IEEE Intelligent Systems
www.computer.org/intelligent

Web Services from an Agent Perspective

Terry R. Payne

Vol. 23, No. 2
March/April 2008

Editor: Terry R. Payne
University of Southampton
trp@ecs.soton.ac.uk

Web Services from an Agent Perspective

Terry R. Payne, *University of Southampton*

Editor's Introduction

Intelligent agents are often criticized as representing technology that is actively pursued in research labs but that rarely appears in deployed applications. In fact, many of the underlying technologies of intelligent agents have migrated into mainstream applications, at which point they're no longer referred to as agents. This department will revisit the evolution and application of intelligent agents and consider how they're shaping emergent technologies or becoming embedded within applications. I plan to look at the pros and cons of intelligent agents, relating them to other technologies and exploring successful deployments in the real world. — *Terry. R. Payne*

Multiagent systems evolved from a need for knowledge-aware, distributed, problem-solving mechanisms. These systems are formally grounded using theoretical approaches, including those that assume mentalistic notions (see the sidebar). As a result, much of this research into multiagent systems has provided formal proofs or proof-of-concept demonstrators (such as example systems or prototypes). It has provided only limited, pragmatic support (systems, software, and tools) for the user community.

Research into web services, in contrast, has focused on the user community, resulting in a pragmatic, bottom-up enabling technology that readily facilitates the robust construction of service-oriented systems. Much of the focus of web services research has been on developing declarative descriptions that application developers can share and that their tools can use to construct and develop large-scale distributed software.

Despite these differing approaches, the inherent component-based structure underlying both agents and web services raises questions about how exactly they differ and whether they can coexist.[1]

How agents differ from web services

The concept of an agent is integral to both the Semantic Web and web services.[2] According to the W3C *Web Services Architecture* note,

> A Web Service is an abstract notion that must be implemented by a concrete agent. The agent is the concrete piece of software or hardware that sends and receives messages, while the service is the resource characterized by the abstract set of functionality that is provided.[3]

So, we can think of a web service as an abstract notion or task that a variety of providers can instantiate and offer.

The note later argues that an agent is a computational resource that can act as a proxy for those entities (human, organizational) that own the service. However, the note ascribes no further mental notions to agents other than that they can exchange messages. So, rather than compare a web service to an agent, maybe it's better to compare it to an agent's functionality or capability in a multiagent system. Unfortunately, the analogy isn't that simple. Web services have traditionally been transient and stateless processes that exist only during service execution. In addition, these services are instantiated to perform a specific task (thus facilitating scalable, concurrent service provision, similar to the provisioning of web pages from a web server). An agent, however, is often persistent and resource bound, providing only a single service to its peers at any given time.

In "Brain Meets Brawn: Why Grid and Agents Need Each Other," Ian Foster, Nicholas Jennings, and Carl Kesselman propose a clearer separation between the notion of an agent and a web (or Grid) service.[4] They argue that the evolution of Grid services focused on the pragmatic development of technologies, standards, and engines that can realize distributed, usable service environments. To deploy reliable, distributed, and ubiquitous platforms that support Grid computing (and likewise web services), the Grid community emphasized the agreement and adoption of standards and policies, and, more recently, has emphasized shared ontologies. This contrasts with multiagent-systems research, which has focused on developing principled, formal mechanisms for distributed problem solving at the knowledge level[5] in terms of what tasks or goals the multiagent community should tackle and which agents should solve the tasks.

While web services research has focused on developing standards for well-defined and declarative interfaces, workflows, and protocols, there's been little focus on the mechanisms that help the service perform the task. Although the multiagent-system community has developed theories about each communication act's intention, it has placed less emphasis on defining well-defined

1541-1672/08/$25.00 © 2008 IEEE
Published by the IEEE Computer Society

Agency and Mentalistic Notions

The notion of agency originally emerged from the field of artificial intelligence—specifically, to help support and coordinate distributed AI problems. Although you might consider many of the characteristics discussed in the main article to be weak notions of agency,[1] stronger, or mentalistic, notions often ascribed to humans have been used to characterize an agent. Such notions reflect those used in human cognition or communication,[2] such as agents having beliefs about the world or certain desires or aims, or agents performing intended actions to progress toward a goal. Other notions, such as obligations—formed through communication with other agents or in response to societal norms—might also affect an agent's actions. These notions might sound somewhat anthropomorphic, but they represent programming abstractions intended to facilitate a more intelligent and deliberative approach to decision making.

References

1. M. Wooldridge and N.R. Jennings, "Intelligent Agents: Theory and Practice," *The Knowledge Eng. Rev.*, vol. 10, no. 2, 1995, pp. 115–152.
2. Y. Shoham, "Agent-Oriented Programming," *Artificial Intelligence*, vol. 60, no. 1, 1993, pp. 51–92.

machinery to pragmatically support agent communication.

Differentiating between agents and web services is thus problematic, because you could argue that you can implement agents using web service technology or build adaptive, intelligent mechanisms into a web service's design. Researchers have proposed many definitions for agents; unfortunately, there's always some example that, despite strictly satisfying the definition, isn't an agent "in spirit." Here, I identify some of the fundamental differences between agents and web services—thus offering some insight into the synergies of using both—by discussing Jennings' five characteristics of an agent.[5]

Agents are problem solvers

Agents are clearly identifiable problem-solving entities with well-defined boundaries and interfaces.

The approaches used to engineer agents and web services are fundamentally different. Typically, when designing web services, engineers define the goals as clearly articulated workflows and formally validated protocols, grounded using well-formed calculi and logical formalisms. Web service architects have invested significant effort into defining data types and data structures, creating mechanisms for routing, securing and addressing messages, and developing tools for constructing, advertising, discovering, and subsequently using services.

In contrast, the agent view assumes that agents can employ a variety of interaction methods—from simple, client-server interactions to rich social interactions—to flexibly achieve their goals. Such interactions generally occur through knowledge-level messages in a declarative language such as the Knowledge Query Manipulation Language or the Foundation for Intelligent Physical Agents' Agent Communication Language. Thus, there is an emphasis on reasoning about received messages, and other (partial) knowledge gleaned from the environment, to determine what actions to take.

Likewise, knowledge about available peers and their capabilities and motivations is essential in dynamically determining how to solve problems at runtime. So, instead of simply performing a task, an agent can decompose a problem into its constituent tasks and elect whether to perform or delegate tasks or coordinate with other available agents to solve the overall problem or goal. This decomposition is done dynamically, rather than being prescribed, and thus can better adapt to changing contexts and environments.

They're proactive

Agents are capable of exhibiting flexible problem-solving behavior in pursuit of their design objectives—being both reactive (able to respond in a timely fashion to changes that occur in their environment) and proactive (able to opportunistically adopt goals and take the initiative).

Agents are inherently communicative and socially aware. They respond to both changes in their environment and messages from peers as a result of internally scheduled tasks. Such triggers can motivate their intention to achieve some goal, resulting in proactive behavior as necessary.

Web services, however, are typically transient processes whose instantiation and existence is triggered when the web server receives a message. An advantage of this "factory-based" instantiation of processes for each service invocation means that providers can offer potentially huge numbers of concurrent service instances in response to simultaneous service requests. Although a web service might initiate communication with another web service when executing its task, this is still reactive because it's part of the prescribed actions triggered by the original instantiating message. Although you could build proactivity into a web service, this would ultimately introduce many notions of agency into the web service design.

They're goal oriented

Agents are designed to fulfill a specific role—they have particular objectives to achieve.

Typically, a web service exists to perform a specific task, such as offering value-added functionality to support B2C or providing e-business functionality to third parties. Companies such as eBay and Amazon.com offer access to their core technologies through web services, either to facilitate third-party trading or to offer access to their resources.

Agent behavior is motivated by more abstract, mentalistic notions, such as knowledge, intention, belief, and obligation. Typically, an agent is designed to maximize some utility through rational behavior. So, when electing to perform a task, an agent can attempt to determine the utility gain in performing this action, on the basis of a possible reward or some perceived advantage (taking into account any costs incurred). If an agent doesn't perceive some gain, it might not perform the task, whereas a web service receiving the equivalent request will generally perform the task.

They're context aware

Agents are situated (embedded) in a

particular environment over which they have partial control and observability—they receive inputs related to the state of their environment through sensors and they act on the environment through effectors.

This characteristic is often ascribed to hardware agents (such as robots) but can equally apply to software agents. However, such sensors can provide only partial knowledge of the environment. Furthermore, in dynamic environments with numerous agents, this knowledge can become stale.

Agents also have only partial control of their environment, so they need to be able to assess their context (so that they can react accordingly). This often necessitates collaboration between peers to achieve desired changes (or acquire desired knowledge) beyond their sphere of influence. Agents that are aware of their environment also have knowledge of new agents, which they can use to solve future problems. However, the ability to observe and interrogate peers can yield a more sophisticated environmental model, which questions whether agents can be trusted to achieve a task or whether they have a reputation of cheating or defaulting on contracts.

Web services are similarly limited with respect to their scope of observable facts and to the actions they can perform to manipulate and affect the environment. However, because web services are typically reactive, the knowledge they process is typically only what the developer considered necessary at design time. This eliminates the possibility of exploiting opportunistic knowledge.

They're autonomous

Agents are autonomous—they have control both over their internal state and over their own behavior.

Autonomy is a defining agent characteristic and a consequence of the characteristics previously listed. For example, by defining a desired, overall behavior as a utility maximization function, agents can acquire information about their environment and either proactively perform tasks or collaborate with others on (joint) tasks. Thus, they can dynamically respond and adapt to a changing environment.

Agents can also be self-aware. By acquiring and retaining knowledge over time, they can learn about alternate strategies and solutions to problems that yield more optimal solutions (at least as far as the agent is concerned). An agent can evolve its own behaviors without direction from its owner or user.

Web services are rarely autonomous, unless the notion of autonomy is included in the service design, which typically involves constructing stateful and persistent services. Researchers are beginning to explore these notions of autonomy and autonomic behavior for web services, but so far they have primarily used notions of agency to achieve autonomy.[6]

Blurring these differences

Web service technology primarily provides a distributed-object definition and invocation framework that lets developers publish and access the code enclosed in a web service container. This code could employ the notions of agency to solve simple tasks or provide component functionality to support complex e-business machinery. By providing persistence, autonomy, and identity to web services, the distinction between agents and web services becomes increasingly blurred.

Yet the web services community hasn't paid much attention to the notion of autonomy. Many systems assume prior knowledge of the resources found in the environment or consider the environment from a single consumer's viewpoint. This assumption emerges from research focusing on solving a specific problem in a controlled scenario, without considering the full implications of resources existing in complex, evolving environments where there might be competition for resources. The "factory" mechanisms that web servers use to create service instances somewhat alleviate the problem of concurrent access to services by creating new service instances on demand.

However, such techniques aren't feasible in resource-bound environments, where the available processing power is limited or where services support physical equipment. Conflicts can occur when demand exceeds supply or when multiple parties generate and enact workflows without forming a commitment or contract. This can lead to a failure of services (owing to prior provisioning by another consumer) or a delay in service execution. So, autonomous mechanisms must support collaboration or cooperation or must refine service planning or provisioning at runtime (rather than assume human involvement at design time).

The Internet's size and diversity provide a rich, valuable, and dynamic knowledge source for both agents and web services to exploit. However, this diversity and heterogeneity keeps such components from possessing prior, up-to-date knowledge about the availability of services and of information sources. It also keeps them from sharing reliable data models outside highly restrictive bounds. Many approaches' implicit assumption of a closed-world environment renders the Web effectively incomprehensible to all but the most carefully crafted, highly specialized, and diligently managed applications. Efforts such as the Semantic Web are beginning to address this limitation by exploiting and adopting logical mechanisms and knowledge-engineering theory to facilitate machine-processible articulation (and inference) of data as usable, accessible knowledge.

References

1. M. Huhns, "Agents as Web Services," *IEEE Internet Computing*, vol. 6, no. 4, 2002, pp. 93–95.
2. J. Hendler, "Agents and the Semantic Web," *IEEE Intelligent Systems*, vol. 16, no. 2, 2001, pp. 30–37.
3. *Web Services Architecture* note, W3C, 11 Feb. 2004, www.w3.org/TR/ws-arch.
4. I. Foster, N. Jennings, and C. Kesselman, "Brain Meets Brawn: Why Grid and Agents Need Each Other," *Proc. 3rd Joint Conf. Autonomous Agents and Multi-Agent Systems* (AAMAS 04), ACM Press, 2004, pp. 8–15.
5. N.R. Jennings, "An Agent-Based Approach for Building Complex Software Systems," *Comm. ACM*, vol. 44, no. 4, 2001, pp. 35–41.
6. E.M. Maximilien and M.P. Singh, "Toward Autonomic Web Services Trust and Selection," *Proc. 2nd Int'l Conf. Service Oriented Computing* (ICSOC 04), ACM Press, 2004, pp. 212–221.
7. M. Wooldridge, *An Introduction to Multiagent Systems*, John Wiley & Sons, 2002.

Terry R. Payne is a lecturer in the University of Southampton's School of Electronics and Computer Science. Contact him at trp@ecs.soton.ac.uk.