. ARCHITECTURE FOR PERSONALIZED WEB SERVICES

We propose an architecture (Figure 1) that leverages agent technology to personalize user interaction with web services. This aspect of personalization is lacking in current web service architectures. Our personalization component can be either integrated with hosted environments like .Net my services or with existing portals/integrators so as to provide user personalization.

In particular, we use agents to implement delegates. These agents proxy for the users, achieve goals of the user through goal-directed behavior (with or without the presence of the user) and interact with other constituents of the personalization component. For example, context managers track the user as its temporal and spatial location changes. Similarly, user profiles maintain static and dynamic information about the user. Static information is pre-registered while dynamic information is learnt by studying user behavioral patterns. As the delegates interact with web services over time their experiences are recorded in the rating services. These rating services can be used by delegates of other users or can also be passed to third party rating services.

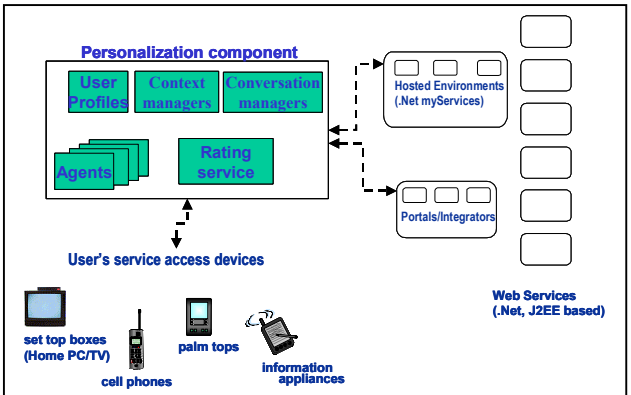

Figure 1 User devices interact with web services via the personalization component.

4.1 AGENTS AS DELEGATES

Multiple definitions of the term "Agent" have been proposed throughout the years, but fundamentally, agents must support the following four properties: autonomy, reactivity, proactivity, sociability [24]. We believe that these characteristics make agents a particularly appropriate technology for implementing delegation. Furthermore, since agents dynamically communicate via message exchanges that conform to specified protocols/patterns, agent-based conversations are naturally compatible with web service interactions.

However, we must address a number of issues in order for agents to interact effectively with web services. Web-services are much more loosely coupled than traditional distributed applications. Web-services are deployed on behalf of diverse enterprises, and the programmers who implement them are unlikely to collaborate with each other during development. Therefore, web services support very flexible, dynamic bindings.

In order for the agents and services to interact dynamically, they must be able to do three fundamental things. First, the agent must be able to discover services that are appropriate given a customer's preferences and requirements. The delegate should also be able to adapt its behavior based on the dynamic characteristics of the web services in a user's current physical and temporal context. Second, services must describe their abstract interfaces and protocol bindings so that agents can invoke them. Third, an agent should be able to carry out complex interactions (conversations) with multiple services while acting on behalf of a given user (for example, the agent should be able to login and then purchase an item from the service). This includes the

monitoring of ongoing interactions so that the delegate can take proactive alternatives should exceptions occur.

4.2 Advertisement / Discovery

Web service repositories, such as UDDI, publish the information needed to discover and interact with web services. This includes "white page" contact information for the businesses that front services, "yellow page" categorizations of the taxonomies and vocabularies used to describe services, and "green page" specifications of the interfaces and protocols used to access and interact with services.

Although UDDI features a SOAP message-based interface for programmatic access and publishes schemas describing its data structures, UDDI is optimized for interaction with humans (e.g., service developers), not agents. Thus, a number of issues must be addressed before agents can use web service repositories to discover and programmatically interact with services without human intervention:

First, in order for agents to select services in a context-sensitive manner, they must be able to discover services based on semantic description. However, in UDDI discovery is primarily by service name, not by service attributes. UDDI tModels can represent the equivalent of vocabularies, but it is cumbersome to perform lookups by tModel reference, and indeed the actual service descriptions in UDDI are unstructured and intended for human consumption. Second, UDDI repositories are fairly static, and intended for stable content. Agents, on the other hand, require dynamic information such as would be provided by a rating service. Finally, UDDI repositories does not support service context, such as location-dependent or context-sensitive information.

A number of research groups use ontologies to express semantic descriptions of web services. The DAML-based Web Service Ontology project (DAML-S) provides a core set of markup language constructs for describing the properties and capabilities of their Web services in unambiguous, computer-interpretable form. The W3C Web Ontology Working Group has published an initial draft of requirements for the Ontology Web Language (OWL) 1.0 specification. However, these efforts need to be integrated with service discovery mechanisms before agents can use them to discover web services. In addition, existing efforts do not address how to create and maintain descriptions that include dynamic attributes such as current load averages or ongoing performance characteristics.

Current web service rating services exist in quite an ad hoc manner. Certain web based businesses like Amazon and e-bay provide mechanisms for users to rate commodities (like books) and vendors (sellers) respectively. These mechanisms capture the user experience while undertaking business with them. BizRate intends to perform similar functions. Most of these are however ad hoc mechanisms. As agents representing users discover and undertake business with other web services, mechanisms for rating would be invaluable for making these web services "customer-friendly".

4.3 Context

In order for a delegate to represent a human user, the delegate must be able to characterize the context of the interaction. The user context can be current temporal and spatial location, terminal device characteristics, user preferences, users nearby, resources nearby, lighting, noise level, network connectivity, communication cost, communication bandwidth, and social situation. Some of this contextual information can be sensed through sensors like GPS, and network bandwidth detection, while others can be inferred from knowledge bases. Each of these contextual parameters are fairly complex. For example, terminal device characteristics could involve audio capability, video capability, image decoder, text capability, local storage capacity, screen size, network capability, content markup language capability. Temporal and spatial location pinpoints a user location in time and in the physical world, so that context specific services can be provided to the user.

Chen et al observe [4] that human contextual awareness is based upon three factors: the sensing of contextual information, the acquisition and sharing of contextual information, and reasoning about contextual knowledge. Much work in context-aware applications addressing all three of these areas has been done in both the agent and mobile computing communities [4, 10].

4.4 Invocation

Emerging standards such as Web Services Definition Language (WSDL) provide general-purpose XML-based languages for describing the interfaces and protocol bindings of web service functional endpoints. WSDL enables the description of web services irrespective of the message formats and network protocols used. For example, in WSDL a service is implemented through a set of endpoints. An endpoint is in turn a set of operations. Each operation is defined in terms of the messages that can be received and sent out.

The Java Agent Services (JAS) project is tasked with defining an industry standard specification and API for the development of network agent and service architectures [9]. The JAS will define Java classes describing the various components of message elements and defining agent names and descriptions, and Java interfaces corresponding to agent services for messaging, directories, and naming. Although JAS is designed to be compatible with standard transports, such as HTTP and SOAP, and it seems quite reasonable that an ACL message could exist as a payload of a SOAP message, JAS does not address how an ACL message could be translated into the format expected by a given web service. Similarly, HP BlueJade integrates agents in the Bluestone Application Server environment, but does not address how agents can leverage web-service standards (e.g., SOAP, UDDI, WSDL) or how services and agents can interact [3].

4.5 Orchestration

Orchestration languages such as WSCL[1], WSFL[23], ebXML[22], and RosettaNet's PIPs [21] provide XML schemas for defining legal sequences of message exchanges (interactions) web services support. Orchestration languages and interface definition languages are highly complimentary -- the latter specifies how to send messages to a service and former specifies the order in which such messages can be sent. The advantage of keeping the two distinct is that doing so allows us to decouple conversational interfaces from service-specific interfaces.

For example, the ebXML associates parties that engage in business with Collaboration Protocol Profiles (CPP). Once a party discovers another party's CPP they negotiate to form a

Collaboration Protocol Agreement (CPA). The intent of the CPA is not to expose business process internals of parties but to expose the visible process that involves interactions between parties. The CPA and the process specification document it references define a *conversation* between parties. This conversation involves multiple *Business Transactions*. A Business Transaction may involve exchange of messages as request and replies. The CPA may refer to multiple process specification documents.

Current web service platforms require services to participate in homogeneous marketplaces, in which participants code to matching conversation protocols; should a protocol change, all participants that support the protocol must be updated and recompiled. In addition, existing systems also couple the message exchanges with the internal state of a service/agent.

Several existing agent systems allow agents to communicate following conversational protocols (or patterns). However, to the best of our knowledge, all of these are tightly coupled to specific agent systems, and require all participating entities to be built upon a common agent platform. For example, the Knowledgeable Agent-oriented System (KaoS)[2] is an open distributed architecture for software agents, but requires agent developers to hard-wire conversation policies into agents in advance. Walker and Wooldridge [20] address the issue of how a group of autonomous agents can reach a global agreement on conversation policy; but require the agents themselves to implement strategies and control. Chen, et al. [5] provide a framework in which agents can dynamically load conversation policies from one-another, but their solution is homogeneous and requires that agents be built upon a common infrastructure.

Our personalization component includes conversation controllers, as proposed in [12, 13]. These Conversation Controllers allow us to require only that a participating service produce two XML-based documents -- 1) a specification of the conversational flows it supports and 2) a specification of the service's functionality (describing how the service can be invoked). This allows us to provide an extremely lightweight solution relieving service developers from the burden of implementing conversation-handling logic and make possible the automated coordination of complex conversations between agents and services that do not support compatible message document types.

5. CONCLUSIONS

We examine here the question of how agent technology can be used to personalize web services. In particular, we address the challenge of how a customer can assign a delegate that will programmatically interact with web services according to context when acting on the behalf of a customer. We have identified a number of outstanding issues that web service and agent platforms must evolve to address in order for the two paradigms to work together, and propose a personalization component that can be integrated with existing web service infrastructures.

6. REFERENCES

[1] A. Banerji et al. Web Services Conversation Language (WSCL) 1.0, W3C Note, Mar. 2002.

[2] J. M. Bradshaw. KAoS: An Open Agent Architecture Supporting Reuse, Interoperability, and Extensibility. Knowledge Acquisition for Knowledge-Based Systems Workshop, 1996.

[3] B. Burg. Agents in the world of active web-services. Second Kyoto Meeting on Digital Cities, 2001.

[4] H. Chen, S. Tolia, C. Sayers, T. Finin, and A. Joshi. Creating context-aware software agents. 1st Workshop on Radical Agent Concepts, Sept. 2001.

[5] Q. Chen, U. Dayal, M. Hsu, and M. Griss. Dynamic Agents, Workflow and XML for E-Commerce Automation. International Conference on E-Commerce and Web-Technology, 2000.

[6]E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Services Description Language (WSDL) 1.1, Mar. 2001.

[7] M. Clark. The Birth of the UDDI Value Added Service Supplier. Web Services Architect, Dec. 2001.

[9] Java Agent Services Project. http://www.java-agent.org/. N. Jennings and M. Wooldridge. Software agents. IEEE Review, pages 17–20, Jan. 1994.

[10] T. Kantor. Adaptive Personal Mobile Communication – Service Architecture and Protocols. PhD thesis, Stockholm University/KTH, Dec. 2001.

[11] W. Kim, S. Graupner, A. Sahai et al. Web E-speak -A Document Interchange Engine for Web based E-Services. IEEE Multi-Media, Jan. 2002.

[12] H. Kuno and M. Lemon. A Lightweight Dynamic Conversation Controller for E-Services. WECWIS 2001.

[13] H. Kuno, M. Lemon, and A. Karp. Transformational interactions for p2p e-commerce. HICSS-35, Jan. 2002.

[14] Liberty Alliance Project. http://www.projectliberty.org.

[15] S. A. McIlraith, T. C. Son, and H. Zeng. Mobilizing the semantic web with DAML-enabled web services. Semantic Web Workshop, 2001.

[16] Personalization Consortium. []http://www.personalization.org/personalization.html.

[17] A. Sahai and V. Machiraju. Enabling of Ubiquitous E-Services Vision on the Internet. E-Services Journal, 2001.

[18] A. Sahai and C. Morin. Mobile agents for enabling mobile user aware applications. Second ACM International Conference Autonomous Agents (Agents 98), May 1998.

[19] Universal Description, Discovery and Integration (UDDI) Project. http://uddi.org.

[20] A. Walker and M. Wooldridge. Understanding the emergence of conventions in multi-agent systems. First International Conference on Multi-Agent Systems, 1995.

[21] RosettaNet. http://rosettanet.org

[22] EbXML http://www.ebxml.org

[23] WSFL. http://www.ibm.com/software/solutions/webservices/

[24] N. Jennings, M. Wooldridge. Software Agents. IEEE Review. January, 1994. Pg 17-20.