



# An Alternative to the Top-Down Semantic Web of Services

Sudhir Agarwal • Karlsruhe Institute of Technology

Charles Petrie • Stanford University (emeritus)

**T**he Semantic Web<sup>1</sup> aims to allow service providers to semantically annotate service descriptions, enabling automatic discovery and composition, which could in turn reduce the time and manual effort otherwise needed to program desired applications. However, after many years of effort, such services are notably lacking. The original Semantic Web vision proposed the development of domain ontologies for this purpose.<sup>2</sup> Although the Semantic Web is distinct from semantic Web services, we conflate them because they've both been predicated on top-down, usually formal semantics. Using this approach, we should be able to repurpose semantic Web services because we'd have a common vocabulary that could enable interoperation among applications developed by different people. Unfortunately, this approach hasn't lived up to its initial promise. Here, we look at an alternative way to achieve such interoperability for individual users and developers.

## Why Does the Semantic Web Fall Short?

Why do we lack a common set of ontologies for general use by open Internet users? Such ontologies are capable of arising in specific vertical domains owing to common need and agreement, as well as in "industrial service parks" in which companies create ontologies for their own use. In a previous article,<sup>3</sup> one of us (Charles Petrie) noted that McDermott's famous objection<sup>4</sup> applies only to the ideal of creating a general open Semantic Web with ontologies that are reusable for all purposes. More specific semantic systems are certainly possible. But semantic

services are scarce, especially on the open Internet.

The original Semantic Web idea envisioned general-purpose "agents" that could automatically find, compose, and act on domain-specific information from various webpages.<sup>2</sup> Implementing this vision required that service providers annotate websites and services with explicit, machine-understandable conceptual models.

However, we clearly can't rely on providers to describe their data in machine-understandable form. If they had seen an added business value in it, they would have done it years ago with Web services. If they haven't provided their data even in a machine-parsable format such as XML, they won't provide it in machine-understandable formats such as RDF and the Web Ontology Language (OWL), because the latter are more complicated than the former. Enterprise semantics haven't happened. Rather, we should look to end users to be both consumers and developers of services.

With existing approaches, end users are unclear on where they should save their own semantic annotations, or how they can share them with others, and they can't change provider-owned webpages. Furthermore, as is the case with most programming, even those programmers who can manage formal ontologies have found it easier to develop their own rather than reuse existing ones. Experience with the Semantic Web Services Challenge<sup>5</sup> showed that over a five-year period, almost no one could simply reuse a previously successful ontology without close collaboration with its authors.

Various approaches have sought to solve this problem, the most current of which is *linked data and services*, which seeks to address it by completely separating webpages from semantically described data.<sup>6</sup> The Linked Data Services project<sup>7</sup> seeks to build on this technique's popularity to produce semantic descriptions for services. The extent to which enterprises will use this technique is still unclear. However, by taking this approach, linked data pushes end users almost completely out of the picture.

### Processes not Data: Principles

Explicit ontologies are neither necessary, sufficient, nor feasible – as anyone who has tried to write or consume a formal ontology knows. However, by considering how people actually use the Web, a different approach for end users becomes obvious. Rather than the top-down approach of ontologies in the Semantic Web, we advocate a bottom-up approach beginning with websites and the end users who browse them using standard browsers. Website owners aren't involved, and annotations consist of exactly those necessary for a particular application.

The Web is no longer simply a set of documents. Rather, it's a set of distributed and networked processes that can require multiple interactions with the user during execution. These processes deliver information and might cause effects in databases and in the physical world. But many of these processes currently exist only in the minds of end users.

To access information or functionality hidden in the "deep Web," users must often perform several steps – for example, submitting Web forms with certain values and clicking links in a certain order. Semantic approaches that focus on browsing miss this important fact. Due to their underlying data-oriented view,

the Semantic Web and linked data largely fail to deal with the deep Web, especially for end users.

Once a user has tediously found the right path through a sequence of websites for a particular goal, he or she should be able to find it much faster and easier the next time. Saving the process consisting of these steps in a reusable way is thus valuable. Another reason that it is the process rather than the data that should be cached is that the paths that lead to information or functionality don't change as frequently as the information or the functionality themselves.<sup>8</sup>

This crucial insight that it is the browsing processes that are valuable and should be saved, rather than the resulting data, informs the following principles of our approach to a new kind of semantic application:

- *Build descriptions on already useful applications.* Users should have useful applications in the first place, rather than be made to reuse semantics for applications unknown when those semantics were developed. In fact, users know what they want to do, and the steps that they take to get their results constitute existing valuable applications.
- *Make users' implicit browsing processes explicit.* These existing applications can be made explicit by capturing the steps that people take to extract website information repeatedly.
- *Make these browsing processes sharable.* People other than the developer should be able to reuse process descriptions, making a completely new type of information available on the Web that until now resided only in the minds of end users.
- *Use natural language descriptions.* Explicit descriptions of browsing processes should have a human readable syntax so that end users can find and comprehend them at a later stage.

- *Make the processes composable.* Users should be able to search suitable scripts that they can directly invoke or use as components in script compositions.

Semantic applications should create semantics. If useful browsing processes can be integrated with one another for new useful applications, users will have an incentive to agree on standard terms with constrained usage – that is, they will develop semantics.

### Web Automation: One Approach to Bottom-Up Principles

At the Karlsruhe Institute of Technology (KIT), we've experimented with these bottom-up principles and have developed proof-of-concept prototypes that convince us they're feasible. The basic approach is to use scripting for Web automation.

A script is a process that coordinates the execution of a set of websites and the data flow among them. Scripts can simulate users' actions in a Web browser (such as clicking, selecting, or entering text) to automate navigation between webpages and Web form submissions, essentially making large keyboard macros. Web automation tools let users access deep Web resources. Users without previous programming experience can develop Web automation scripts using natural language commands and programming by demonstration.<sup>9</sup> This makes any user a potential script developer.

### Script Creation

A user creates a script to accelerate his or her ability to navigate a group of websites and thus carry out or solve a recurring task. A browser plug-in that can record a user's browsing actions can create a script containing those actions and the order in which they should execute. To make the script usable even with different input values, as well as for privacy

reasons, the user can replace the constant input values with variable names. Because a script is directly executable, the user can always run a newly created script to test it before saving it in the script repository.

We've found it useful to start with open source Web automation script frameworks such as the IBM CoScripter (<http://coscripter.researchlabs.ibm.com/>) and then identify lacunae. The CoScripter system executes scripts and provides a language for simplifying "screen scraping." People can immediately use scripts others have developed because they're a form of controlled natural language that make understanding and testing easy. Users can access and navigate the script repository using the browsing interface, and can access existing descriptions and refine them with the browser editor.

CoScripter doesn't offer all the functionality we need. The language doesn't support instructions for aligning newly extracted information to the already extracted information, nor are the databases – called "scratchtables" – persistent. We might need persistent scratchtables for integration across scripts because they provide a standard way of associating variables with data in websites and sharing these associations as well as information extracted from websites with other users. Additionally, a library of operations is necessary for transforming the data from one format to another – for example, a temperature value in Fahrenheit to its Celsius equivalent. Unfortunately, we believe IBM has missed a bet by ceasing development on CoScripter when it needed only a few more features to be widely useful.

### Script Integration with Common Functions

At KIT, we've used several prototypes to experiment with these ideas; one

that showed that a portal of integrated scripts is feasible for finding the cheapest airline flights has proven particularly successful (see <http://km.aifb.kit.edu/sites/fairmarket/hubInputForm.html>). Others can use this portal to link their scripts, add new websites to a script, or add new scripts to a composite script.

The main challenge for composition is coordinating the data and control flow among multiple concurrently running scripts, where each script can invoke more than one website or even other (component) scripts. If a data connection exists between two scripts, the data coming from the first script must be transformed into the second script's format on the fly.

In our airline flight portal project, we've demonstrated that end users will make the effort to unify variable names in return for being able to re-use standard conversion functions in the portal. This not only addresses the data conversion problem but also distributes the effort of mapping script variables to all users instead of concentrating it only at the script's developer. A future objective is for everyone to be able to extend the functions library. In the portal, end users also have access to a basic set of data transformation functions.

### Bottom-Up Emergence of Semantics

The script-based automations described so far aren't Web services (<http://tinyurl.com/webservdef>) because they run client-sided and have no machine-readable descriptions; yet there is actually a use for shared semantics and thus a reason such semantics will arise.

For one thing, variables in different scripts must be shared for script integration to occur. As mentioned, shared library functions provide an incentive for users to do this standardization and do it distributively.

Subclass relationships will arise naturally, as we can see in the

following scenario. Suppose that Joe has a script *Mypeeps* that searches for all faculty members in a given department at a given university. Joe uses the variable *FacultyMember* to define the members of the resulting sets. Joe would like to standardize based on the date of the last published work.

Jack has a script *Birthdays* that, given the *Location* by country of a *Person*, converts their *Birthday* to US format, and Joe would like to use this function for the last publication date of the faculty members. First, Joe realizes that he must adjust his script to extract the *Location* of the universities and perhaps use another function that returns the country given the city and postal code. Joe could just change *LastPublicationDate* to *Birthday* in his script (as in the airline portal) and reuse the date function, but this would be the semantically wrong approach.

Fortunately, the persistent scratchtable function has been extended to include class relationships. Joe states that *LastPublicationDate* is a subclass of *Date*, and coordinates with Jack to do the same with *Birthday* so that both scripts can use the same function. Thus a community and semantics arise because there is a reason for them to do so.


**T**he experiments at KIT convince us that scripting is a powerful mechanism that can lead to a community-developed semantics as an alternative to top-down formal semantics. We encourage everyone to experiment with scripts, using the principles we've espoused in this article to develop practical semantic Web-based services.

Clearly, much work remains to be done. Future work includes the issue of how best to represent such simple variable names in the functions that scripts use, and perhaps use these as keywords in search. We're also

studying whether and how the actual flows in the scripts might be reified. Giving scripts URIs seems like a good idea. With appropriate methods for analyzing scripts, providers could draw important guidelines for improving their websites and develop innovative ideas for new ones.

One important limitation of this approach is that since we have no machine-readable descriptions, automated Semantic Web service composition isn't applicable. But this could change with enough people researching the problem from a user perspective. For instance, it might turn out that some formal languages, such as Datalog or process calculi, might be good ways to capture browsing-based processes.

Approaches other than scripting could also work: this is only one method that seems likely to work. We advocate holding workshops to collaborate on different methods for semantic applications that create semantics from already useful Web applications.

It's a promising approach that is very likely to gain momentum in contrast to the Semantic Web and has the potential to empower users to be an emergent collective of developers, as Tim Berners-Lee originally intended. 

### References

1. M. Hepp, "Semantic Web and Semantic Web Services," *IEEE Internet Computing*, vol. 10, no. 2, 2006, pp. 86–88; [www.heppnetz.de/files/ieee-ic-no-sw-without-sws-final-official.pdf](http://www.heppnetz.de/files/ieee-ic-no-sw-without-sws-final-official.pdf).
2. T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific Am.*, vol. 184, no. 5, 2001, pp. 34–43.
3. C. Petrie, "The Semantics of Semantics," *IEEE Internet Computing*, vol. 13, no. 5, 2009, pp. 80–82; [www-cdr.stanford.edu/~petrie/online/peer2peer/semantics.pdf](http://www-cdr.stanford.edu/~petrie/online/peer2peer/semantics.pdf).
4. D. McDermott, "A Critique of Pure Reason," *Computational Intelligence*, vol. 3, no. 1, 1987, pp. 151–160; <http://dx.doi.org/10.1111/j.1467-8640.1987.tb00183.x>.
5. C. Petrie et al., *Semantic Web Services Challenge: Results from the First Year*, Springer, 2008.
6. C. Bizer, T. Heath, and T. Berners-Lee, "Linked Data – The Story So Far," *Int'l J. Semantic Web and Information Systems*, vol. 5, no. 3, 2009, pp. 1–22.
7. S. Speiser and A. Harth, "Integrating Linked Data and Services with Linked Data Services," *Proc. 8th Extended Semantic Web Conference*, Springer, 2011, pp. 170–184.
8. S. Agarwal, "iBookmarks: Synthesis and Execution of Solution Templates for Efficient Usage of Recurring Web-Process Combinations," *Proc. 5th IEEE Int'l Conf. Semantic Computing*, IEEE Press, 2011, pp. 35–38.
9. A. Cypher et al., *No Code Required: Giving Users Tools to Transform the Web*, Morgan Kaufmann, 2010.


---

**Sudhir Agarwal** is a senior researcher and project leader at the Karlsruhe Institute of Technology, Germany. His research interests include distributed service-based systems, semantics, ontologies, process calculi, and temporal logics. Agarwal has a PhD in computer science from the University of Karlsruhe (now part of KIT). Contact him at [sudhir.agarwal@kit.edu](mailto:sudhir.agarwal@kit.edu).

---

**Charles Petrie** retired from Stanford University as a senior research scientist with the CS Logic Group. He is a guest professor at Karlsruhe University, Germany, and at the University of St. Gallen, Switzerland, for 2012. Petrie has a PhD in computer science from the University of Texas at Austin. He was a founding member of the technical staff of the MCC AI Lab, founding editor in chief of *IEEE Internet Computing*, founding executive director of the Stanford Networking Research Center, and founding chair of the Semantic Web Services Challenge. Contact him at [petrie@stanford.edu](mailto:petrie@stanford.edu).

---

 Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.