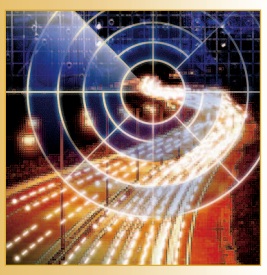

Editors: Michael N. Huhns • *huhns@sc.edu*
Munindar P. Singh • *singh@ncsu.edu*

A Semantic Web Services Architecture

The Semantic Web Services Initiative Architecture (SWSA) committee has created a set of architectural and protocol abstractions that serve as a foundation for Semantic Web service technologies. This article summarizes the committee's findings, emphasizing its review of requirements gathered from several different environments. The authors also identify the scope and potential requirements for a Semantic Web services architecture.

Mark Burstein
BBN Technologies

Christoph Bussler and Michal Zaremba
Digital Enterprise Research Institute

Tim Finin
University of Maryland, Baltimore County

Michael N. Huhns
University of South Carolina

Massimo Paolucci
DoCoMo Communications Laboratories Europe GmbH

Amit P. Sheth
University of Georgia

Stuart Williams
Hewlett-Packard Laboratories

Formed in February 2003, the Semantic Web Services Initiative Architecture (SWSA) committee's mission is to develop the necessary abstractions for an architecture that supports Semantic Web services. The resultant framework builds on the W3C Web Services Architecture working group report (and is motivated in part by Tim Berners-Lee's vision for the Semantic Web[1]). Other groups developing Semantic Web services frameworks contributed to the discussions, including the Web Ontology Language for Services (OWL-S) consortium, the Web Service Modeling Ontology (WSMO; www.wsmo.org) group at the Digital Enterprise Research Institute (DERI), and the Managing End-to-End Operations-Semantics (METEOR-S; http://lsdis.cs.uga.edu/projects/meteor-s/) group at the University of Georgia.[2,3]

In this article, we describe the protocols exchanged between the interacting entities or agents that interpret and reason with semantic descriptions in the deployment of Semantic Web services. We focus specifically on those capabilities that extend the potential range of Web services; we also discuss security, reliability, and a flexible means of recovery from the problems that can occur in open and evolving environments. The SWSA architectural framework attempts to address five classes of Semantic Web agent requirements — dynamic service discovery, service engagement, service process enactment and management, community support services, and quality of service (QoS) — which we cover in detail here as well.

Multiple Distributed Environments

Systems developed via Web service technologies are often limited by their need to agree in advance on the syntax and semantics of various communications. Whereas the World Wide Web is successful in part because it makes it easy to interact with and gather information from

 Published by the IEEE Computer Society 1089-7801/05/$20.00

sites that are discovered dynamically, traditional Web services are designed to function more like distributed object-oriented computing systems. In contrast, semantically transparent services will make it possible for clients to successfully use services that are dynamically discovered without prior negotiations between client and service developers.

Such goals are important for commercial Web service environments, including business-to-business and business-to-consumer applications, grid computing, ubiquitous computing, and information management. For commercial Web services, it will be increasingly important for service providers to be able to adapt their interfaces to support new products and service options without interrupting or requiring changes to the software that clients use to access those services. Likewise, clients that wish to comparison shop or use alternate services when the ones they traditionally use are unavailable will need to adapt flexibly to the differences between the interfaces those alternative services present. Similar issues arise with grid computing services, in which computational resources are often oversubscribed and different sites that can perform the same functions often have different interaction requirements.

These environments often have two barriers to interoperability: incompatible information models and mismatches in different service providers' interaction protocols. Dynamically accessible semantic descriptions of service capabilities and utilization protocols, based on shared semantic models published on the Semantic Web, are seen as a way to overcome these barriers, but they will require additional infrastructure so that individual software agents can directly interpret published service descriptions (which sometimes use unfamiliar ontologies). Given the open-ended nature of Web-oriented distributed environments, this architecture must also handle semantically interpretable security authorizations and ensure the privacy of transmitted information.

Some aspects of our proposed architecture will be more central to particular applications than others, and some will take longer to be widely adopted. Our near-term emphasis is on semantics for user-directed service interactions: users specify what they want from a service, rather than the details of how to ask for it. For this family of uses, a service client will need to avoid hard-coded knowledge of the syntax for interacting with particular providers, instead using information from published semantic service descriptions to mediate interactions with classes of functionally similar providers, even when their detailed interfaces differ. These software clients must translate user requests into suitable forms for each potential provider, and then use the interaction methods specified by those providers by reasoning from the methods' published semantic descriptions.

As the technology matures, we anticipate that methods now being explored will support more widespread use of mechanisms for automated service discovery and matchmaking. A key element of this phase will be the development of shared, extensible, community-wide ontologies for describing capabilities, services, and goods, and — equally critically — for making these ontologies publicly accessible for use by Web applications. Open-source ontologies for different kinds of services and products will enable broad-based, automated, service discovery in the same way search engines now make it easy to discover new Web sites.

Underlying Assumptions

Our architectural framework builds on two emerging technological concepts: Web services and the Semantic Web. Web service providers can publish descriptions of service interfaces on the Web using the XML-based Web Services Description Language (WSDL). These descriptions include information about the message forms used to invoke the services, which can be serialized using HTTP and SOAP protocols, among others. WSDL does not, however, have a systematic way to associate meanings with the messages and message arguments that appear in those descriptions. The Semantic Web vision takes Web-publishing of descriptions to the next level by introducing semantic description languages built on XML, which lets people publish and share ontologies — set of conceptual terms labeled by URLs — that can be used in describing other published materials. Semantic Web services are Web services in which semantic Web ontologies ascribe meanings to published service descriptions so that software systems representing prospective service clients can interpret and invoke them.

This is a good time to talk about how clients and services can act as software agents with goals. In the discussions that follow, we assume the following general capabilities of agents in Semantic Web service environments (in this context, "agents" include requesters [clients], service providers, and middle agents).

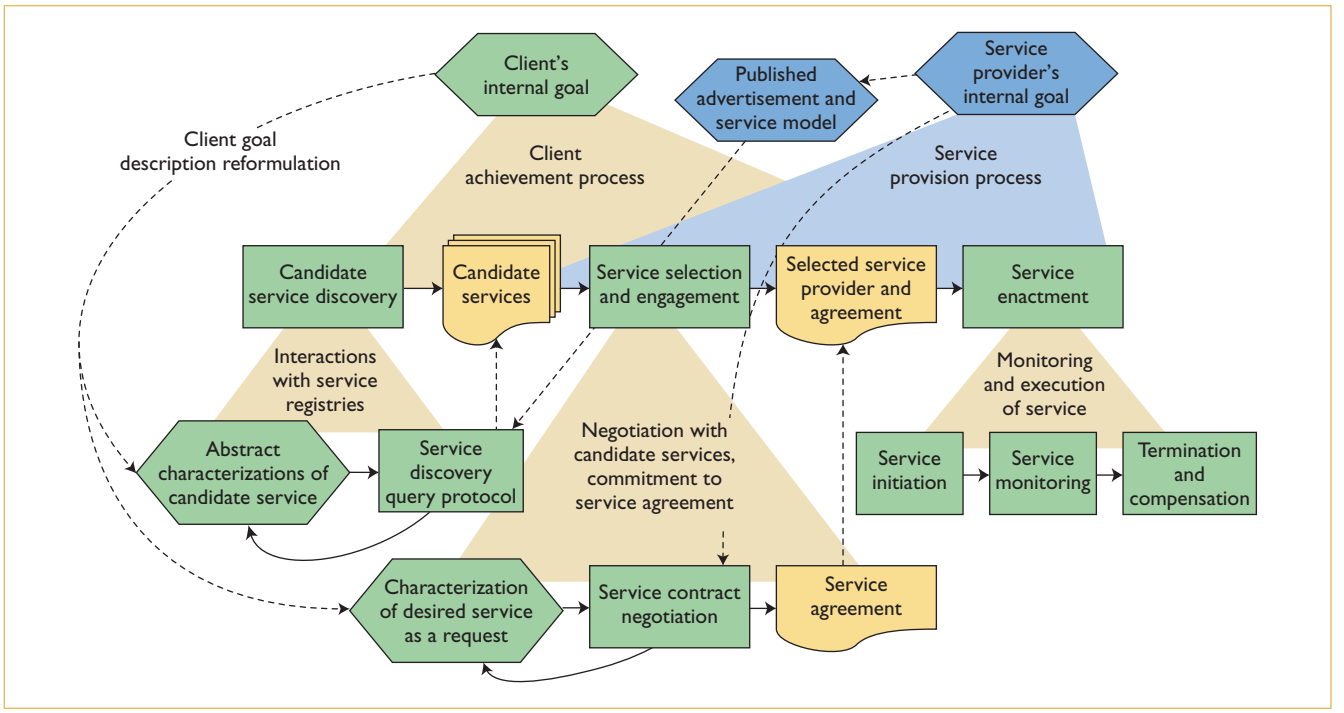

Figure 1. Service interaction process. Client (green) and service provider (blue) goal descriptions (hexagons) drive the three main phases of interaction (discovery, engagement, and enactment). At the lower level, these goals are communicated during message exchanges utilizing protocols (green boxes) that follow general, phase-specific patterns.

- All agents can access and interpret Web-published ontologies, and can communicate using messages whose content is represented, or can be interpreted, in terms of published ontologies.
- Service providers publish semantic descriptions of their service capabilities and interaction protocols, which prospective consumers can then interpret when selecting appropriate services and when formulating their interactions with those services.
- Requestor agents wishing to delegate internal objectives to external agents can reformulate those objectives as well-formed requests to service providers by using those providers' semantically described service interfaces as guides.

For dynamic Web services to function, clients must be able to interpret the published semantic descriptions of unfamiliar services to help them decide which services to use and how to interact with them. As a result of this style of interaction, clients will be able to adjust smoothly as service interfaces evolve. Such interaction also lets clients discover and substitute new services for ones that are no longer available, even if those new services use different protocols or message types.

Phases of Semantic Web Service Interaction

The overall process of discovering and interacting with a Semantic Web service will, in general, include three phases:

- *Candidate service discovery* is the distributed search for available services that can (potentially) accomplish some set of a client's internal goals or objectives. One architectural approach to this phase is interaction with a semantic matchmaker, a registry agent, or a peer agent serving either function.
- *Service engagement* includes the process of interpreting candidate Web service enactment constraints (partly or fully described in each service's published self-description), and then negotiating with prospective services until reaching an agreement. This phase concludes when both service and client agree to the service-provision terms in an explicit or implicit service contract. Negotiations can include service price, product attributes, and the quality and timeliness of service, security and privacy, and so on.
- *Service enactment* is the process that completes the mutually agreed upon objectives of client

and service by following the service's published protocols. If the contract's primary objectives aren't accomplished, then the client and service can use a compensation protocol to restore financial equity or a stable operating state. If the underlying message transport mechanisms allow for asynchronous interactions, then clients can use protocols to monitor the service process's status during execution.

Figure 1 gives an overview of the workflow within and relationships among these three phases. A client (a user plus a software agent) starts with an internal goal that it intends to accomplish via an external service request. Correspondingly, service providers have the general goal of providing the services they were designed to provide — often in exchange for some form of compensation. During the three interaction phases, client goals are represented in different forms by the semantic descriptions created to query semantic service registries and in the semantic descriptions of requests to potential service providers; service provider goals are explicitly represented in the set of effects produced by those services in published service descriptions.

In the overall service-utilization process, service requests (messages from clients to prospective services) are part of the engagement stage. A provider can simply honor these requests (in which case we move directly to the enactment stage) or these requests can serve as preludes to multistep negotiations, in which case engagement culminates in a "handshake" that indicates mutual acknowledgment of a joint contract. Interactions during the service-enactment stage can use one of several alternative protocols for each subphase: initiating service activity, monitoring service processes, and confirming service completion. If the service terminates abnormally after a contract is formed, a final set of protocol interactions can address compensation issues.

The rest of this article summarizes requirements for each phase, and its architecture in terms of abstract protocols for accomplishing phase-specific requirements. By "abstract protocols," we mean a set of message exchanges characterized in terms of abstract semantic concepts and roles. The protocols also identify the agents' state changes that result from such messages. An ontology represents abstractions of the various message types used in these protocols, in terms of the performatives of the Foundation for Intelligent Physical Agents' communication language (FIPA; www.fipa.org/specs/fipa00037/). FIPA performatives represent different speech acts and abstractly identify a sender's intent (such as `INFORM`, `QUERY`, `REQUEST`, and `AGREE`). Due to space limitations, we show only one abstract protocol in detail.

Service Discovery

Service discovery is the process by which a client (service requestor) identifies candidate services to achieve its objectives. It involves three types of stakeholders:

- *service providers* indicate that they will perform services,
- *service requestors* seek services that can accomplish an internal objective, and
- *matchmakers* accept descriptions of available services from providers and match them against requirements from requestors.

Service providers use publish protocols to advertise their services with matchmakers; service requestors use query protocols to ask the matchmakers which services most closely satisfy their needs. For today's Web services, this process is manual: service client developers, rather than requestors, query registries such as Universal Description, Discovery, and Integration (UDDI). In contrast, Semantic Web matchmakers process queries to find appropriate services from among those advertised using Semantic Web language descriptions.[4]

Selecting the abstraction level of the terms in a capability description query to be handed to a matchmaker involves several trade-offs. Typically, a requestor has a specific goal to be achieved at a particular time, whereas service providers publish generalized descriptions of their capabilities that will enable clients to find them. For effective matches to occur, capability queries should be more abstract than the specific goals of the agent at the moment, but they should include information about how the goal should be achieved, and under what constraints, to avoid receiving too many extraneous candidates. This use of abstract capability descriptions in matchmaker queries is one of the reasons for the architectural distinction between discovery and negotiation. Discovery involves cached service-capability advertisements, and clients can afford to do more detailed filtering of and negotiation with potential providers during the engagement process, ultimately leading to an agreement between

the requestor and a valid provider.

We can divide the requirements for service discovery into three parts. *Language requirements* help express capabilities and goals, and they include

- available services' characteristics and constraints (preconditions and fulfillment limitations),
- protocols to be followed during interactions (message semantics), and
- requestor requirements (goals, quality, security, and privacy).

Functional requirements specify the tasks each entity will perform:

- Providers must describe the capabilities and constraints on offered services.
- Requestors must create abstract characterizations of required services to facilitate matching with published capabilities.
- Requestors must locate and interact with peers or matchmakers that can respond to queries for advertised service descriptions.
- Matchmakers must compare descriptions of queries and capabilities.
- Requestors must decide if they can satisfy the preconditions specified in a prospective service's self-description in order to use it.

Finally, architectural requirements identify the various classes of agents necessary to produce the final result of the phase: clients that know what services with which to negotiate. Discovery phase protocols include

- advertising protocols used by service providers to announce capability availability (this advertising can be largely passive, such as postings on Web pages), and
- candidate service-discovery protocols used by requestors looking for services that satisfy their goals.

Matchmakers, like other kinds of registries, can also be federated and organized by community or activity domains.[5] Matchmakers that both find and invoke services as proxies for requestors — called brokers — play the role of the client in discovery, engagement, and enactment. As such, they use different protocols for interacting with requestors, not covered here.

Service Engagement: Negotiation and Contracts

Service engagement is the initial phase of interaction between a requestor and a potential provider. The result of this phase is an agreement between requester and provider such that both parties expect, explicitly or implicitly, that a specific service will be provided by the provider. Although it is during this phase that service contract negotiation occurs, negotiation might be necessary during the later enactment phase to ensure compliance with prior agreements or to resolve disputes.

Functional requirements for engagement vary with the interaction's complexity, but we can nonetheless divide them into four basic areas:

- *Service request formulation*. The requestor must be able to acquire the message and protocol information required for composing valid service requests or participating in service negotiation and invocation protocols. It must also be able to interpret clarification requests and counterproposals.
- *Contract preliminaries*. Potential partners need a means to exchange information about respective goals and capabilities.
- *Contract negotiation*. Potential partners need a means for reaching agreement regarding a service's provisioning.
- *Agreement*. All partners need a means for identifying the terms of an agreement and when an agreement is reached.

Architectural requirements for engagement can vary with the complexity of the negotiations appropriate to the domain, but can include

- *Negotiation protocols*. Protocols that let parties propose and accept or reject aspects of a service agreement must be available in a standard form, and at least one must be appropriate for the particular domain.
- *Negotiation services*. One or both parties can contract with a neutral, stand-alone negotiation expert (similar to an authentication service).
- *Auditing services*. Compliance with a negotiated agreement might require tracking commitments and auditing the parties' enactment activities.

Service-engagement protocols describe the messages exchanged between providers and requestors that eventually result in agreements.

Table 1. Engagement message semantics.

Message type	Performative	From	To	Comment
RequestService	Request	Client	Server	Message content describes desired service, identifies requestor, provides authorization, and conveys nonfunctional preferences and conditions
AcceptRequest	Agree	Server	Client	Acknowledges receipt of request and intention to perform the service
CancelRequest	Cancel	Client	Server	Informs that the client no longer needs the server to perform the requested service
OfferService	Inform	Server	Client	Provides clients with a service description (typically used to make a counteroffer)
AcceptOffer	Inform	Client	Server	Informs server that client agrees to perform the offered service
RefuseOffer	Inform	Client	Server	Informs server that client doesn't agree to the offered service

Three abstract protocols reflect the range of sophistication among the parties to an agreement. The simplest, equivalent to the FIPA query-reply protocol, establishes agreement to a service's provision without any negotiation or formal contract. It's the equivalent of saying, "please provide your service for me" and requires no acknowledgment or agreement from the provider, which automatically attempts to enact its service. The second protocol, which is equivalent to the FIPA request protocol, requires the provider to agree to or refuse a request and results in a non-negotiated but explicit commitment to provide a service.

The third protocol, negotiate-commitment, establishes a formal negotiated contract for a service's provision. This abstract protocol provides a model for the temporal flow and semantic underpinnings of negotiations that can occur between requestors and providers. Each party is required to notify the other if they abandon the negotiation, and explicit recognition of time-outs ensures that no party is left hanging when another party misbehaves or communication fails. The result of a successful negotiation is an explicit shared acknowledgment of a contract, which can then be modeled as a commitment between the parties, as in this dialogue two people might have:

- Requestor: "Will you provide your service for three dollars?"
- Provider: "Only if you pay 10 dollars."
- Requestor: "I will pay five dollars if you provide your service by 5:00 pm."
- Provider: "OK." | "No."

The result of this dialogue, if between Web agents, would be an explicit commitment (perhaps captured by a message trace) between the agents by which the service provider agrees to provide the service under the negotiated terms and the client agrees to any negotiated compenstating actions.

Figure 2 (next page) shows the full abstract model, with the common subdialogue for clarification summarized. Table 1 shows the semantics of the messages used in all the engagement protocols of Figure 2, as well as their respective FIPA performatives.

Service Process Enactment and Management

Once the requestor and the provider agree on the service to be performed, the service is ready to be initiated. The requester determines what information is necessary for requesting the performance of the service and how to react when the service responds with either success or failure.[6,7] Functional requirements for enactment include

- *Response interpretation.* The client must be able to interpret responses to its requests (as described in the service description).
- *Response translation.* A translation must be produced when a requestor and provider use different ontologies for communication.
- *Choreography interpretation and execution.* A semantically grounded language is necessary to describe the temporal constraints among process elements, so that the requestor agent can produce the correct sequence of behaviors for the chosen service.
- *Process mediation and delegation.* To hide choreography differences when interacting clients and services use fixed, incompatible protocols, agents can use process mediation services.
- *Dynamic service composition.* Requestors need support, not only to invoke dynamically dis-

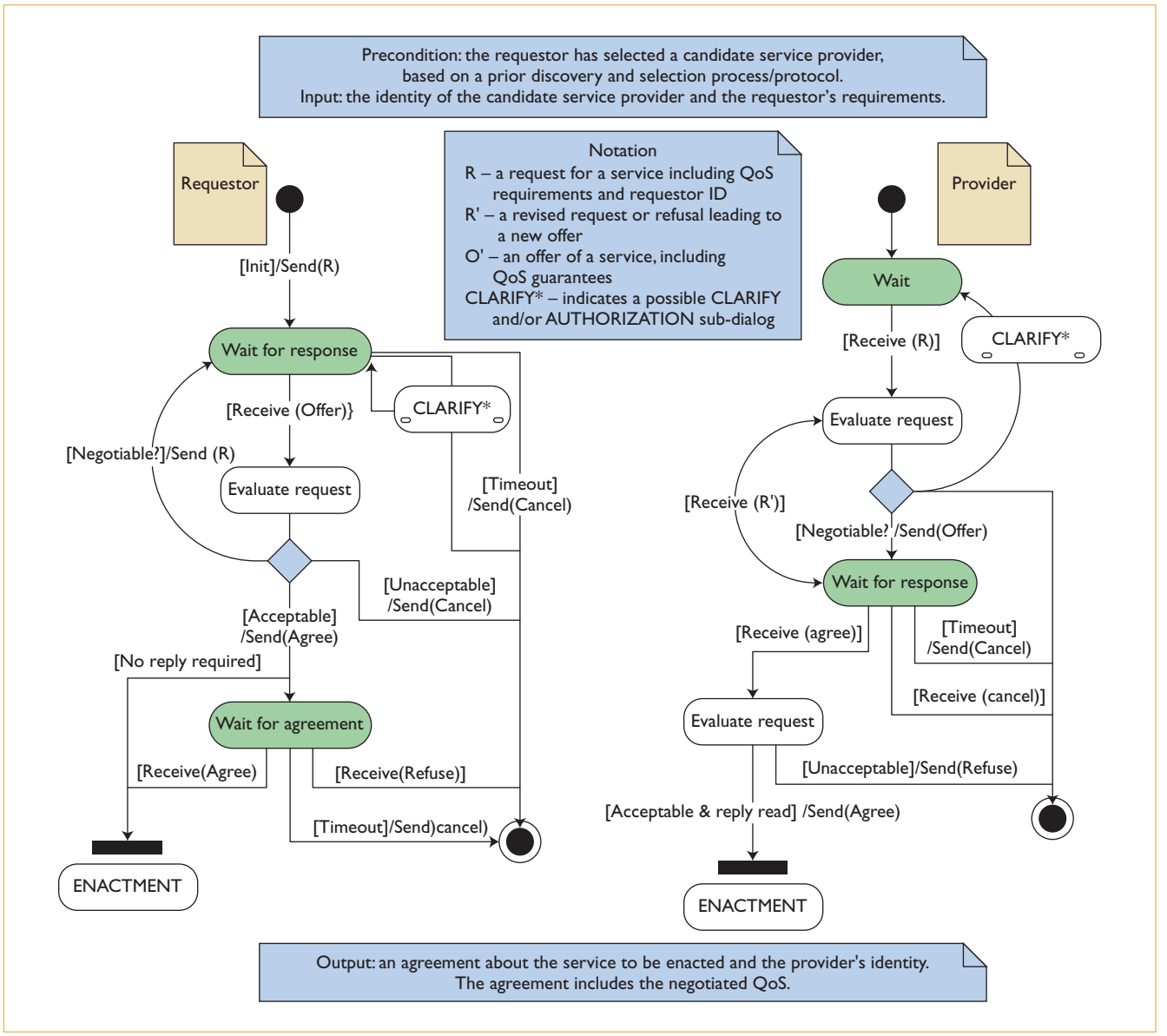

Figure 2. Negotiate-commitment protocol. The left and right graphs show the state transitions of the requestor and provider, respectively. Conditions on transitions are in square brackets. The `CLARIFY` *state in both diagrams represents a subdialog, not shown.*

covered services but also to compose them when no single service meets their needs. A composition can use automated planning techniques to invoke a collection of services for a joint purpose.[6,8]

- *Process-status monitoring and event notification.* Tracking an execution's state can help a requestor determine when and whether it will complete.
- *Service-failure handling and compensation.* Services must provide declarative descriptions of their failure modes and associated means of recovery as well as their types of (and protocols for) compensation.
- *Dispute resolution and compliance.* Clients and service providers can invoke third-party services for conflict resolution, proof of verification, claim adjudication, and settlement compliance.
- *Nonrepudiation, audit tracking, and explanation.* All parties must be confident that a transaction is secure, that all parties are authentic, and that the transaction is verified as final. Systems must ensure that a party can't reject a transaction after this point.

Architectural requirements arise when middle agents provide support services. We will require standardized protocols and standard ontologies for interacting with these agents, which could include

- *process-mediation services* between agents for whom semantically compatible interactions are possible, but whose choreographies don't align (in terms of the number and types of messages to be exchanged);
- *process-scheduling and composition services*, which require ontologies for describing temporal and domain constraints, objectives, and preferences;
- *process-execution and status-logging services* to support execution monitoring, failure recovery, and compensation processes; and
- *policy-monitoring services* to ensure that invoked services follow contractually guaranteed QoS agreements.

In our framework, we've developed abstract enactment protocols for three types of enactment interaction. One of our three enactment protocols assumes synchronous communication, in which a requestor sends a message and then waits for a return message. The other two assume asynchronous communication, in which a requestor sends a message to the provider, but doesn't expect a synchronous message in return.

Each scenario has the same preconditions — namely, the discovery and engagement processes lead to an explicit or implicit service contract. A service requestor initiates the enactment phase by sending an invocation message or by simply completing the engagement phase (when the parties don't explicitly acknowledge the service agreement; see www.swsi.org/swsa for more information on these protocols).

Community Support Services

Another class of infrastructural services will be needed to support communally maintained Semantic Web service activities. In turn, these services will support requirements, such as

- ontology lookup, mapping, and version control services for communities that create shared, centrally managed ontologies and therefore need mechanisms to provide authenticated definitions and mappings among concepts and their derivatives;[9]
- information and access security, privacy, and confidentiality management and monitoring, support for those ubiquitous concerns of commercial and public-access Web services;
- group membership and trust reasoning services that extend simple ID-based authentication, to enable policy and relationship-based access by automated reasoning about known trust relationships between parties;
- community-based preference and reliability reporting services based on collected feedback from service clients;
- policy and protocol management services, like ontology management services, to provide communities with authenticated semantic descriptions of shared policies and protocols, as well as validation and dispute resolution services; and
- lifecycle management services to control the spawning (factories) and management (monitoring and allocation) of service resources.

These requirements and service categories were derived from scenarios using semantically rich service agents, and are not necessarily core parts of the architecture for all environments. However, when present, they play roles in the effective use of other services, and thus they can impact the complexity of the protocols used in conjunction with those domain services. These interactions are topics for future investigation by the committee.

Quality of Service

In Semantic Web service applications, suppliers and customers define binding agreements or contracts among themselves in part by specifying QoS-level agreements, using metrics such as deadlines, accuracy, and cost.[10,11] QoS metrics can affect how services are advertised, can be the topic of negotiation processes, and must be monitored during enactment; thus, when clients' procedures or workflows involve multiple services, the underlying discovery, coordination, and execution systems must be able to monitor QoS measures and control the services accordingly. In complex workflows, these monitors must reason about aggregate measures over the life of the workflow. Enforcement of these metrics can require complex resource reasoning, prioritization, and service resource reallocation. These topics are currently being studied, thus weren't addressed in detail by the committee.

Rather than specific software components, our architectural framework is based on abstract

Further Reading

For the full SWSA report, visit www.swsi.org/swsa/. For additional information on related approaches, visit these links:

- W3C Web Services Architecture WG: www.w3.org/TR/ws-arch/
- OWL-S: www.daml.org/services/owl-s/
- WSMO: www.wsmo.org
- METEOR-S: http://lsdis.cs.uga.edu/projects/meteor-s/

characterizations of protocols and functional descriptions of capabilities. Our objective is to define an interoperability model that can underpin a variety of architectures without prescribing specific implementation decisions. If service developers build concrete components consistent with our proposed abstract protocols, then Semantic Web service clients will be able to interpret published descriptions of these components in terms of the abstract message ontologies and protocols we've developed. This sharing of abstract models, rather than syntactic and procedural agreements among developers, could give developers maximal freedom in building components that can adaptively interoperate.

Our belief is that this approach is consistent with the long-term vision of the Semantic Web at the W3C; we anticipate that our architecture will indicate requirements for Semantic Web service description languages, which are being designed by our sister committee, the Semantic Web Services Initiative Language committee (www.swsi.org/swsl/).

Acknowledgments

We thank the other members of the committee, past and present, and its organizers, who contributed to the ideas presented here: Bob Balzer (Tecknowledge), Fabio Casati (HP Labs, UK), Mike Dean (BBN Technologies), Andreas Eberhart (AIFB, University of Karlsruhe), Dieter Fensel (DERI, Insbruck), Carole Goble (University of Manchester), Mark Greaves (DARPA), Frank McCabe (Fujitsu Laboratories, Sunnyvale, California), Enrico Motta (Open University, UK), Juan Miguel (DERI, Innsbruck, Austria), Chris Priest (HP Labs, UK), Norman Sadeh (CMU), Katia Sycara (CMU), Michael Uschold (Boeing), and Kunal Verma (LSDIS Lab, University of Georgia).

References

1. T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific Am.*, vol. 284, no. 5, 2001, pp. 34–43.
2. D. Martin et al., "Bringing Semantics to Web Services: The OWL-S Approach," *Proc. 1st Int'l Workshop Semantic Web Services and Web Process Composition* (SWSWPC 04), 2004; http://www-2.cs.cmu.edu/~softagents/papers/OWL-S-SWSWPC2004-final.pdf.
3. I. Foster et al., *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration*, tech. report, Open Grid Service Infrastructure Working Group, Global Grid Forum, June 2002; www.globus.org/alliance/publications/papers/ogsa.pdf.
4. K. Sycara et al., "Dynamic Service Matchmaking among Agents in Open Information Environments," *J. ACM SIGMOD Record*, special issue on semantic interoperability in global information systems, vol. 28, no. 1, 1999, pp. 47–53; http://www-2.cs.cmu.edu/~softagents/papers/ACM99-L.ps.
5. K. Sivashanmugam, K. Verma, and A. Sheth, "Discovery of Web Services in a Federated Registry Environment," *Proc. IEEE Int'l Conf. Web Services*, IEEE CS Press, 2004; http://lsdis.cs.uga.edu/lib/download/MWSDI-ICWS04-final.pdf.
6. D. McDermott, M. Burstein, and D. Smith, "Overcoming Ontology Mismatches in Transactions with Self-Describing Agents," *The Emerging Semantic Web: Selected Papers from the First Semantic Web Working Symp.*, IOS Press, Amsterdam, 2002, pp. 228–244.
7. A. Eberhart, "Ad-Hoc Invocation of Semantic Web Services," *Proc. IEEE Int'l Conf. Web Services*, IEEE CS Press, 2004; www.aifb.uni-karlsruhe.de/WBS/aeb/pubs/icws2004.pdf.
8. S. Narayanan and S. McIlraith, "Simulation, Verification and Automated Composition of Web Services," *Proc. 11th Int'l World Wide Web Conf.* (WWW-11), 2002; www.daml.org/services/owl-s/pub-archive/nar-mci-www11.ps.
9. M.H. Burstein, "Dynamic Invocation of Semantic Web Services that Use Unfamiliar Ontologies," *IEEE Intelligent Systems*, vol. 19, no. 4, 2004, pp. 67–73.
10. J. Cardoso et al., "Quality of Service for Workflows and Web Service Processes," *J. Web Semantics*, 2004; http://lsdis.cs.uga.edu/lib/download/CSM+QoS-WebSemantics.pdf.
11. L. Zeng et al., "Quality Driven Web Services Composition," *Proc. 2003 WWW Conf.*, 2003, pp. 411–421; http://sky.fit.qut.edu.au/~dumas/www03.pdf.

Mark Burstein is a division scientist and director of the Human Centered Computing Group at BBN Technologies, and cochair of the Semantic Web Service Initiative's Architecture committee. His research interests include semantic translation and planning system support for Semantic Web services, mixed-initiative distributed systems, knowledge representation, and machine learning. Burstein has a PhD in artificial intelligence and computer science from Yale University. He is a member of the IEEE and AAAI. Contact him at burstein@bbn.com.

Christoph Bussler is executive director of the Digital Enterprise Research Institute in Galway, Ireland, and cochair of the Semantic Web Services Initiative's Architecture committee. His research interests include Semantic Web services,

business-to-business integration, and workflow management. Bussler has a PhD in computer science from the University of Erlangen, Germany. He is a member of the ACM and the IEEE Computer Society. Contact him at chbussler@aol.com.

Tim Finin is a professor of computer science and electrical engineering at the University of Maryland, Baltimore County. His research interests include applications of artificial intelligence to problems in information systems and intelligent interfaces; software agents; the Semantic Web; and mobile computing. Finin has a PhD in computer science from the University of Illinois. He is a member of AAAI and the ACM. Contact him at finin@umbc.edu.

Michael N. Huhns is the NCR professor of computer science and engineering at the University of South Carolina, where he also directs the Center for Information Technology. He recently wrote *Service-Oriented Computing: Semantics, Processes, Agents* (John Wiley & Sons, 2005). Contact him at huhns@sc.edu.

Massimo Paolucci is a senior researcher at DoCoMo NTT in Munich, Germany. His research interests include automatic Web service composition and discovery and the application of Semantic Web service technology to mobile and ubiquitous computing. Paolucci is also a member of the OWL-S coalition, and a former member of the UDDI Technical Committee.

Amit P. Sheth is a professor of computer science at the University of Georgia, the director of the Large-Scale Distributed Systems Lab, and CTO and cofounder of Semagix. He also started the WSDL-S initiative for semantic annotation of Web services. His research interests include the Semantic Web, Semantic Web services, and semantic applications in bioinformatics, healthcare, and risk and compliance. Sheth has a PhD in distributed databases from Ohio State University. He is a senior member of the IEEE and a member of the ACM. Contact him via http://lsdis.cs.uga.edu/~amit.

Stuart Williams is a group manager at HP Laboratories in Bristol, UK, and an elected member of the W3C Technical Architecture Group, which is working to document the Web's architecture. His research interests include protocol design and the application of the Semantic Web for solving interoperability problems between networked software components, such as Web services. Stuart has a PhD from the University of Bath. Contact him at skw@hp.com.

Michal Zaremba is a postdoctoral researcher at National University of Ireland. His research interests include Semantic Web services and its architectures, e-business, enterprise application integration, B2B integration, and business process management. Zaremba has a PhD in industrial engineering from the National University of Ireland and an MSc in computer science and management from the Wroclaw University of Technology in Poland. He is a member of OASIS, W3C, and the ACM. Contact him at michal.zaremba@deri.org.