The Pragmatic Web: Preliminary Thoughts

Munindar P. Singh*
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-7535, USA
singh@ncsu.edu

Abstract

The core mission of the semantic web, namely, to enable information to be shared across the web, faces significant challenges. The challenges come from the difficulty of capturing semantics in a manner that is reusable across applications, the priority of process over data, the importance of interaction, and the critical need for accommodating user context. Overcoming these challenges takes us from the realm of semantics and brings us into the realm of pragmatics, which we view as supervenient upon the semantics. We claim that when the vision of the semantic web is realized, it will be via the pragmatic web.

*The ideas described herein have been in development for a few years. Most recently, this work was supported by the National Science Foundation under grant ITR-0081742 and by awards from IBM and Cisco. I have benefited from discussions with several colleagues and students, especially including Mike Huhns, Mladen Vouk, Bin Yu, Zhengang Cheng, Ashok Mallya, Mike Maximilien, Raghu Sreenath, and Pınar Yolum. Only I am responsible for the opinions expressed, of course.

1 Introduction

In his contribution to the encyclopedia of unified science, Charles Morris popularized the science of *semiotics*, namely, the study of symbolic systems [1938]. Morris defined semiotics as consisting of three components: syntax, semantics, and pragmatics. Syntax deals with the structure of symbols, semantics with their meanings, and pragmatics with their contexts of usage. These terms were picked up by the early logicians and computer scientists and, especially the first two, are frequently the objects of attention in computer science.

It is commonplace to view the web as a symbolic system. As its structural aspects have become better understood, the semantics is drawing increasing attention. This position paper focuses on the pragmatics that lies beyond the semantics of the web. Notice that just as the syntactic aspects have not died off when the semantics is considered but have instead become more rigorously defined, so too will the semantics flourish as the pragmatics expands. One might think of the pragmatics endeavor as a special direction within the scope of the semantics endeavor now under way.

Interestingly, in common parlance, the word pragmatic also refers to practical, and in this case the double meaning of the title is intended.

A brief history of data. In the early days of computing, data and applications were inextricably intertwined leading, as we all know, to poor maintainability and upgradeability. By separating out the data from the applications, database management systems enabled each to exist independently of the other. The hope really was that data could be used for applications other than those for which it was intended and that applications could access databases other than those they were designed to access. This hope was partially realized in that, with sufficient effort, you can build new applications that access old data and feed new data into old applications. However, in its fullest form, this hope was dashed, because although access methods became almost standardized on SQL for relational databases, the semantics of the data remained as ad hoc as ever.

The same problem occurred for the Internet. In the early days, applications and data formats were ad hoc. The standardization on HTML enabled browsers through which people could access information anywhere on the web. This is fine as long as humans are engaged in understanding and processing the information, but doesn't lend itself well to automation. So it is natural that semantics will draw increasing attention on the web as well.

Pragmatic concerns. However, if there is one lesson to be learned from the long history of databases, it is that it is practically impossible to describe data well enough for it to be used in arbitrary applications. It will be worthwhile to take this lesson to heart. Doing so has some major ramifications on how we should proceed in our investigations on the web.

- The current hype surrounding the semantic web is reminiscent of previous hype cycles associated with artificial intelligence (AI). Like for AI, researchers seem to be dreaming up scenarios where the semantic web could change how information on the web is prepared and used. Each such scenario is within the scope of current technology to implement as a

demonstration system. However, when the scenarios are put together and expected to be automated based on general-purpose representations, they pose a huge engineering challenge, especially because current abstractions and theories are quite limited.

- The best hope for the semantic web is to encourage the emergence of communities of interest and practice that develop their own consensus knowledge on the basis of which they will standardize their representations. For example, such standards have emerged in narrow areas of personal information management, e.g., with the vCard standard. Such standards have also emerged in the handling of email attachments.

- The above will still remain inadequate, however, because it addresses the knowledge acquisition and standardization challenges for the current focus on semantics, but ignores the need for fundamental new representational frameworks. We cannot engineer the semantic web merely by doing more of what we are doing with current buzzwords: XML, RDF, or even DAML, because these after all no more than new syntax for ideas that originated in the early days of work on data modeling and knowledge representation.

What we need is a special form of semantics, namely, pragmatics, which will have the additional abstractions to enable the engineering of open information systems over the web.

Organization. This position paper develops its case by motivating services as essential to the vision of the web, identifying technical challenges, and taking an initial cut at some principles to address those challenges. Some of the arguments are admittedly exaggerated, but only for the sake of making a point. Section 2 introduces our take on service composition, its necessity, and the inadequacies of current approaches. Section 3 synthesizes the technical challenges from the study of Section 2, identifies some principles for addressing them, and argues that multiagent systems provide the right basis for an approach in which to realize those principles.

Although the main argument of this paper is new, some of the constituent claims have been studied in our previous papers and columns on related subjects. Specifically, the following publications are relevant and widely accessible: [Jain et al., 1999; Singh, 1998, 2000, 2001; Singh et al., 2001].

2 Web Service Composition

The essence of the semantic web is to enable access to web services. Web services enable application development and integration over the web by supporting program-to-program interactions. The current examples of the semantic web might not involve a formal description of services, but that is the obvious direction it is taking, and rightly so. In other words, when the semantic web is realized and applied, it will mostly be through services.

Importance of composition. Although there can be some value in accessing a single service through a semantically well-founded interface, the greater value is clearly derived through enabling

a flexible *composition* of services. Composition leads to the creation of new services from old ones and can potentially add much value beyond just a nicer interface to a single service.

From a business perspective too, intermediaries that primarily offer access to a single service would have a tough time thriving or even surviving. Airline travel agents are a case in point. Traditional travel agents provide a nice user interface: friendly and with a human touch, but little more. However, airlines do not like to pay commissions for services that merely repackage their offerings. As a result, they compete with the travel agents and reduce their commissions, slowly squeezing them out of the business. This is as one would expect where the offerings are conceptually simple, especially for frequent customers. By contrast, package tour operators, who combine offers from airlines and other vendors, can do all right. In other words, the increased complexity due to subtle compositions is essential for intermediaries to flourish.

Describing and invoking services. It is natural to think of description and invocation as two sides of the same coin. A description is how a provider offers its service; an invocation is how a customer exercises an offered service. Web Services Description Language (WSDL) is a proposed standard for describing services, essentially in terms of the method signatures they support [Christensen et al., 2001]. Simple Object Access Protocol (SOAP) is a proposed standard for invoking services, essentially through remote method invocations over HTTP [Box et al., 2000].

The DARPA Agent Markup Language (DAML) [Hendler and McGuinness, 2001] enables the creation of ontologies in the description of specific web sites. It is not specifically geared to services, but has obvious applications. Some emerging approaches add structure to service descriptions through the use of ontologies, e.g., [Trastour et al., 2001]. Klein & Bernstein develop a richer approach for describing and indexing services based on process models [2001].

Discovering services. Universal Description, Discovery and Integration (UDDI) [UDDI] is an upcoming standard intended for announcing and discovering services. Despite its name, it doesn't offer support for integration. The idea is that services can be listed by their providers at a logically central registry, which can be searched by prospective customer of service to discover a desirable service. Because the indexing of the registry depends on how the services are described, this aspect of discovery is intimately related to service description.

Describing compositions. No good means exist for describing desired compositions of services. Currently, the best descriptions are procedural, typically based on previous work on workflow management. For example, Web Services Flow Language (WSFL) was proposed to describe compositions of services in the form of a workflow [Leymann, 2001]. WSFL specifications tell us how different services ought to be invoked, e.g., in terms of ordering and parallelism among them. DAML-S [Ankolekar et al., 2001] is a web service ontology from the DAML community. DAML-S provides a core set of markup language constructs for describing the properties and capabilities of web services. It also includes a traditional workflow-style process model for compositions of services.

3 Pragmatics

Below we identify some major challenges for semantic web service that are addressed neither by current industry approaches nor by upcoming semantic approaches. Resolving these challenges requires developing an approach that takes some principles seriously. The desired principles are oriented toward the pragmatics of the web.

3.1 Challenges

Now we can consider the challenges for web services when they are geared toward composition.

Service description. An essential assumption of current approaches is that services can be described in a manner that is independent of how they are used. However, services may not be fully described through their methods. Descriptions given by service providers, even if using standardized languages, might be incomplete. This is because of two reasons. One, specialized communities of practice would use services in novel ways. Two, instead of a mere method invocation, the appropriate metaphor is negotiation where the service provider and consumer may reason about whether or not and, if so, how to interact with each other.

This becomes interesting and difficult when semantic exceptions are considered. Semantic exceptions are not just about programming language or operating systems exceptions such as divide-by-zero or file-not-found, but deal with the meaning of the task at hand [Luo et al., 1998]. The challenge is to handle exceptions in service compositions, not simply by throwing a programming language exception, which is not necessarily the correct response. In any case, exceptions suggest a case where the descriptions of services might evolve and might form an input to the questions of compliance and trust mentioned below.

Service discovery and location. Current approaches implicitly assume that a logically central registry of services can make the right matches for all comers. However, in real-life, prospective service consumers would need to discover providers that are trustworthy in a manner that is sensitive to their particular needs. A registry as an implicitly trusted third party seems unrealistic for the following reasons. One, the registry has not interactions of its own to be able to judge the quality or trustworthiness of a service provider or a prospective consumer. Two, what would matter is the mutual trust placed in each other by a provider and consumer contemplating interacting further. Three, the trustworthiness perceived by each in the other would depend upon their specific interactions and contexts, something that a registry would not and should not be expected to model for all participants. A proper solution would b social mechanisms for evaluating and discovering trustworthy parties, both providers and consumers.

Interaction. While current approaches for service invocation represent much progress, they carry the baggage of traditional distributed objects approaches. Current approaches are geared for low-level invocation of services—they are not specially geared for enabling composition. Services are integrated through method invocation without regard to any higher-level constraints. WSDL

allows us to capture the various methods but does not support constraints among those methods. Either too many methods will always be enabled or too few. However, WSDL's functionality is required to specify the methods supported by a service.

Method invocation is appropriate for closed systems, but services are inherently autonomous and often to be used in long-lived interactions. For example, a long-lived interaction occurs in e-commerce when you try to change an order because of some unexpected conditions or try to get a refund for a faulty product. Even short-lived settings involve protocols, e.g., checking if the service requester is authenticated and properly authorized before accepting its order. Consequently, interesting service compositions will often involve subtle constraints on interactions among the participating services. The services won't simply be invoked as methods by an aggregator but will engage in rich protocols for potentially long-lived interactions.

Engineering composition. Current approaches take a procedural view of service composition formulating workflow graphs that can be stepped through. Because of this, the main engineering challenges that they bring up deal with standardizing on the data, e.g., through syntax or the semantics. However, web services have attributes that set them apart from traditional closed applications: they are autonomous, heterogeneous, long-lived, and interact in subtle ways to cooperate or compete. Engineering a composition of such services requires abstractions and tools that bring these essential attributes to the front. When the requirements are expressive, they highlight the potential violations, e.g., the failure modes of a composition.

Engineering composition thus requires capturing patterns of semantic and pragmatic constraints on how services may participate in different compositions. It also requires tools to help reject unsuitable compositions so that only acceptable systems are built.

Compliance and trust. Because traditional approaches take a low-level, procedural stance on service composition, they cannot deal with high-level interactions among services. Consequently, they do not provide a basis for judging whether a particular service that is bound in the workflow will in fact deliver the right interactions. Compliance is a lot more subtle than matching a method signature. It should be rigorous even if context sensitive. Likewise, trust should be based on interactions viewed in a fine-grained manner.

3.2 Principles

While developing community consensus to standardize the semantics for various domains seems like a reasonable approach, we would still like to understand the principles through which web applications and data can be linked. These principles lie not so much in the semantics as in the pragmatics of the web. The shift to pragmatics has the following ramifications, which we formulate as principles for building pragmatically sound web systems.

- *User before provider*. Instead of merely listing the methods or capabilities a service provider might offer, it would help to model the needs of the consumers of the service. These consumers will be presented architecturally as composers of services with potentially quite idiosyncratic needs and composition requirements that would be highly context dependent.

The description, discovery, and invocation of services must be tied to the context of the intended compositions.

- *Process before data*. The context of the usage of data is important and difficult to capture. But understanding the processes behind the data give a better clue to its meaning than a perfectly reusable (and therefore impossible to build) semantics would.

- *Interaction before representation*. Services provide an analog to a functional interface to data. It is "analog" in that, as we argued above, services should go beyond method interfaces. However, they provide a high-level interface that hides the irrelevant details of the semantics of the data, while exposing the components of the semantics that are relevant for a particular purpose. Just as functional interfaces to data structures hide their implementational details, so too will interaction specifications of services hide the "excess" semantics of the data that might otherwise be revealed.

Accommodating the above principles means giving a central position not to the data but to the processes and contexts in which it is used. We can achieve the vision of the semantic web by first concentrating on services rather than data and second by modeling services as *agents* rather than as distributed objects and viewing service providers and consumers as participating in rich multiagent systems.

3.3 Multiagent Systems

Agents are long-lived, persistent computations that can perceive, reason, act, and communicate [Huhns and Singh, 1998a]. Agents act with varying levels of autonomy, depending on environmental constraints and their ongoing interactions, e.g., as reflected in their previous commitments. Because of their autonomy and heterogeneity, services can be naturally associated with agents. Agents make it possible to capture interactions among services and the creation of new services as subtle compositions of others.

Agents are not a panacea. However, applying agents enables us to naturally capture deeper constraints on what services are willing to offer, capturing richer requirements for service composition, discovering trustworthy services, negotiating within teams of providers, a basis for judging the compliance of service providers with their contracts with respect to specific compositions.

4 Conclusions

Please note that our purpose is not to compete with or replace the existing offerings, but to augment and refine them. Ultimately, we will need all the plumbing support and semantic descriptions that existing approaches offer, but also a lot more.

This paper has raised more questions than it has answered. We close by invoking the metaphor of Polya's inventors paradox: "the more ambitious plan may have more chances of success" [1957]. The inventors paradox is the phenomenon in mathematics where a more general claim is easier to prove, for example, when a stronger inductive hypothesis is required to prove the inductive step of

a proof by mathematical induction. In much the same way, we believe, the semantic web will be the easier to realize if we follow it pragmatically, and that would be the obvious approach for any pragmatic computer scientist.

References

Anupriya Ankolekar, Mark Burstein, Jerry R. Hobbs, Ora Lassila, David L. Martin, Sheila A. McIlraith, Srini Narayanan, Massimo Paolucci, Terry Payne, Katia Sycara, and Honglei Zeng. DAML-S: Semantic markup for web services. In *Proceedings of the International Semantic Web Working Symposium (SWWS)*, July 2001.

Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Frystyk Nielsen, Satish Thatte, and Dave Winer. Simple object access protocol (SOAP) 1.1, 2000. www.w3.org/TR/SOAP.

Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana. Web services description language (WSDL) 1.1, 2001. www.w3.org/TR/wsdl.

James Hendler and Deborah L. McGuinness. DARPA agent markup language. *IEEE Intelligent Systems*, 15(6):72–73, 2001.

Michael N. Huhns and Munindar P. Singh. Agents and multiagent systems: Themes, approaches, and challenges. In *Huhns and Singh [1998b]*, chapter 1, pages 1–23. 1998a.

Michael N. Huhns and Munindar P. Singh, editors. *Readings in Agents*. Morgan Kaufmann, San Francisco, 1998b.

Anuj K. Jain, Manuel Aparicio IV, and Munindar P. Singh. Agents for process coherence in virtual enterprises. *Communications of the ACM*, 42(3):62–69, March 1999.

Mark Klein and Abraham Bernstein. Searching for services on the semantic web using process ontologies. In *Proceedings of the International Semantic Web Working Symposium (SWWS)*, July 2001.

Frank Leymann. Web services flow language. TR WSFL 1.0, IBM Software Group, May 2001.

Zongwei Luo, Amit Sheth, John Miller, and Krys Kochut. Defeasible workflow, its computation and exception handling. In *Proceedings of the CSCW Workshop: Towards Adaptive Workflow Systems*, Seattle, 1998. At ccs.mit.edu/klein/cscw98/.

Charles Morris, editor. *Foundations of the Theory of Signs*. University of Chicago Press, Chicago and London, 1938.

George Polya. *How to Solve It: A New Aspect of Mathematical Method*. Princeton Science Library. Princeton University Press, Princeton, NJ, 2nd edition, 1957.

Munindar P. Singh. Agent communication languages: Rethinking the principles. *IEEE Computer*, 31(12):40–47, December 1998.

Munindar P. Singh. The service web. *IEEE Internet Computing*, 4(4):4–5, July 2000. Instance of the column *Being Interactive*.

Munindar P. Singh. Physics of service composition. *IEEE Internet Computing*, 5(3):6–7, June 2001. Instance of the column *Being Interactive*.

Munindar P. Singh, Bin Yu, and Mahadevan Venkatraman. Community-based service location. *Communications of the ACM*, 44(4):49–54, April 2001.

David Trastour, Claudio Bartolini, and Javier Gonzalez-Castillo. A semantic web approach to service description for matchmaking of services. In *Proceedings of the International Semantic Web Working Symposium (SWWS)*, July 2001.

UDDI. UDDI technical white paper, 2000. www.uddi.org/pubs/Iru-UDDI-Technical-White-Paper.pdf.