IEEE Intelligent Systems
www.computer.org/intelligent

Semantic Web Services, Part 2

David Martin, John Domingue, Amit Sheth, Steve Battle, Katia Sycara, and Dieter Fensel

Vol. 22, No. 6
November/December 2007

Semantic Web Services, Part 2

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

In part 2 of this Trends & Controversies installment, we continue exploring the state of the art, current practices, and future directions for Semantic Web services. SWS aims to bring Semantic Web technology—for representing, sharing, and reasoning about knowledge—to bear in Web service contexts. The objective is to enable a fuller, more flexible automation of service provision and use and the construction of more powerful tools and methodologies for working with services. For an overview of SWS, please see our introduction to part 1 in the September/October 2007 issue. That introduction also includes references for major SWS initiatives, such as SAWSDL, OWL-S. WSMO, SWSF, and the Internet Reasoning Service. Part 1 also includes essays by Michael L. Brodie and Frank Leymann that discuss service technology needs from a long-term industry perspective. In this issue, we conclude with four more essays.

The first two essays are primarily concerned with nearer-term directions—steps that will let us build out from the current state of the art toward greater adoption and applicability of SWS approaches. Amit Sheth lays out a near-term roadmap of steps that will be essential for industry acceptance of SWS approaches, starting from SAWSDL and current industry practice. (See the "Frequently Used Acronyms" sidebar for explanations of SAWDSL and other terms.) Among other things, he counsels that essential steps are required to make SWS approaches sufficiently accessible and economically attractive to industry. Steve Battle starts with an analysis of OWL-S's strengths and limitations. He then discusses the necessary evolution of business ontologies for SWS. Along with the evolution of business practices, this will allow for Web services and SWS approaches to come together.

The final two essays put forward longer-term agendas for the evolution of SWS. Katia Sycara argues that SWS could benefit from decoupling itself from the basic stack of Web service standards rather than following a more incremental trajectory tied to their evolution. She also identifies two important opportunities in which this strategy could pay off. Dieter Fensel takes a broad perspective, arguing that the characteristics of Internet-scale service usage, and problem solving in general, call for an entirely new conceptualization of some of the core challenges of computer science for the 21st century.

—David Martin and John Domingue

Beyond SAWSDL: A Game Plan for Broader Adoption of Semantic Web Services

Amit Sheth, *Kno.e.sis Center, Wright State University*

After a flurry of research led by the OWL-S, WSMO, SWSF, and WSDL-S groups, the SWS community has taken the first concrete steps toward building a SWS-based solution: the W3C proposed recommendation SAWSDL (www.w3.org/2002/ws/sawsdl), its associated tools and use cases, and initial applications.[1] Where do we go from here? The researchers among us might be fully convinced of the importance and benefit of adding semantics to Web services and impatient to see their research translated into technologies and adapted for real use. However, I believe we might need to do a lot more work before we see substantial adoption of SWS technologies.

First, we must be patient. Both Web services and the Semantic Web are prerequisites for SWS, at least in the form in which SAWSDL or its extensions could evolve. Despite a clear uptick in adoption of Web services by industry, initial success stories, and early adoption of Semantic Web technologies,[2] broad acceptance and adoption could take another three years or so. For the Web services part, the reason is the complexity and confusion resulting from WS-* specifications, some of which (such as security and policy/agreement) are critical and either not yet mature or yet to be broadly adopted. For the Semantic Web, the challenges are to convince practitioners

- that ontologies can be and have been built and are manageable,
- that we can do much with limited semantics and without becoming experts in description logic,
- that we don't need a single ontology for everything and we know how to work reasonably well in a multi-ontology environment, and
- that technology is ready for building robust applications.

We might also have to wait out the current Web 2.0 euphoria, bordering on infatuation. Clearly, Web 2.0 is seductive, easy to use and adopt, and highly valuable for some purposes, but it isn't a solution to all problems and requirements. Technologies such as SWS won't get attention until there is a broader realization of what problems Web 2.0 can't solve and why Web 3.0 (the new code for Semantic Web technologies-enabled Web) is needed.

Second, we must do a better job of understanding, explaining, and managing the complexity of using SWS. Take, for example, the SWS Challenge (http://sws-challenge.org), in which six teams of researchers have implemented or tried to implement a solution to a realistic but relatively simple application using widely different approaches, formalisms,

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

techniques, and technologies. Only one team used SAWSDL. In SAWSDL's defense, this challenge started well before it became a proposed recommendation. But a WSDL-S submission to W3C (see www.w3.org/Submission/WSDL-S), which became the starting point for W3C's work toward SAWSDL, has continued at the same time. More important, the exercise, which has already involved four meetings and spanned well over a year, shows how complex the challenges are, especially when the problem involves legacy systems or interfaces for preexisting services. Lessons from this exercise lead to two observations related to technical issues (tools and methodologies) and business issues (ROI).

Before practitioners (developers and realword users) will embrace SWS, they must have robust tools as well as methodologies to streamline all aspects of managing the SWS life cycle, including annotation, publication, discovery, data mediation, composition or configuration, orchestration, and execution. In the context of SAWSDL, initial versions of most of these have been developed—some by the LSDIS (Large Scale Distributed Information Systems) lab (see www.iswc2006.semanticweb.org/projects/meteor-s/SAWSDL), Kno.e.sis Center (see http://knoesis.wright.edu/research/webservices), IBM (see www.alphaworks.ibm.com/tech/wssem), and others. But these tools must be robust and in a form that developers find attractive to learn and use. An end-to-end open source toolset developed by an international collaboration of researchers could facilitate this process.

Perhaps the most important step is to clearly answer questions any manager or business decision maker will ask: What is the ROI, what is the cost of adapting SWS technology, and what are the concrete benefits? In my view, we must focus on two benefits we've all talked about: service reuse and mediation/interoperability/integration. In the late '90s, our work led to the development of a commercial workflow-management product, METEOR EAppS (Enterprise Application Suite of Tools and Services), that was licensed to numerous organizations. The company I founded, Infocosm, also used it to develop some realworld applications. In the case of existing tasks or activities, I learned that 90 percent of the total time spent developing workflows is related to data-mediation issues (transforming the output of one task into the input of the subsequent task or tasks). Clearly demonstrating how SWS can ease this or equally vexing problems that process developers face will be important. Today's enterprise service bus, application server, and service-based middleware vendors claim to solve the data-mediation problem using XSLT (Extensible Style Sheet Language Transformation)-based transformation. Although this addresses numerous basic problems, as my student Karthik Gomadam put it, "the idea of mediating at the level of instance or between two fixed schemas is hackneyed." To accelerate this technology's adoption, we must focus on issues such as data and process mediation where using semantics will likely provide solid cases of ROI. Efforts that use semantic and ontological approaches for interoperability and data mediation[3] will need to be put side by side with current nonsemantic approaches to drive the point home.

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
SWSF: Semantic Web Services Framework
SOA: Service-Oriented Architecture
UDDI: Universal Description Discovery and Integration
WSDL: Web Services Description Language
WSMO: Web Services Modeling Ontology

Finally, although early examples of realworld SAWSDL-based services already exist (for example, see http://glycomics.ccrc.uga.edu/stargate/web_services.jsp), we will need many more before potential adopters feel that SWS technology has reached critical mass.

As a follow-up to SAWSDL and with encouragement from members of the SWS community such as Dieter Fensel, Charles Petrie and I have started a SWS testbed incubator (www.w3.org/2005/Incubator/swsc). It will likely continue building additional experience in developing SWS-based solutions with the help of the SWS Challenge. Kno.e.sis Center, in some cases in collaboration with other researchers, is looking at three issues discussed in SAWSDL calls that we hope will be components of future solutions:

- Development of SA-REST[4] for semantic annotation of REST (Representational State Transfer) and other lightweight services. Such non-SOAP/WSDL services are widely used, and I see a clear value in providing a semantics-based solution to create powerful *smashups* (semantic mashups[5]) where it's easier to integrate data and services on the client side.
- Supporting precondition and effects in SAWSDL. OWL-S and others have noted this need for some time, and more recently clear use cases have emphasized a need for such support.
- Semantic annotation of policy descriptions. From a business perspective, this will be very important, especially to support nonfunctional or quality-of-service requirements.

In addition, we intend to collaborate with others who are developing an ontology for SWS, which I believe will build on work being done in the WSMO and OWL-S groups. On a more strategic and longer-term path, we hope to continue to push toward understanding and demonstrating semantics' value in making processes more agile, adaptive, and dynamic.

Rather than look for a clear winner among various SWS approaches, I believe that in the post-SAWSDL context, significant contributions by each of the major approaches will likely influence how we incrementally enhance SAWSDL. Incrementally adding features (and hence complexity) when it makes sense, by borrowing from approaches offered by various researchers, will raise the chance that SAWSDL can present itself as the primary option for using semantics for real-world and industry-strength challenges involving Web services.

References

1. K. Verma and A. Sheth, "Semantically Annotating a Web Service," *IEEE Internet Computing*, vol. 11, no. 2, 2007, pp. 83–85.
2. A. Sheth and S. Stephens, "Semantic Web: Technologies and Applications for the Real

World," *Proc. 16th World Wide Web Conf.* (WWW 07), 2007; http://www2007.org/tutorial-T11.php.

3. M. Nagarajan et al., "Semantic Interoperability of Web Services–Challenges and Experiences," *Proc. 4th IEEE Int'l Conf. Web Services*, IEEE CS Press, 2006, pp. 373–382.
4. J. Lathem, K. Gomadam, and A. Sheth, "SA-REST and Smashups—Adding Semantics to RESTful Services," to be published in *Proc. 1st IEEE Int'l Conf. Semantic Computing*, 2007.
5. A. Sheth, K. Verma, and K. Gomadam, "Semantics to Energize the Full Services Spectrum," *Comm. ACM*, vol. 49, no. 7, 2006, pp. 55–61.

Business Semantics in Service-Oriented Architectures

Steve Battle, *HP Labs Bristol, UK*

SOAs offer a powerful level of abstraction promising loosely coupled, modular design. They break away from conventional object-oriented programming, which works for tightly knit developer communities in a well-defined problem space but is less well adapted to the Web. Service interfaces cross organizational boundaries. This isn't merely a technical issue. Service interfaces must cater to the diverse business needs and terminologies of the different parties involved. They also cross functional boundaries; no longer just the province of the IT function, they must be described in a way that the business analyst can understand. At the heart of the Information Technology Infrastructure Library/Service Management (ITIL/ITSM)[1] vision is the assumption that IT powers the business. No longer just an infrastructure provider, the IT function is key to business innovation. We must try thinking about services in new ways before they become hopelessly mired in the baggage of traditional programming techniques.

Service ontology is alive and well

The Semantic Web offers a new modeling paradigm that seems to fit the bill. It embodies the open nature of the Web in its semantics and provides an approach for capturing business vocabulary and ontology. However, we haven't yet succeeded in making this a business reality. The OWL-S ontology for services (originally DAML-S) was one of the first attempts to describe a service interface's semantics. It captures what is known as a *glass-box model*, providing users with a description of what the service does while hiding the gory implementation details. OWL-S defines a vocabulary for describing the actions that a service could perform in terms of inputs, outputs, (pre)conditions, and effects. In this respect, it offers a tantalizing vision of the future, but it doesn't go far enough. I don't intend to criticize OWL-S; it points the way, but much work remains before we have a compelling service ontology for business use. Here, I look at the OWL-S service grounding, process, and profile for inspiration and see how research on related ontologies has grown out of this original vision.

Grounding

The OWL-S grounding in WSDL focuses on Web services' basic plumbing, looking at individual inputs and outputs. The service grounding has the difficult task of overcoming the semantic mismatch between OWL-S and WSDL. The SAWSDL working group has adopted a more incremental approach, expanding WS* standards by extending the existing toolset. SAWSDL builds on an approach touched on in OWL-S, which is to embed semantic annotations directly in WSDL. This results in a more direct grounding that gets straight to the gory details of lifting and lowering messages.

A complementary approach could be to work directly with the WSDL-to-RDF mapping.[2] Although this grounding is important, it provides only basic hooks into the message content, where the real business semantics lies. The challenges are to understand the complex document-exchange style interactions typical in business-to-business interactions and to figure out how to model this business data within a Semantic Web framework.

> To make sense of the business process, we need sophisticated models that capture both message exchanges and their associated real-world activities.

Process

It remains unclear whether the OWL-S process model describes only the necessary (nonexecutable) process constraints on the conversation between service and client or if it provides sufficient information to describe an executable workflow (or orchestration). Efforts on the SWSF (www.w3.org/Submission/SWSF-SWSO) sought to underpin the process model with a deeper ontological analysis based on the Process Specification Language. This approach has an advantage over pure process calculi such as the pi-calculus (which provides semantics for BPEL) because it can capture rich descriptions of conditions and effects.

In practice, service exchanges are typically simple sequences that are weak on control structure and only really understandable in terms of the permissions they carry (for example, you can now deduct money from my account) and the obligations they commit the organization to (for example, you must deliver the goods). To make sense of the business process, we therefore need sophisticated models that capture both message exchanges and their associated real-world activities.

Profile

The OWL-S service profile is surely the place where we finally get to describe the nature of the service being offered, as distinct from its process or plumbing. However, there's little ontological guidance here, other than a handful of nonfunctional properties and a summary of the inputs, outputs, preconditions, and effects as seen in the process model.

Experience with UDDI has proven that we can't describe everything a service does in a simple name (even if it's a uniform resource identifier) and retain any real semantics. WSMO demonstrates the use of rich service descriptions, enabling the matching of service offers and requests in service discovery. These representations of service capabilities must be expressed in domain-specific terms. OWL-S is an upper ontology and was always intended to be extended by users into their specific domains. Business users are only now becoming savvy to the power of extensibility that ontologies offer. Part of the difficulty in describing business data, processes, and services is the sheer variety of business models and processes out there. No single ontology could capture them all. OWL-S and WSMO are unlike existing programming languages that provide all the vocabulary up front.

Where are these emerging business ontologies? One interesting data point is the MIT Process Handbook and its associated OWL mapping.[3] Established in 1991, this online repository includes over 5,000 business activities. In media industries, the Functional Requirements for Bibliographic Records[4] ontology provides a useful base for extensions. The distinctions it makes between works, expressions, and their manifestations coincide with different kinds of rights that must be tracked (copyright, performing, and mechanical rights). In financial industries, the Financial Product Markup Language (FPML; see www.fpml.org) has emerged as a lingua franca for e-commerce in the currency and interest-rate-derivatives markets. These seem a world apart from OWL-S, but really, they highlight the wide span of business semantics in real-world applications, ranging from industry-specific vocabularies to generic high-level ontologies.

Crossing the RDF/XML chasm

XML is the de facto standard for exchanging business messages in SOAs. Moving to RDF message exchange is beneficial where there are strong requirements to communicate structured content unanticipated at design time. An example is the Open Archives Initiative protocol for metadata harvesting, where embedding RDF/XML in the message content is a natural fit.[5] In other Web service applications where the message content is relatively locked down, XML remains the ideal choice, being both compact and easy to validate against an XML schema. Presenting XML and RDF as an either/or choice is therefore a mistake; Web service developers will rightly go with the tools they're familiar with. However, RDF can be viewed as an augmentation of XML providing a new level of abstraction;[6] individual lexical items are cast into their value space, and the document structure is mapped to an object-relational model conforming to the XML schema.

Viewing RDF as an extension of XML enables us to look at new ways in which the Semantic Web can add value to existing SOAs. The ontological approach makes us look at schema with new eyes, viewing them as definitions of business objects assembled into a single message. RDF and OWL provide a syntax-neutral way to model and manage these business objects within a business object repository. This approach offers a future migration path for service providers that want to go native with RDF/XML, avoiding the intermediate mapping into XML while retaining backward compatibility with the XML world.

Service management software provides a comprehensive range of capabilities for managing the entire service life cycle. Even before service implementation, creating detailed, formal specifications could support powerful tools for assessing whether the new service will actually meet real business requirements. Once processes go live, they can be monitored and health-checked against their idealized specifications. The ongoing task of managing (and eventually decommissioning) services requires us to maintain dynamic models of business services, supporting assets, and their virtualized infrastructures. The Configuration Management Database lies at the heart of this ITIL configuration-management process. The ability to map these models into a Semantic Web framework enables ease of integration, sophisticated inference, and complex query that would provide invaluable business intelligence to managers who rely on high-level views of their service operations.

> Upper service ontologies can address only a small part of the problem; the bulk of the work lies in developing industrial-scale ontologies that capture real business semantics.

The take-home message is that upper service ontologies can address only a small part of the problem, laying out the overall modeling framework. The bulk of the work still lies in the development of good, industrial-scale ontologies that capture real business semantics. These ontological extensions come from industrial stakeholders, as seen in vocabularies such as FPML, and represent a healthy pull on the Semantic Web from the real world.

References

1. *The HP IT Service Management (ITSM) Reference Model*, Hewlett-Packard, 2003; ftp://ftp.hp.com/pub/services/itsm/info/itsm_rmwp.pdf.
2. J. Kopeck and B. Parsia, eds., *Web Services Description Language (WSDL) Version 2.0: RDF Mapping*, World Wide Web Consortium (W3C) note, June 2007; www.w3.org/TR/wsdl20-rdf.
3. T.W. Malone et al., "Tools for Inventing Organizations: Toward a Handbook of Organizational Processes," *Management Science*, vol. 45, no. 3, 1999, pp. 425–443; www.ifi.unizh.ch/ddis/fileadmin/user_upload/mklein/malone99tools.pdf.
4. "Functional Requirements for Bibliographic Records," final report, IFLA Study Group on the Functional Requirements for Bibliographic Records, K.G. Saur München, 1998; www.ifla.org/VII/s13/frbr/frbr.pdf.
5. E. Crewe, "Extending the Open Journals System OAI Repository with RDF Aggregation and Querying," *Proc. 9th Delos Network of Excellence Thematic Workshop: Digital Repositories*, 2005; http://delos-wp5.ukoln.ac.uk/dissemination/pdfs/crewe.pdf.
6. D. Booth, "RDF and SOA," position paper, W3C Workshop: Web of Services for Enterprise Computing, 2007; www.w3.org/2007/01/wos-papers/booth.

Untethering Semantic Web Services

Katia Sycara, *Carnegie Mellon University*

Researchers and practitioners are increasingly realizing that semantics would be a beneficial addition to Web services. Since the release of DAML-S (precursor of OWL-S) version 0.5 in 2001, Semantic Web services have seen a lot of activity. Recent notable events include the W3C-proposed recommendation status of SAWSDL and the recent SWS testbed incubator activity. So, this is a good time to pause and think, "What next?" What's the way forward to greater adoption of SWS in industry and diffusion of SWS technologies into the research and business communities?

Next steps

One approach is to follow the current incremental trajectory. SWS ontologies (such as OWL-S) were developed with the realization that SWS was the child of the Semantic Web and Web services. This realization led to the explicit strategy of developing SWS technologies not in isolation, but on top of Web services and the Semantic Web. This strategic goal translated into design decisions to build on technologies at the core of the Semantic Web, such as RDF and OWL, and on Web Services standards, such as WSDL, SOAP, UDDI, and BPEL4WS (BPEL for Web Services). This philosophy is also evident in SAWSDL, which prescribes incrementally adding semantics to WSDL.

Following this incremental view binds the SWS adoption to the adoption of Web services. Web services aim to enable interoperable, dynamic business-to-business interaction; they must, therefore, also strongly support enterprise application integration.

Because organizations' IT infrastructures consist of numerous disparate legacy systems, enterprises want to bridge these systems and enable them to interoperate flexibly. From a technical perspective, an important advantage of Web services over previous distributed computing technologies is that they employ open, platform-independent standards. Adopting open standards promises not only to facilitate interoperability among disparate application platforms, but also to lower the barriers to application integration. The complexity of integration, the *integration challenge*, is increasing much faster than the ability to integrate. Service-oriented architectures and Web service technologies have been heralded as a solution.

As a matter of fact, most attention and studies on whether Web services will successfully deliver on their promise have focused on the integration challenge and IT improvement issues (such as cost reduction and increased infrastructure capability and flexibility). Undoubtedly, introducing semantics into SOA-based implementations can aid in the integration challenge by helping clients understand the intended semantics of services and the meaning of exchanged messages' content. It can also add flexibility in discovery of services that can be reused across a workflow.

However, despite Web services' promise, it's unclear that adoption is widespread. Security is a major concern.[1] Web services allow direct access to a company's applications, exposing corporate networks to security threats. Web services standards' immaturity is another concern,[2] as is the lack of standardized measurement procedures to ascertain the quality of the service provided.[1] Finally, some organizational cultures might be averse to Web services. If this indeed is the case, I am not sure that introducing semantics to Web services would go far enough to rapidly overcome the barriers. This would indicate that the value and power of Semantic Web services lie elsewhere.

Two big opportunities

Incrementally adding semantics to Web services standards might be in line with business caution vis-à-vis semantics (this caution possibly has its roots in the AI experiences of the '80s). However, focusing solely on inserting semantics incrementally might mean missing out on bigger opportunities to show SWS' value. I see at least two such opportunities:

> A need exists to model informal business requirements in ways that make it feasible to translate them into precise business-service specifications.

- supporting *higher-level enterprise modeling* to enable integration of business goals with services that computationally enact those goals, and
- supporting increased *autonomous and dynamic* Web services behavior.

Besides being big opportunities, these two areas are attractive because, for various technological and cultural reasons, they're not tightly coupled (technically) or committed (culturally) to using the current Web services standards infrastructure.

Business process semantics

The business community has been increasingly interested in the possible impact of Web services on ROI and other business issues, while decrying the dearth of studies on these issues.[3,4] Such big-picture issues include

- operational issues, such as improving quality and productivity;
- managerial issues, such as improved decision making, planning, and resource management;
- strategic issues, such as support for business alliances, business growth, innovation, and product differentiation; and
- organizational issues, such as support for organizational change, building a common vision, and facilitation of organizational learning.

To better understand this impact, we must understand how business goals translate into computational business processes that enact those goals. This reflects a top-down view.

IT is under pressure to justify the value it adds to the enterprise, and its continued efforts to do so have echoed this realization in a bottom-up way. This has resulted in a clamor for automated integration of higher-level business functionality to the IT services that implement high-level business goals and relations. Achieving this integration will require technologies and standards for modeling and re-engineering the enterprise—not just integrating business applications.[1] A need exists, then, to model informal business requirements in ways that make it feasible to translate them into precise business-service specifications, including operational interfaces and rules for procedures, timing, integrity, and quality. Such modeling must be driven from the top down, directly from business requirements. The models express intended business semantics in a vertically integrated way. The modeling would provide a functionality that's entirely understandable from a business perspective; it would depend on business context, goals, and operational standards, but shouldn't depend on the technology used to implement them. The models would provide business value directly relating to business purposes and could be understood and used without knowledge of underlying IT artifacts.

Current Web services proposals don't enable the semantic representation of business relations, contracts, or business rules in a machine-understandable way. Current business-process languages aim at representing and describing business processes and achieving a joint understanding of a process involving several parties. However, they're at low abstraction levels and don't provide formal business semantics. For example, the Business Process Modeling Notation[5] is a graphical notation with XML as the underlying representation, lacking formal semantics. BPEL is well suited for describing communication behavior from a particular individual service's point of view and can map to the code to execute the process representation. The technical literature spends much time discussing BPEL's shortcomings (for example, it can't represent choreographies) and the Web Services Choreography Description Language's inability to map to BPEL or to support scenarios where the number of participants is known only at runtime. For all the nomenclature similarities between service and business processes, current process languages don't concern themselves with trying to link workflows and message choreographies to high-level business notions, such as organizational goals, commitments, and contracts.

Web service autonomy and dynamism

The cell phone's emergence as a primary access point for Web content and services has fueled the need for Web services to operate more autonomously and dynamically. Ultimately, the growing Web services infrastructure (the Web services stack) facilitates the emergence of agreements between programmers and the coding of those agreements. The result, however, is an inherently brittle infrastructure that's difficult to reconfigure to accommodate new Web services or react to failures and that's inevitably expensive to maintain. For example, service registries don't directly support sufficiently flexible, dynamic, automated service discovery; nor do they support service context, such as location-dependent or context-sensitive information. The way to overcome the infrastructure's brittleness is to make Web services more autonomous by letting them reconfigure their interaction patterns. Any increase in autonomy lets Web services react to changes while minimizing direct intervention from programmers. Crucially, the lack of explicit semantics prevents Web services from acting autonomously, which prevents them from dynamically discovering one another or understanding what each other's messages mean and what tasks each performs.

A possible way forward

The presence of semantics is a sine qua non for making progress in these two opportunity areas, and work by the SWS research community is under way.[6–8] The skeptic asks, "But what about the barriers to adoption that you mentioned?" A number of answers come to mind. Regarding the barriers to current adoption of Web services, decoupling SWS from the Web services stack might mitigate some challenges and make the new technology nimbler. Additionally, neither area of opportunity involves an accumulation of legacy technologies, so embracing the SWS promise wouldn't entail abandoning past investment (unlike the case of the application integration challenge). Moreover, the higher level of conceptualization called forth would increase understanding of security issues and how security and other service-quality concerns might tie in with business goals and commitments. Finally, if the SWS approaches can provide breakthroughs that add value in orders of magnitude, as I anticipate, this alone will provide compelling incentives to deal with any apparent barriers.

> The notion of 100 percent completeness and correctness doesn't make sense anymore; the underlying fact base is changing faster than any reasoning process can process it.

The SWS community has the opportunity to lead the development of standards and specifications for concepts such as contracts, business-level agreements, and commitments and how they relate to the processes that computationally enact them. Current standards can't express these processes because they require more semantically rich representations. Additionally, by supporting dynamism and autonomy, SWS can support service decentralization and adaptive business processes that support changing business needs. This new orientation of SWS work, decoupled from the current Web services stack, could lead to a more flexible and expressive services infrastructure that meets business needs.

References

1. D. McGoveran, *Embracing SOA: The Benefits of Integration Independence*, report 20060125, Alternative Technologies and TIBCO Software, 2006; www.tibco.com/resources/solutions/soa/embracing_soa_wp.p
2. H. Deitel et al., *Web Services: A Technical Introduction*, Pearson Education, 2003.
3. R. Hailstone and R. Perry, "IBM and the Strategic Potential of Web Services: Assessing the Customer Experience," ICD white paper, IBM, 2002.
4. A. Manes, *Web Services: A Manager's Guide*, Addison Wesley, 2003.
5. *Business Process Modeling Notation Specification*, tech. report, Object Modeling Group, 2006; www.bpmn.org/Documents/OMG%20Final%20Adopted%20BPMN%201-0%20Spec%2006-02-01.pdf.
6. N. Desai et al., "OWL-P: A Methodology for Business Process Modeling and Enactment," *Proc. 7th Int'l Bi-Conf. Workshop Agent-Oriented Information Systems III* (AOIS 05), LNCS 3529, Springer, 2006, pp 79–94.
7. M. Paolucci and K. Sycara, "Autonomous Semantic Web Services," *IEEE Internet Computing*, vol. 7, no. 5, 2003, pp. 34–41.
8. R. Masuoka, B. Parsia, and Y. Labrou, "Task Computing—The Semantic Web Meets Pervasive Computing," *Proc. 2nd Int'l Semantic Web Conf.*, LNCS 2870, Springer, 2003, pp. 866–881.

Computing for the World: Incomplete, Incorrect, but Requested!

Dieter Fensel, *University of Innsbruck and STI International*

Dave Brown from Worcester Polytechnic Institute recently asked whether the work on problem-solving methods of the late '90s is dead.[1] I answered, "Yes, but only in a certain sense." Heuristic problem solving for closed, fixed applications doesn't deserve much attention anymore. However, heuristic problem solving has begun to reappear on a worldwide scale. Two distinct approaches define its essence:

- a goal-driven approach where the solution process is only partially determined ahead of time and actually decided at runtime on the basis of available data and services, and
- a heuristic approach that gives up on the

absolute notion of completeness and correctness to gain scalability.

Both aspects seem to be essential for working in an open, heterogeneous, distributed, and fast-changing environment such as the World Wide Web. Therefore, areas of computer science that initially aimed toward 100 percent complete and correct solutions start to incorporate these principles as soon as they want to be usefully applied in a Web context.

In a large, distributed, and heterogeneous environment, the classical ACID (atomic, consistent, isolated, and durable) guarantees of the database world no longer scale in any sense. It's completely out of scope to assume that the Web could remain static until some crawler has searched it in its entirety to produce a defined query result. Therefore, a simple read operation in an environment such as the Web, a peer-to-peer storage network, a set of distributed repositories, or a space can't guarantee completeness in the sense of assuming that if data wasn't returned, it didn't exist. It's often the case that the data just wasn't found in the amount of time given to search for it. Similarly, a write can't guarantee that a consistent state replicates immediately to all storage facilities at once.

What holds for simple data lookup holds in an even stronger sense for reasoning on Web scale.[2] The notion of 100 percent completeness and correctness as usually assumed in logic-based reasoning doesn't make sense anymore because the underlying fact base is changing faster than any reasoning process can process it. So, we must develop a notion of usability of inferred results and relate them with the resources that are requested for those results.

Even in information retrieval, the notion of completeness (recall) becomes more and more meaningless in the context of Web-scale information infrastructures. It's unlikely that users will request all the information available about and relevant to a certain topic because this might exceed the amount of information processing they're investing in achieving a certain goal. So, instead of investigating the full space of precision and recall, information retrieval is starting to focus more on improving precision and ranking of results.

In a world of billions of services, the cost of finding the "optimal" service might outweigh the benefits. Pragmatic service-discovery approaches will focus on utility (stopping the search when a service is found that's good enough to fulfill a request). Also, it's unrealistic to assume that semantic descriptions of services are correct and complete—that is, that they duplicate a service's functionality at the description level.

In all areas, you must define the trade-off between the guarantees you provide in terms of service-level agreements (completeness and correctness are just examples of very strong guarantees) and what this requires in terms of assumptions and computational complexity. Various relationships exist between these factors, and they're worth further analysis and more detailed study. Different heuristic problem-solving approaches are just different combinations of three factors:

- *Service-level agreements* (or goals) define what type of result the problem solving must provide. Do we request an optimal solution, a semi-optimal solution, or just any solution?
- *Assumptions* describe the generality of the problem-solving approach. Assuming that the problem has only one solution allows stopping the search for an optimum once a solution has been found. Instead of a global optimization method, a much simpler heuristic search method can be used in this case and would still deliver a global optimum.
- *Computational complexity* (scalability) denotes the resources required to fill the gap between the assumptions and the goals.

Computer science in the 20th century was about perfect solutions in closed domains and applications. Computer science in the 21st century will be about approximate solutions and frameworks that capture the relationships of partial solutions and requirements in terms of computational costs—that is, the proper balance of their ratio. In a nutshell, heuristic problem solving on the Web scale will likely soon become an entire research area. This shift is comparable to the transition from classical physics to relativity theory and quantum mechanics, where relativistic notions and the principle limits of precision replace the notion of absolute space and time.

First steps in this direction are efforts such as WSMO, which translates earlier modeling approaches for heuristic problem solving such as KADS,[3] CommonKADS,[4] and the Unified Problem-Solving Method Development Language (UPML)[5] to conceptual models for open, distributed, and heterogeneous problem solving. WSMO is augmented by a family of formalization languages called the Web Service Modeling Languages:

- WSML-Core captures RDFS.
- WSML-DL (Description Logics) captures OWL.
- WSML-Rule captures the Rule Interchange Format (RIF).
- WSML-Full provides a full-fledged, layered logical framework.

As a third cornerstone, WSMX (Web Service Execution Environment) and SEE (Semantic Execution Environment) provide reference architectures for scalable problem solving on the Web.

Recent work derives simple description means for services from this framework. WSMO-Lite combines

- SAWSDL as means to reference semantic descriptions,
- a small meta-ontology in RDFS to define the pragmatic meaning of a semantic description (that is, specifying whether it's a condition or an effect of an operation), and
- RDF syntax.

MicroWSMO will drive this into a simpler, useful framework for describing services semantically. It will provide structured tags (concepts with attributes) that link to ontologies in the background as means for communities of service users to tag services.

Returning to Brown's question with which I started this essay: problem-solving methods are alive in the same way a baby is alive when it leaves its closed and cozy world. A child's transition to a new, open world can be inconvenient and painful. Its a new world is full of risk, but also large enough to provide for future growth and development towards its full potential. ■

References

1. D. Fensel, *Problem-Solving Methods: Understanding, Development, Description, and Reuse*, LNAI 1791, Springer, 2000.

2. D. Fensel and F. van Harmelen, "Unifying

Reasoning and Search to Web Scale," *IEEE Internet Computing*, vol. 11, no. 2, 2007, pp. 94–96.

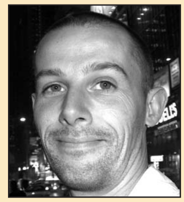
3. A.T. Schreiber, B.J. Wielinga, and J.A. Breuker, eds., *KADS: A Principled Approach to Knowledge-Based System Development*, Academic Press, 1993.

4. G. Schreiber et al., *Knowledge Engineering and Management: The CommonKADS Methodology*, IT Press, 1999.

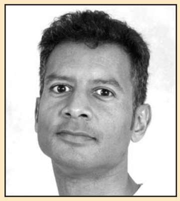
5. D. Fensel et al., "The Unified Problem-Solving Method Development Language UPML," *Knowledge and Information Systems*, vol. 5, no. 1, 2002, pp. 83–131.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.

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

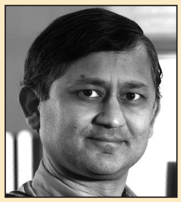


Steve Battle is a researcher at HP Labs, Bristol. Contact him at steve.battle@hp.com.

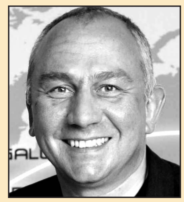


John Domingue is deputy director of the Open University's Knowledge Media Institute. Contact him at j.b.domingue@open.ac.uk.

Katia Sycara is a research professor at Carnegie Mellon University, where she directs the Intelligent Software Agents and Semantic Web Technologies Lab, and holder of the Sixth Century Chair in Computing Science at the University of Aberdeen. Contact her at katia@cs.cmu.edu.



Amit Sheth is the LexisNexis Ohio Eminent Scholar director of Kno.e.sis Center at Wright State University and an IEEE fellow. Contact him at amitpsheth@ieee.org; http://knoesis.wright.edu/amit.



Dieter Fensel is a computer science professor at the University of Innsbruck and president of Semantic Technology Institute International. Contact him at dieter.fensel@deri.at.