

Will the Semantic Web Quietly Revolutionize Software Engineering?

Greg Goth

For many people, the Web's future is the Semantic Web. According to the World Wide Web Consortium, the Semantic Web "provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries" (www.w3.org/2001/sw/). Basically, it's an attempt to express Web content in the usual natural-language version for human consumption while annotating it semantically for computers to process. These annotations are based on *ontologies*—representations of concepts in a domain and the relationships between those concepts.

Along with the more public debate about the benefits the Semantic Web is—or isn't—delivering to the Internet, a related undercurrent of discussion is growing about software engineering's future. In the vanguard of this discussion, some researchers and software engineers are advocating that Semantic Web technologies should be key components of the next paradigm of software development.

These advocates, however, are discovering that the ongoing debate about the Semantic Web's ultimate usefulness is being fought over the user experience—whether or not the principles behind the Semantic Web are just too complex to be truly useful to the bulk of Internet users.

This debate obscures the more subtle qualities the Semantic Web technologies bring to developers. It is they, these pioneers say, and not end users, who could benefit most from the more formal development models and enriched search capabilities that semantic technologies offer.

The Semantic Web contradiction

IBM senior information technology architect Philip Tetlow is one of the advocates of *Semantic Web-enabled software engineering*. Tetlow was coeditor of several W3C papers explaining SWESE (for example, see www.w3.org/2001/sw/BestPractices/SE/ODA and www.w3.org/TR/sw-oosd-primer).

Tetlow says he becomes pessimistic when people talk about the Semantic Web being an infant. The thinking underlying the Semantic Web has been around over a decade, he says, and its foundational technologies have been stable for three years. Yet there has been no mass clamor for it, as there has been for other technologies that have received market acceptance much faster.

"If you were to ask, 'In five years' time, will a small number of people be seen as the prophets who foresaw this sea change?' 'Probably not' is the answer," Tetlow says. "Ask, 'Has the time come and gone for Semantic Web technologies?' Some would say yes; some would say no."

Perhaps the confusion regarding Semantic Web technologies' suitability for advanced software engineering stems from the contradiction inherent in the term "Semantic Web." On the one hand, people see the Web as the great leveler, the provider of information from any given data source to any given user anywhere in the world. Technologies such as AJAX frameworks and scripting languages are becoming ever easier to use and reaching wider audiences of developers, often those with no formal training in software development. On

the other hand, the technologies underpinning the Semantic Web are, in the mathematical sense, very formal and unforgiving of programming errors.

Two Webs becoming one

David Hyland-Wood, entrepreneur-in-residence of the University of Maryland's Semantic Research Group, says you need to think of the Web as two Webs to appreciate the properties of both and to fully comprehend where Semantic Web technologies fulfill their promise. According to Hyland-Wood, the first Web is the one the user sees. The other Web, he says, is the infrastructure of a "continuing formulation of further abstraction—abstraction we're doing in programming languages, not close to the machine anymore." This second Web is made to order for Semantic Web technologies, Hyland-Wood says.

"The Internet is starting to be used in the way it was intended," says Hyland-Wood, "with Web 2.0 applications and mashups, where the application the user sees is really made up of stuff from all over the place, assembled at the client in the way the user wants to see it. And we got there, importantly, by systematically, over the decades, increasing the level of abstraction in the process."

"Information is coming from various sites and it's all virtual," Hyland-Wood says. "Everything is hidden behind a URI. So the more we abstract, the more we need to manage that abstraction. ... The tools and techniques for managing that mean we occasionally have to have machines communicating with each other. And the way we do that is with a web of data, not documents." According to Hyland-Wood, this means using Semantic Web tools and techniques—"RDF, RDF Schema, SPARQL, OWL—all URI accessible, all manipulated by the REST architecture." (For an explanation of RDF and these other Semantic Web terms, see the sidebar.)

Eventually, Hyland-Wood says, these two discrete Webs will indeed fuse back together as "the Web, as we figure out how to use those machine-generated data and human-generated data in the same context." "Eventually," he says, "every human interaction with a machine will be

Semantic Web Terminology

- **OWL:** Web Ontology Language, used to describe classes and how they relate to each other over distributed knowledge documents, knowledge bases, and applications (see www.w3.org/TR/owl-features)
- **RDF:** Resource Description Framework, "a language for representing information about resources in the World Wide Web. It is particularly intended for representing metadata about Web resources, such as the title, author, and modification date of a Web page, copyright and licensing information about a Web document, or the availability schedule for some shared resource" (www.w3.org/TR/rdf-primer). RDF also serves as a flexible data exchange language.
- **RDF Schema:** A vocabulary description language for RDF (see www.w3.org/TR/rdf-schema)
- **REST:** Representational State Transfer, an architectural style for distributed hypermedia systems such as the Web (see <http://rest.blueoxen.net/cgi-bin/wiki.pl?FrontPage>)
- **SPARQL:** A query language for RDF (see www.w3.org/TR/rdf-sparql-query)

augmented by machines talking. It will all come back into one Web, but that's the next generation after this one."

Semantic technologies in the research and enterprise communities are still in the early adoption phase. However, they're being merged slowly into the larger technology continuum and Internet-based applications such as database search. A 2006 report (www.ncrr.nih.gov/CRInformatics/EHR.pdf) that MITRE prepared for the US National Institutes of Health, for example, outlines some of the research emerging from semantics-specific vendors as well as acquisitions of semantics technologies by major players. For instance, in 2003, Google bought Applied Semantics and IBM bought UML pioneer Rational Software. "Semantic Web vendors are in the early commercialization phase, while the majority of ontology languages and related methodologies are emerging from applied research and beginning to enter early commercialization," the report concluded.

Public-sector technologists are also expanding their use of Semantic Web technologies. In the US, NASA used RDF and OWL to build an expertise location service for its employees. A hallmark of this POPS (People, Organizations, Projects, Skills) project was the decision to build an RDF federation of three existing

databases at NASA facilities instead of building a new database from scratch.

"We quickly had three disjoint databases integrated into one RDF federation, using unique identifiers in each database to relate relevant information within the federated store, including unique identifiers for NASA personnel and projects," NASA researchers wrote in a paper describing the project (<http://2006.xtech.org/schedule/paper/147>). "Since this data does not change rapidly, we are able to periodically repopulate the RDF federated store without interrupting service at the host databases and without incurring the performance penalty of live, dynamic querying of the host databases."

Many possible routes to mass acceptance

The philosophical and logistical foundation of SWESE is the belief that the open and increasingly distributed nature of general Web-enabled development, such as that engendered in the POPS project, will require a new definition of systems engineering and design.

"When I talk with my colleagues and friends who I would consider to be software engineering old-school gurus," says Tetlow, "they typically talk about software in the confines of closed-systems definitions." He says this closed-system

model is essentially the design and implementation of systems within well-defined boundaries of a particular problem, such as an accounting system in a specific organization with specific goals.

However, Tetlow also says that advanced software engineering techniques such as formal methods failed to revolutionize development even within those confines, pre-Internet. At these methodologies' peak, which Tetlow places in the mid- to late 1980s, they failed to achieve significant adoption because they were too abstract for the average developer.

The promise of Semantic Web technologies, Tetlow contends, is that they're using the same underlying computational theory of these older methods. For example, RDF triples are based on a mathematical representation of formal concepts, without using mathematics syntax. Because of these properties, Tetlow says developers can use the Semantic Web languages, which retain mathematical formality, "in an exceptionally informal way." He adds, "the real bonus for me is essentially lowering the barrier around formal definition in software."

The particular means by which these barriers will be lowered are still unclear. Some SWESE advocates believe the Semantic Web-enabled model will emerge from large enterprise and government agency projects that must integrate multiple disparate domains, lines of business, languages, and vocabularies. Others believe that using subsets of Semantic Web technologies for smaller tasks, such as software maintenance and reuse of existing code bases, will be more tenable than the daunting task of creating massive ontologies and their associated full-scale Semantic Web frameworks.

Elisa Kendall is CEO and founder of Sandpiper Software, which specializes in ontological technology. She's also a coeditor of one of the W3C SWESE papers mentioned earlier. Kendall believes that Semantic Web technologies are best used in large-scale integration projects in which other technologies fail to capture the wide variety of data inherent in multiple domains. One such project was InSight, a remote-monitoring tool developed by General Electric Infrastructure's Water & Process Technologies unit. At

the 2006 Semantic Technology Conference, GE researcher Mark Dausch illustrated the technological, organizational, and geopolitical complexities facing the InSight development team, including the absorption of technologies from companies that GE had purchased (www.semantic-conference.com/2up_BW/Dausch-Mark-bw.pdf). The team chose an ontology-based approach, using Sandpiper's Visual Ontology Modeler.

Kendall says the obstacles in integrating management data across these various boundaries also presents a mother lode of opportunity in which Semantic Web technologies play a key role. "In the paradigm where you're moving from hard sales to services, for a company like GE, managing water and power and space equipment, the potential has got to be huge," she says. "There are opportunities in businesses like mining and agriculture, oil and gas, electrical power, and core power management in hotels. Across all these factors, there's no way to manage it without something like an ontology, or some sort of really rich model."

Ontologies: Time to lighten up?

Over the long term, Kendall says this ontology-based approach can add tremendous value to the software engineering process by abstracting business vocabularies away from coding algorithms and processes. Developers won't

have to reinvent the wheel with every new device a particular application is meant to run on.

"Business vocabularies shouldn't change that much over time," she says. "Interfaces change, platforms change, user devices change, and so on, but business vocabulary is stable enough that you should be able to abstract it."

However, Kendall also acknowledges the conundrum inherent in large corporations building rich ontologies. The way to build a global Semantic Web is through open access to these resources. But why should a company that has invested large sums of money in such an ontology share a valuable resource with competitors?

"We're having a business dilemma around that," she says. "We're asking ourselves those tough questions, like what do we want to do for our customers and what do we make publicly available? How do you determine the trade-off? We've been erring on the side of not making money, but that can't last. There's a huge business challenge. I don't know the answer."

"Fortunately for us, some of our work has been government funded, and the government wants us to make some things available. Maybe that's the right answer, that the government is going to have to be involved."

Bruce Spencer, senior research officer at the National Research Council of Canada, says another risk facing ontologically based engineering is the opposite of the "walled garden" proprietary model. "With Semantic Web technologies," Spencer says, "we could build ontologies and find that nobody wanted to use them."

Hyland-Wood says those interested in pursuing SWESE should step back from looking at daunting ontology orthodoxy and hearken back to the aphorism coined by Semantic Web pioneer Jim Hendler, a professor at Rensselaer Polytechnic Institute: "A little semantics goes a long way."

According to Hyland-Wood, some researchers are seeking to define "the minimal possible subset of OWL that we can make use of simply and easily and cleanly." He adds that this subset "is more

**The obstacles
in integrating
management data
across these various
boundaries presents
a mother lode of
opportunity.**

than RDF Schema, but it's much, much, much less than OWL Full or OWL-DL" (two other subsets of OWL).

The subset, called OWL Tiny, was characterized by University of Southampton researcher Steve Harris as "the subset of OWL Lite that contains property characteristics and identity relations from OWL Lite. It is the part of OWL that cannot produce conflicts."

Part of Hyland-Wood's own research, on developing a software maintenance methodology using Semantic Web techniques, demonstrates another attempt to "lighten" Semantic Web technologies. Although his experiment uses an OWL-DL ontology, it does not employ an OWL reasoner, citing scalability issues (because a reasoner must recompute all inferences following a change to the underlying data). By using a technique of SPARQL queries upon RDF stores, Hyland-Wood and his colleagues avoid the reasoner performance issues.

Beyond OWL and RDF

At first glance, the construction of a SWESE ecosystem might appear to focus on the core technologies of OWL, RDF, and SPARQL. However, other semantics-based technologies might supply vital resources to a wider range of developers.

For example, the University of Sheffield's TAO Project (Transitioning Applications to Ontologies, www.tao-project.eu) aims to define a low-cost route to transitioning legacy systems to

open semantic service-oriented architectures. Such transitioning will enable semantic interoperability between heterogeneous data resources and distributed applications. TAO is using another Sheffield project, GATE (General Architecture for Text Engineering), as a test case in building an ontology for developers worldwide. GATE is one of the world's foremost open source natural-language-processing projects.

TAO coordinator Kalina Bontcheva says the TAO team aims to provide the global community of GATE developers improved, semantically based information (using GATE's language-processing facilities) for several knowledge access areas. These areas include automatic generation of documentation pages from the ontology, natural-language-based queries for semantic search, semantic-based filtering of forum postings, and—like the NASA POPS project—expertise location.

"All of these scenarios depend on the availability of a domain ontology and semantic-annotation tools, which can annotate all software artifacts with the mentions of concepts and properties from this ontology," Bontcheva says. "This is the core research challenge in TAO, especially how we involve developers in the knowledge elicitation process, without expecting them to be experts in building ontologies."

Bontcheva's words illustrate that projects such as TAO might represent the forefront of the fusion of the "two

Webs." Also, Semantic Web technologies could benefit developers of all ability levels in a widely dispersed community on any project, with their own contributions added back automatically to the project's body of knowledge.

"We envisage all these knowledge access scenarios, where such inexperienced developers will be able to search better through the wide array of software artifacts, or locate other developers who can help them with questions, etc.," Bontcheva says. "In a sense, the TAO perspective on this is that effective software development requires effective knowledge management. This is why it makes sense to take semantic technologies, such as knowledge access tools originally developed for knowledge management in organizations, and adapt these to the specifics of software development practices and teams."

BM's Tetlow succinctly sums up the gist of the SWESE effort: "99.99 percent of the code alive and kicking on the planet today is inefficient for one reason or another, but nevertheless the dream is still alive. If you look at the work the W3C has done, the best minds in the world came together for the past decade, and their sole purpose was to provide the best technology possible for the betterment of the people of this planet. And they have done a job par excellence with the Semantic Web."

IN BRIEF

The Possible Futures of Service Software

Dave and Virginia Dorenbos

Will your career center around service software? Will you be inventing, aggregating, supporting, and using service software? Eighty percent of our economy is services, so software to support and provide them is a significant opportunity. Carnegie Mel-

lon West and two University of California entities (IT Services and UC Berkeley's Haas School of Business) organized a conference to provide background and define possible futures for service software. The New Software Industry: Forces at Play, Business in Motion, took place in April in Mountain View, Califor-

nia. The presentations are at <http://west.cmu.edu/sofcon>, and videos of the talks and panels are available.

The definition of a service is broad, including maintenance support for traditional applications, outsourcing IT operations, and aggregating specialized knowledge from and about users to resell.

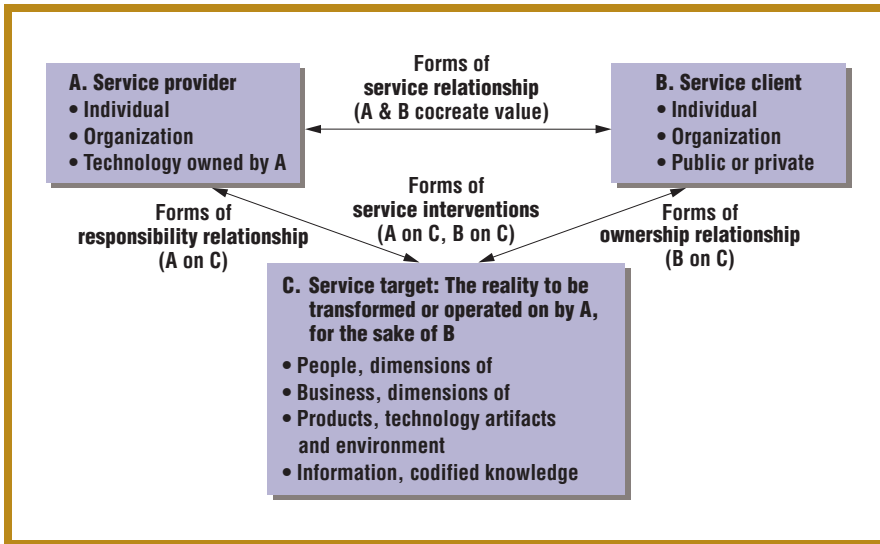


Figure 1. A view of service as a system of relationships.

Table 1

How services contribute to a company's bottom line*

| Inflection points | | Shape of the curve [†] | | |
|-------------------|--------|---------------------------------|------|------|
| First | Second | First | Then | Then |
| 22% | 63% | Up | Down | Up |

* From Michael Cusumano's presentation, showing the contribution of services as a percentage of sales to operating income for all product categories.

† Up: a rise in service percentage improves operating income; down: a rise in service percentage worsens operating income.

IBM's Paul Maglio gave a definition that covers all these cases:

Service systems are value co-creation configurations of people, technology, internal and external service systems connected by value propositions and shared information (such as language, laws, measures, models).

Maglio's pictorial expansion (see figure 1) of this definition is helpful in understanding all the possible variants a service can take. (This figure is based on a diagram Jean Gadfrey published in *Productivity, Innovation, and Knowledge in Services* [Edward Elgar, 2002]).

This change to providing a service, rather than a purchased product, via software is evidenced by the declining number of public companies selling software as a product: from 400 in 1997 to 200 in 2004 (see Michael Cusumano's presentation). Furthermore, the three major product players, Microsoft, Oracle, and SAP, take 75 percent of the product revenue (see Ray Lane's presen-

tation). Even as these companies take this share of product revenue, they're executing their own strategies on services.

Software engineers who will have major impact in terms of innovation and profitability will need deep knowledge of the methods and tools for building service software but will also need the breadth to contribute to defining opportunities for services. Bob Glushko pointed out that the deep knowledge must cover the front stage that envisions an attractive service interface and a back stage that enables flexibility and customization in the front stage. To develop services, you'll need a toolset appropriate to your market (for example, Ruby on Rails) and processes and methods that can create a sustainable service.

Paths to service software

Speakers highlighted two paths to service software. The first is creating services around products. For the major product players, and for everyone trying to leverage their products into services,

the revenue from services has sweet, and sour, spots. According to Cusumano's research, services make a positive contribution to operating income, but there are inflection points on the operating-income curve. Table 1 shows that when a company providing products shifts to also providing services, there's a period, when services constitute 22 to 62 percent of sales, where service income does *not* improve operating income for the effort expended. Beyond the second inflection point (63 percent), service income again improves operating income as a business reaches dominance in its market.

The second path is productizing services by making a service so unique that it becomes the sole solution. Amazon.com is the premier example; it has found ways to enrich a basic catalog of ISBN numbers that are difficult for any competitor to match.

For service software, open source is a key to both creating customer confidence and lowering adoption costs. Those who wish to innovate and lead will need solid (deep) knowledge of open source software and respected participation in an open source community (see the presentations by Jim Herbsleb, Kim Polese, and Tony Wasserman).

Value is subtle; when creating software as a service, you'll need broad knowledge to define value in ways that resonate with investors. Investors will expect you to understand your market and the details of how money will be made (see the presentations by Bill Burnham, Scott Russell, and Ann Winblad). Opportunities exist in both the uncharted spaces and existing spaces reorganized for service.

Highlights

The following are highlights of some of the presentations.

Timothy Chou

A possible path to creating services is to mine the deep, deep, deep Web. There's opportunity in the 1,000,000+ Tbytes of data inside enterprises as compared to the 100 Tbytes available on the Web. Figuring out the value held in the d-d-deep Web and mining it provide a significant opportunity.

David Messerschmitt

To broaden the opportunity to provide service software, the industry must collaborate. For instance, cellular-network operators complain that there are too many operating systems for mobile devices. Could we create a bigger market opportunity for everyone by pushing standards?

Ray Lane

The Web initially had just two dimensions of use: the “near” for email and the “far” for browsing. According to Ray Lane, the Web today is growing in six dimensions:


- near—content sent to me,
- far—browsing and surfing,
- here—the Web with you everywhere,
- weird—people to devices,
- B2B—business to business, and
- D2D—devices to devices.

This growth is enabled by the conversion of 45 million content readers in 1996, to 1 billion-plus readers and writers in 2006, to the future Web. The user participation in content creation increases the Web’s value and creates opportunity for innovation in all six dimensions.

Craig Mundie

Use multicore desktops, which are mainly idle, to provide a service’s local presence such that it’s reliable, predictable, humanistic, performant (that is, that performs well), context aware, model-based, personalized, adaptive, immersive, and visually rich.

You can prepare for service software creation by creating your own reading list. Paul Maglio pointed out that Michigan Technological University is preparing students for service software through eight new courses: World of Ser-

vice Systems Engineering, Service System Design and Dynamics, Analysis and Design of Web-Based Services, Human Influences on Service Systems, Service System Operations, Optimization and Adaptive Decision Making, Project Planning and Management for Engineers, and Managing Risk. A reading list based on these courses should provide a solid foundation. For overview material, visit the Web sites for the 2007 Software as a Service Conference (SaaScon 07, www.saacon.com/live/48) and the Web 2.0 Summit (www.web2summit.com). 

Dave Dorenbos has worked at Motorola, Gould, and Rockwell International in a variety of development and research roles. He’s also the Industrial Advisory Board chairperson for the Software Engineering Research Center (www.serc.net) and a member of the IEEE Software Advisory Board.

Virginia Dorenbos has worked at GTE and Rockwell International and taught math and computer science at Elmhurst College.

Contact the authors at david_dorenbos@ieee.org.

Call for Articles

Be on the Cutting Edge of Artificial Intelligence!

Publish Your Paper
in IEEE Intelligent Systems

IEEE Intelligent Systems
seeks papers on all aspects of
artificial intelligence, focusing
on the development of the latest
research into practical, fielded
applications. For guidelines, see
[www.computer.org/mc/
intelligent/author.htm](http://www.computer.org/mc/intelligent/author.htm).



The #1 AI Magazine
www.computer.org/intelligent

**Intelligent
Systems**
IEEE