IEEE Intelligent Systems
www.computer.org/intelligent

Semantic Web Services, Part 1

David Martin, John Domingue, Michael L. Brodie, and Frank Leymann

Vol. 22, No. 5
September/October 2007

Semantic Web Services, Part 1

David Martin, *SRI International*
John Domingue, *Knowledge Media Institute, Open University*

Semantic Web services (SWS) has been a vigorous technology research area for about six years. A great deal of innovative work has been done, and a great deal remains. Several large research initiatives have been producing substantial bodies of technology, which are gradually maturing. SOA vendors are looking seriously at semantic technologies and have made initial commitments to supporting selected approaches. (See the "Frequently Used Acronyms" sidebar for explanations of SOA and other terms.) In the world of standards, numerous activities have reflected the strong interest in this work. Perhaps the most visible of these is SAWSDL.[1] SAWSDL recently achieved Recommendation status at the World Wide Web Consortium.

SAWSDL's completion provides a fitting opportunity to reflect on the state of the art and practice in SWS—past, present, and future. This two-part installment of Trends & Controversies discusses what has been accomplished in SWS, what value SWS can ultimately provide, and where we can go from here to reap these technologies' benefits.

SWS overview

As its name indicates, SWS lies at the intersection of two important trends in the World Wide Web's evolution. The first is the rapid development of Web service technologies, whose long-term promise is to make the Web support shared activities (such as transactions, processes, and the formation of virtual organizations) as well as it supports shared information. In the shorter term, the driving objective behind Web services has been reliable, vendor-neutral software interoperability across platforms, networks, and organizations. A related objective has been the ability to coordinate business processes involving heterogeneous components (deployed as services) across ownership boundaries. These objectives, in turn, have led to the development of widely recognized Web service standards such as WSDL, UDDI, and BPEL.

The second trend—the Semantic Web—focuses on the publication of more expressive metadata in a shared knowledge framework, enabling the deployment of software agents that can intelligently use Web resources.[2] Essentially, the Semantic Web brings knowledge-representation languages and ontologies into the fabric of the Internet, providing a foundation for powerful new approaches to organizing, describing, searching for, and reasoning about information and activities on the Web (or other networked environments).

The central theme of SWS, then, is the use of richer, more declarative descriptions of the elements of dynamic distributed computation—services, processes, message-based conversations, transactions, and so on. These descriptions, in turn, enable fuller, more flexible automation of service provision and use and the construction of more powerful tools and methodologies for working with services.

Because a rich representation framework permits a more comprehensive specification of many aspects of services, SWS can provide a solid foundation for a broad range of activities throughout the Web service life cycle. For example, richer service descriptions can support

- greater automation of service selection and invocation,
- automated translation of message content between heterogeneous interoperating services,
- automated or semiautomated approaches to service composition, and
- more comprehensive approaches to service monitoring and recovery from failure.

Farther down the road, richer semantics can help automate such activities as verification, simulation, configuration, supply chain management, contracting, and service negotiation. This applies not only to the Internet at large but also within organizations and virtual organizations.

SWS research, as a distinct field, began in earnest in 2001. A 2001 article in this magazine by Sheila A. McIlraith, Tran Cao Son, and Honglei Zeng was perhaps the first to point out the potential and importance of bringing Semantic Web techniques to services.[3] Earlier work, however, had considered the construction of knowledge-intensive applications on the fly from online problem-solving methods.[4] Also in 2001, the initial release of OWL-S (originally DAML-S) became available.[5] Other major initiatives began not long thereafter, including WSMO,[6] SWSF,[7] WSDL-S,[8] and the Internet Reasoning Service.[9] But it would be a mistake to consider only these larger, coordinated research efforts. Individual researchers and small teams have also done much valuable work, sometimes drawing on one of these larger efforts, sometimes not.

In this issue

This department's authors include three who have been primarily active in academic research (Amit Sheth, Katia Sycara, and Dieter Fensel) and three whose primary locus to date has been in industry (Frank Leymann, Michael L. Brodie, and Steve Battle). But all have made contributions in both worlds, and all can be regarded as effective bridges between the two worlds. Each has made significant, multifaceted contributions to SWS—defining the vision, leading the research, creating the community and raising awareness, charting relevant directions in industry, and shepherding early deployment. The first installment of this

1541-1672/07/$25.00 © 2007 IEEE
Published by the IEEE Computer Society

department includes two essays giving a broad overview of what trends and requirements motivate the development of SWS. The second, which will include four essays in the November/December issue, will discuss near-term directions and longer-term agendas for the evolution of SWS.

The essays in this issue effectively define service technology needs from a long-term industry perspective. Brodie starts by recognizing that, although industry has embraced services as the way forward on some of its most pressing problems, SOA is a framework for integration rather than the solution for integration. He outlines the contributions that are needed from semantic technologies and the implications for computing beyond services. Leymann emphasizes the broad scope of service-related technical requirements that must be addressed before SWS can effectively meet businesses' IT needs and semantically enabled SOA can be regarded as an enterprise solution rather than a mere packaging of applications. He argues that a great deal remains to be done in several important areas. *—David Martin and John Domingue*

References

1. J. Farrell and H. Lausen, eds., *Semantic Annotations for WSDL and XML Schema*, World Wide Web Consortium (W3C) recommendation, Aug. 2007; www.w3.org/TR/sawsdl.
2. T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American*, May 2001, pp. 35–43.
3. S.A. McIlraith, T.C. Son, and H. Zeng, "Semantic Web Services," *IEEE Intelligent Systems*, vol. 16, no. 2, 2001, pp. 46–53.
4. V.R. Benjamins et al., "IBROW3—An Intelligent Brokering Service for Knowledge-Component Reuse on the World Wide Web," *Proc. 11th Workshop Knowledge Acquisition, Modeling, and Management (KAW 98)*, 1998; http://ksi.cpsc.ucalgary.ca/KAW/KAW98/benjamins3.
5. D. Martin et al., "Bringing Semantics to Web Services with OWL-S," *World Wide Web J.*, vol. 10, no. 3, 2007, pp. 243–277.
6. D. Fensel et al., *Enabling Semantic Web Services: The Web Service Modeling Ontology*, Springer, 2006.
7. S. Battle et al., *Semantic Web Services Framework (SWSF) Overview*, World Wide Web Consortium (W3C) member submission, Sept. 2005; www.daml.org/services/swsf/1.0.
8. R. Akkiraju et al., "Web Service Semantics—WSDL-S," v. 1.0, tech. note, Apr. 2005; http://lsdis.cs.uga.edu/library/download/WSDL-S-V1.html.
9. L. Cabral et al., "IRS-III: A Broker for Semantic Web Services Based Applications," *Proc. 5th Int'l Semantic Web Conf.* (ISWC 2006), LNCS 4273, Springer, 2006, pp. 201–214.

Semantic Technologies: Realizing the Services Vision

Michael L. Brodie,
Verizon Communications

Computer science is in the midst of the biggest shift in its brief, half-century history. Whether by fascination, by genuine delivered value, or by perceived value, the Web has contributed to transforming computer science. The Web is a driving force in the inconceivable growth of data, transactions, devices, and users, which is surpassing the capabilities of conventional computing. If MySpace were a country and each of its users a citizen, it would be the sixth-largest country in the world. Web users perform approximately 7 billion searches per month, using Google for 4 billion of these. In 2006, over 161 exabytes (1.6×10^{19}) of new unique information was generated, more than was generated in the previous 5,000 years. As of 2007, the amount of technical information available doubles every two years. At the current rate, by 2010, it will double every 72 hours. Conventional computer science and human use of information simply won't scale to address this growth. Information and automation are growing rapidly for a reason: they meet real and perceived needs of people, governments, and industry. The challenge to meet these needs is leading to major opportunities for semantic technologies as a pillar of computer science 2.0—the first fundamental redesign in the history of computing.

Two core components of computer science 2.0—services and the Semantic Web—have long histories in computer science. Although their current manifestations provide some value, they're a decade away from maturity. Here, I outline the need for semantic technologies to realize the services vision as an enterprise class technology, in what is called Semantic Web services. However, this is only the beginning. If semantic technologies can address the search and integration challenges that services pose, then we can use them to address search and integration across the computing landscape. But this isn't an easy path. It requires much innovation and hard work, and the major software vendors are already moving in this direction.

Frequently Used Acronyms

BPEL: Business Process Execution Language
DAML-S: DARPA Agent Markup Language for Services
OWL: Web Ontology Language
OWL-S: Web Ontology Language for Services
RDF: Resource Description Framework
RDFS: RDF Schema
SAWSDL: Semantic Annotations for WSDL
SOA: Service-oriented architecture
Sparql: Sparql Protocol and RDF Query Language
SWSF: Semantic Web Services Framework
WSDL: Web Services Description Language
WSMO: Web Services Modeling Ontology

The services vision

In the services vision, a service (an executable program function) can discover and invoke any service anywhere on the Web, independently of the language, location, machine, or other implementation details. Services can range from small to large components and can be combined into composite services to achieve more sophisticated goals. Imagine a service-oriented transportation system in which all components—vehicles, road elements, traffic management devices such as traffic lights, and so on—cooperate to manage traffic flow to minimize congestion, energy consumption, and accidents. Millions of components will produce billions of service calls per minute in one urban area alone.

How will services discover the right ser-

vices that meet specific requirements in such an environment? We're far from that scale at the moment. In Verizon Communications' IT Workbench, one of the world's largest service-oriented environments with over 1,000 services, developers are challenged to find a single service to reuse, aided by a simple metadata service description taxonomy and browsing capabilities. Inability to find a service defeats a core technical benefit of SOA: reuse. In the near term, we need powerful service description languages that allow discovery of the correct service—that is, a service that's an exact match or one that's a good candidate for modification. Augmenting Web services with service descriptions based on semantic languages results in Semantic Web services. What if the match is perfect, but the service doesn't meet the requirement? As service orientation catches on, discovery in scale is critical to realizing the services vision.

Service orientation

Service orientation is a fundamentally new computing model. Although it's based on concepts developed over the past 30 years, it's been in development only since 2000. All major software vendors are developing service-oriented products, infrastructures, and standards using armies of architects and developers. Because service orientation is in its infancy, now is the perfect time to introduce semantic technologies as a pillar for search, match, and other metadata tasks.

In 2006, the major SOA vendors announced what will become the core of SOA development and execution environments—registries and repositories—to manage services metadata. Major vendors such as Oracle, IBM, BEA, and SAP have made initial commitments to support semantic technologies in this context. Vendors currently support languages such as RDF, RDFS, and OWL with modest extensions for processing support—for example, Sparql. Vendors are proceeding cautiously because the requirements aren't fully understood, but, more significantly, semantic-technologies don't currently provide an enterprise-class solution.

It's a significant step for semantic technologies to be adopted by major vendors as a potential pillar of computer science 2.0. All SOA vendors recognize metadata as being the center of their development, and ultimately execution, environments. It's now up to the semantic-technologies community to understand the requirements and provide solutions that will ultimately scale to address not just search, discovery, and match, but all other functions of the service life cycle, including negotiate, adapt, compose, and mediate.

Requirements challenges

Understanding and addressing requirements for semantic technologies in core SOA isn't simple. What requirements for languages, tools, and processing at design time will dramatically improve productivity and reuse? How close a match is sufficient for real reuse when exact match fails?

Dynamic SOA

These challenges are qualitatively different at execution time. The services vision implies dynamic SOA; that is, services discover and execute services at execution time. In industry, this is far from realization, except in the most trivial cases. If a service fails to discover the correct service—that is, not just a technical match but also the service required to deliver the required result—the system could fail. Single failures like this are completely unacceptable in industry; in systems regulated by law, they can involve penalties well in excess of US$100,000. Hence, service discovery and execution must produce the correct result within well-defined, narrow boundaries. How might semantic technologies meet these requirements?

> Single failures are completely unacceptable in industry, so service discovery and execution must produce the correct result within well-defined, narrow boundaries.

Integration

Dynamic SOA is a long-term goal. Enhancing design-time productivity is an immediate requirement, but this too is complex. A myriad of SOA products and infrastructures (at least one from each major vendor) are realizing the service vision. Right now, however, these products don't interoperate, so the services vision is just a vision. It's likely that SOA environments will eventually interoperate in a federated fashion. Ideally, a developer will work from a virtual-services vision that grants transparent access to all services independently of their local environment. At a minimum, this will require all registries and repositories to federate. How will we federate the potentially heterogeneous semantic-technologies solutions provided by the different vendors?

So far, we've considered semantic technologies as an integration solution for core SOA, but there's a much greater scope for integration in SOA. SOA is not, as is often claimed, the solution for integration. It's a framework for integration. Every service call involves mappings between two services, including protocols, data, and process. The mappings are commonly called integration. SOA provides no such mappings. They are provided by vast numbers of adapters and translators that must be matched to requirements—another challenge for semantic technologies.

But let's step beyond core SOA to the rest of computing, which will become "integrated" into SOA. SOA hasn't yet integrated the approximately 40 major software categories, including data management (for instance, in database management systems and data warehouses), metadata management, information integration, content management, application integration, and business process management. Integration has been a central requirement and challenge in these software categories for decades. Approximately half the cost of a major information system project is devoted to integration. If semantic technologies can address the core SOA challenges, they can also address integration beyond core SOA. Indeed, semantic technologies must address these other technologies because core SOA must federate or incorporate them. Hence, the semantic technologies solutions must also map. Since 2000, the major software vendors have committed to using metadata in all software categories. This is good, because metadata is a step toward being amenable to semantic technologies. It's also a challenge because of the need to federate the many heterogeneous metadata solutions.

Collaboration: A potential solution

Computer science 2.0 is redefining computer science in which conventional technologies are reaching their natural limits. Computer science 2.0 must address the unbounded growth of information and automation. For the first time in my career, the major software vendors have recognized semantic technologies' potential to address challenges in their primary product lines that stand in the way of realizing the services vision. The semantic-technologies community must fully understand the requirements posed by this vision and its realization in competing software products and infrastructures.

To achieve scalable solutions, the semantic-technologies community should partner with the systems and database communities, which are used to billion-transaction workloads over exabytes of data. Such collaboration can be mutually beneficial. While the database community can address scalable infrastructures for the services vision, the semantic-technologies community can address the decades-old database challenge of semantic heterogeneity, which remains elusive to this day. At the same time, the semantic-technologies community can step beyond core SOA to provide integration solutions across the computing landscape. These two communities should also find mutual benefit in addressing computer science 2.0's need to deal better with the real world and its real—imprecise, distributed, contradictory, collective, and ever-changing—information. Computer science is inching toward reality.

Semantics for the Cloud

Frank Leymann, *University of Stuttgart*

Service technology is revolutionizing the way we think about IT.[1] A service is an IT artifact (a hardware component or software function) made available at a network endpoint, accessible via (possibly many different) transport protocols and message formats. New middleware (known as a *service bus*[2]) lets requestors use a service without knowing its location and communication requirements. Services are discovered by using business terms instead of technical language, and services qualifying under the same business terms are considered to provide identical functionality. In this sense, the service bus virtualizes collections of interchangeable services that reside "somewhere in the cloud" into single service types.

Composite services

Individual services are of limited value. Composite services provide much higher value than individual services, so this kind of service is predominant in the business world. For example, a service taking care of all aspects of booking a trip—including reserving flights, hotels, rental cars, and so on—is much more valuable than a service that just books a flight. Besides being more convenient to use, composite services typically provide additional functionality (by emergence or by implementation) that the collection of underlying individual services doesn't provide: for example, the trip-booking service will make sure that hotels are booked only when suitable flights are available.

Composite services provide much higher value than individual services, so this kind of service is predominant in the business world.

Typically, composing services into a composite service is based on business process models.[3] These models describe the order in which services are used to achieve a certain business goal. For example, to book a trip, the service first receives the client's itinerary, then books suitable flights and hotels in parallel, and finally returns all confirmations. A business process itself can be turned into a composite service offered for use by others.

A service interacts with the outside world by receiving or sending messages. While a simple, noncomposed service typically receives a single message and (typically) returns a single response, a composite service can receive and send many messages. In this case, the order in which messages can be received and will be sent is important: for example, cancelling a trip before booking it doesn't make sense. So, specifying message order is a key aspect of properly describing a composite service's meaning. It instructs a client in the service's proper use; such "operating guidelines" are an important aspect of the semantics of services.[4]

Applications

Applications help companies perform their business efficiently. An application is the implementation of a set of business processes via software artifacts. Implementing an application based on service technology creates a set of composite services with ingredient services as implementations of steps in business processes.[1] So, service orientation fosters application structures that consist of artifacts at two levels:[3] business processes, and services that the corresponding composite services use. Because a key promise of service technology is increased flexibility, business processes must not assume the use of concrete services. Instead, a business process specifies the service types it needs to execute its overall functionality, allowing the proper services to be discovered at runtime ("late binding") via the service bus.[2]

Executing such an application consists of enacting the business processes. Whenever a step within an enacted business process prescribes using a service type, the service bus discovers and invokes a concrete service. Discovery is based on matching properties that annotate both the steps within the business processes and the set of interchangeable services representing the service types needed. Such properties describe semantically what the business process requires and what the services offer. The spectrum of properties includes

- economic aspects such as price range for service usage and payment methods,
- functional descriptions detailing what the service does beyond giving just its type, and
- technical behavior such as support of a particular transaction model or encryption mechanism.

Precisely specifying such properties is a key aspect of supporting the proper specification of (composite) services' semantics. Annotating business processes and individual services with the correct properties is another fundamental aspect of defining service semantics and vital for ensuring that the overall application behaves properly.

Templates

Applications must become far more adaptive.[5] Years ago, using standard applications such as enterprise-resource-planning or customer-relationship-management systems gave companies a competitive advantage. The advantage even justified changing company business processes according to those the standard applications assumed. Today, most companies use such applications, so the advantage is gone. Companies remember that their competitive advantage is in their individual business processes. So, standard applications must support these processes—that is, it must be easy to adapt a standard application to support a company's business processes.

But the business processes that standard applications impose include the specification of constraints that companies must follow to guarantee functional consistency and integrity. For example, performing a funds transfer requires both successfully crediting and debiting the corresponding accounts. So, a business process typically contains fragments that must remain unchanged to ensure that the overall application semantics isn't violated. Specifying this aspect of adaptability is another fundamental aspect of defining the semantics of composite services. The corresponding specification of an adaptable composite service is called an *application* template: it specifies which ingredient artifacts of a composite service can be changed without jeopardizing the consistency of its overall functionality. The semantic concept of an application template will become the basic building block of adaptable applications.

Solutions

Companies look primarily for solutions, not just applications. An application is an important element of a solution, but it also needs a hosting environment. For example, an application might assume an environment consisting of an application server and a relational-database system. Like applications themselves, the hosting environment must be adaptable to support various companies' operational needs. For example, a large company and a small company will require very different environments: the former might use a host system and high-end middleware, while the latter uses small servers and open source software. Describing the environment an application requires and its degree of adaptability is another fundamental aspect of composite-service semantics: it allows deploying the same composite service in different setups.

The corresponding specification of an adaptable environment is called the *environment* template. Environment templates are described in business terms and must support inferring a concrete corresponding environment in a format that the provisioning software can understand.[5,6] For example, specifying that the environment must be "highly available" must translate into some sort of script that can be run to install the corresponding middleware. Mechanisms to describe environment templates and to transform those templates into installation scripts of proper software components are important to service semantics. The semantic concepts of application templates and environment templates will become the basic building blocks of creating solutions.

> To ensure that proper services are used in highly distributed environments, semantic descriptions of services are of utmost importance.

The cloud

Companies want to focus on their core competencies, so IT infrastructure management must be minimal. Service orientation helps to achieve this goal by directly supporting outsourcing:[5] individual services can run offsite, and complete business processes can be rendered as services and run offsite, which implies that the middleware and hardware hosting a service also run offsite. Reducing the need to manage IT artifacts is one value proposition of *software as a service*.[7,8] To ensure the outsourced solutions' flexibility, some SaaS providers let users adapt the underlying application and environment—that is, they already follow the conceptual structure of a solution.

Composite services might consist of a mixture of services running onsite and offsite (that is, as SaaS). The service bus virtualizes the location of the concrete services that are composed: they are "somewhere in the cloud"—that is, within the set of IT resources that are available as services and can be used to build solutions. These services can be scattered around the globe, and various external providers might host them. Using services from external providers as vital parts of a company's business processes requires particular attention to ensure that the services' nonfunctional properties, such as response time and availability, are comparable with those of the services available onsite. For that purpose, legally binding agreements about these properties—*service-level agreements*—are important. Describing such properties in business terms is mandatory; the focus is on core competencies, so minimizing IT skill is the goal. Corresponding descriptions will be associated with the services available for composition as well as specifications of services required within composite services. So, composing services must consider SLAs too. SLAs are another fundamental aspect of specifying service semantics and making the cloud ready for practical use.

In summary, virtualization is at the heart of service technology: it makes the location of services transparent for requestors, resulting in the impression of a cloud of service. To ensure that proper services are used in this highly distributed environment, semantic descriptions of services are of utmost importance. These descriptions must include semantic descriptions of the adaptability of services in terms of business logic (application template) and operational behavior (environment template, SLA) to be able to tailor services for specific use cases. This will result in the semantic specification of a cloud of services, letting companies easily outsource IT tasks and focus on their core competencies, making them more competitive. Although recent research in Semantic Web services has recognized many challenges and taken useful initial steps, a great distance remains to solve the problems about the broader meaning of Web services. ■

References

1. S. Weerawarana et al., *Web Services Platform Architecture*, Prentice Hall, 2005.
2. M.-T. Schmidt et al., "The Enterprise Service Bus: Making Service-Oriented Architecture Real," *IBM Systems J.*, vol. 44, no. 4, 2005, pp. 781–797.

3. F. Leymann and D. Roller, *Production Workflow*, Prentice Hall, 2000.

4. P. Massuthe, W. Reisig, and K. Schmidt, "An Operating Guideline Approach to the SOA," *Annals of Mathematics, Computing, and Teleinformatics*, vol. 1, no. 3, 2005, pp. 35–43.

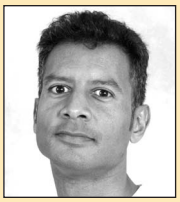
5. F. Leymann, "Combining Web Services and the Grid: Towards Adaptive Enterprise Applications," *Proc. 1st Int'l Workshop Adaptive and Self-Managing Enterprise Applications* (ASMEA 05), FEUP Edições, 2005, pp. 9–21.

6. K. Appleby et al., "Policy-Based Automated Provisioning," *IBM Systems J.*, vol. 43, no. 1, 2004, pp. 121–135.

7. G. Carraro and F. Chong, "Software as a Service (SaaS): An Enterprise Perspective," Microsoft, 2006; http://msdn2.microsoft.com/en-us/architecture/aa905332.aspx

8. A. Dubey and D. Wagel, "Delivering Software as a Service," *The McKinsey Quarterly*, June 2007; www.mckinseyquarterly.com/article_page.aspx?ar=2006&l2=4&l3=43&srid=17&gp=0.

David Martin is a senior computer scientist in the Artificial Intelligence Center at SRI International. Contact him at martin@ai.sri.com.



John Domingue is deputy director of the Open University's Knowledge Media Institute. Contact him at j.b.domingue@open.ac.uk.

Michael L. Brodie works at Verizon Communications, USA. Contact him at michael.brodie@core.verizon.com.

Frank Leymann is a professor at the University of Stuttgart and the director of the Institute of Architecture of Application Systems. Contact him at leymann@iaas.uni-stuttgart.de.