



Agents and Web Services

Presented By

Ming He

Instructor Dr. Behrouz H. Far

Background

Distributed computing

Let's think a little bit on how distributed computing technology has evolved.

In the beginning, things were built and deployed typically in the form of client and server model in which clients talk to a single server, for example, doing remote procedure calls.

The second phase can be called web-based computing in which many clients talk to many servers through the net. In this phase, communicating partners still have to go through some pre-arrangement in terms of what common object model they have to use or what common communication protocol they have to agree upon.

Finally, the web services model. Service users and service providers are dynamically connected. And the pretty much every computing device and application participate as both service user and service provider.

Web Services

If you believe the hype inundating the IT industry, every man, woman, and child across the globe is doing heads-down development of Web services. The first complication you often encounter when investigating Web services projects is that different people have different interpretations of what 'Web services' are. For example, some people see Web services as simply the XML-enablement of any application. In their eyes, just slap some XML imports and exports in front of a legacy system and then you have Web services.

The common definition of Web services in the industry – and the one in this paper here – is that for an application to qualify as a Web service component, it must use one or more of the following : Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and Universal Description, Discovery, and Integration (UDDI) [2]. The use of XML is implicit in all of these standards [6].

The grand scheme for Web services is to provide a framework for deploying both applications within an enterprise (intra-enterprise) and applications that span enterprises (inter-enterprise) [1]. The deployed application may be constructed of new software component written to Web services standards as well as legacy software components that have been adapted for Web services.

In the vision of Web services, application components can interconnect regardless of the language they were written in or the platform they run on [1]. However, in reality, Web services are implemented in conjunction with specific platforms, application servers, languages, and even database products. For example, Microsoft's .NET is rooted in the Windows platform, while J2EE trends toward Unix (in all fairness, J2EE can run on Windows and on other platforms, but the majority of implementations are under Unix or Linux).

Impact of Web Services on Software: “Application Dis-Integration”

A web service is some business functionality exposed over the internet. What it entails is that today's monolithic applications will be disintegrated into several forms of web services which exposed as

services over the internet. We believe web services can be categorized into application and system service web services. Examples of Application Services are a commerce shopping cart or a calendar appointment service. System services are expected to provide common services that would be used by application servers and the examples of System Services are identity, policy/authorization, directory service, notification, logging, file storage service, and so on.

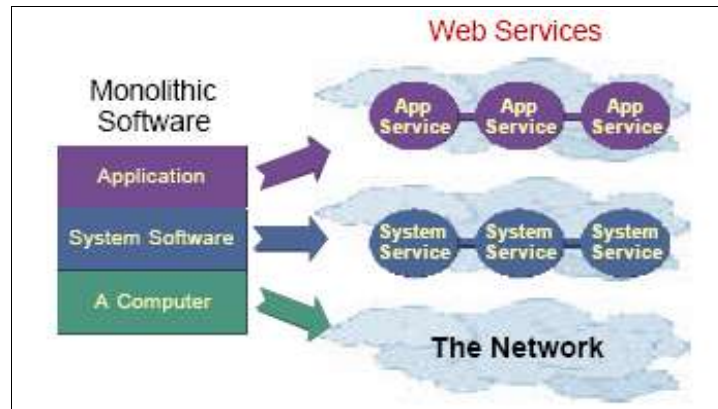


Illustration 1 Impact of Web Services on Software

Impact of Web Services on Integration: Trigger the Network Effect

Another impact of web services is that they will trigger the Network Effect for integration technology [1]. Metcalfe's Law describes the an effect that is often illustrated with an example of FAX machines. The first FAX machine had zero value because it could communicate with no one. When a second came on line, the value increased. And as the network reached a critical mass, it compelled more and more users to get FAX machines. This is also called the Network Effect.

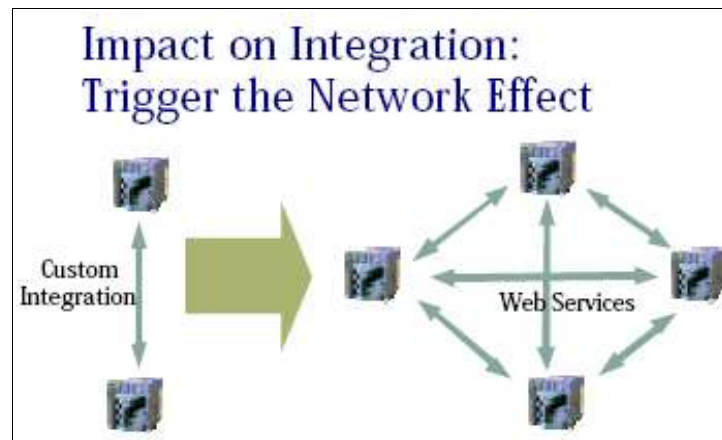


Illustration 2 Impact on Intergration

Agents

By an *agent-based system*, we mean one in which the key abstraction used is that of an *agent*. Agent-based systems may contain a single agent, (as in the case of user interface agents or software secretaries, but arguably the greatest potential lies in the application of *multiagent* systems. By an *agent*, we mean a system that enjoys the following properties:

Autonomy: agents encapsulate some state (that is not accessible to other agents), and make decisions about what to do based on this state, without the direct intervention of humans or others [7];

Reactivity: agents are *situated* in an environment, (which may be the physical world, a user via a graphical user interface, a collection of other agents, the INTERNET, or perhaps many of these combined), are able to *perceive* this environment (through the use of potentially imperfect sensors), and are able to respond in a timely fashion to changes that occur in it [7];

Pro-activeness: agents do not simply act in response to their environment, they are able to exhibit goal-directed behavior by *taking the initiative* [5] [7];

Social ability: agents interact with other agents (and possibly humans) via some kind of *agent-communication language*, and typically have the ability to engage in social activities (such as cooperative problem solving or negotiation) in order to achieve their goals [4] [7].

Web Services and Agents

Agents and Roles

The provider entity and the requester entity exchange messages, agree on both the semantics and the mechanics of the message exchange [3]. The mechanics of the message exchange are documented in a Web service description (WSD), a machine-processable specification of the Web service interface representing a “contract” governing the mechanics of interacting with a particular service. The *semantics* of the message exchange represents the "contract" between the requester entity and the provider entity regarding the *purpose and consequences of the interaction*. In addition, *Humans* still play a role to *agree* on the semantics and the service description and *create* the requester and provider agents.

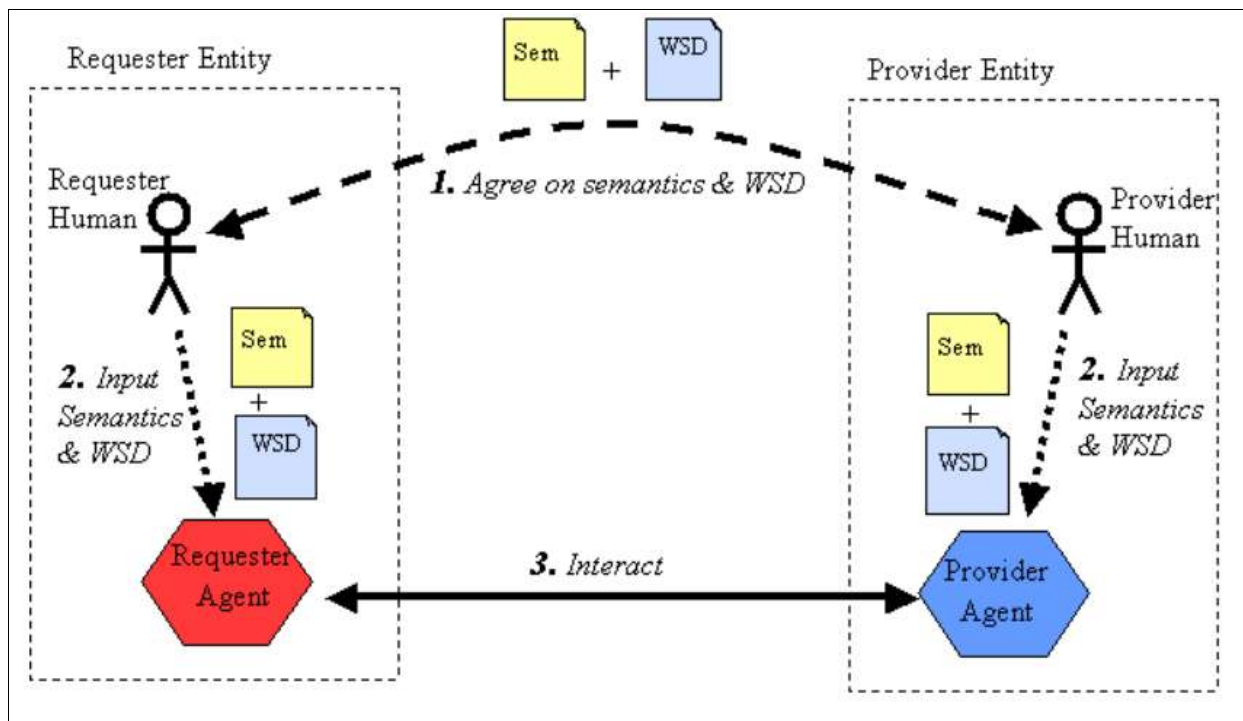


Illustration 3 Web Service Agents

Use Case: A Vacation Booking System

Let's use a vacation booking system as an example to illustrate the agents in the Web service.

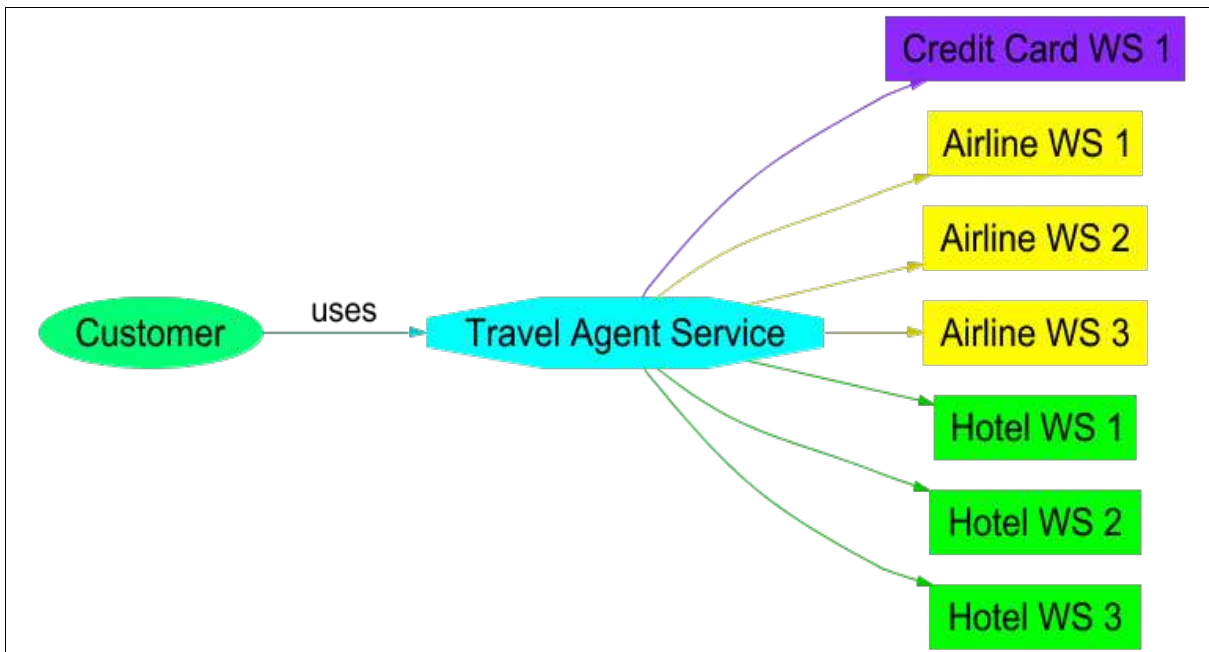


Illustration 4 Use Case: Vacation Booking System

The travel agent service should understand the airline services and should be able to send meaningful information to them, a *travel industry ontology* needs to exist and be used by the Web services taking part in this scenario .

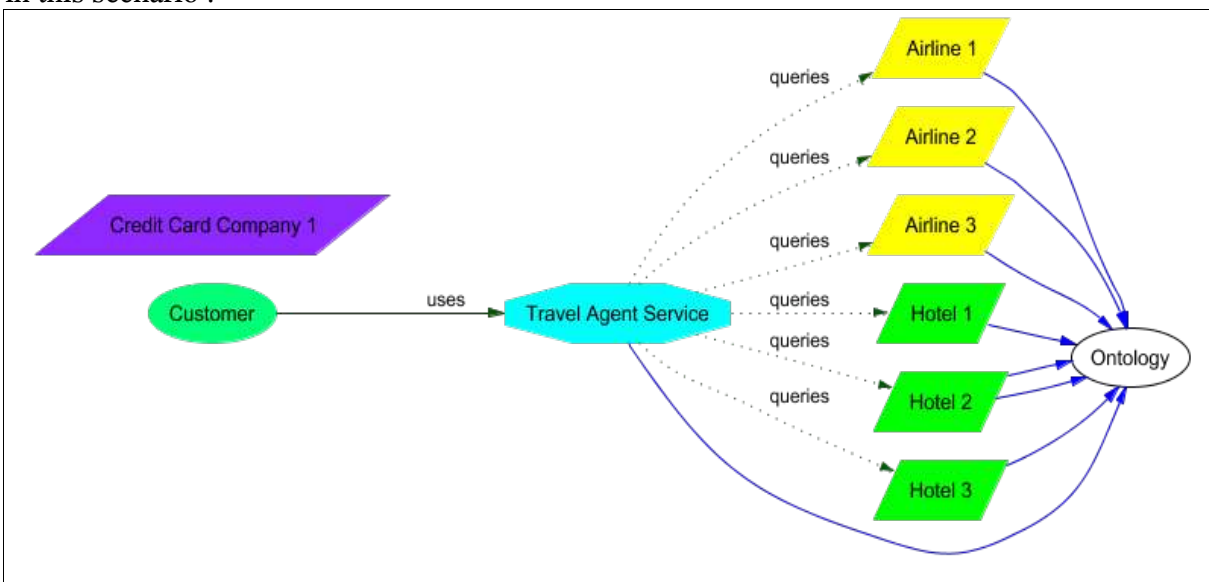


Illustration 5 Travel Industry Ontology

The user has been presented with options for hotels to go to his/her destination and a means of payment. The user chooses a hotel option. The travel agent service contacts a payment service for

payment authorization. The service books the hotel and confirms the flight, using the payment authorization from the payment service (i.e. a credit card company).

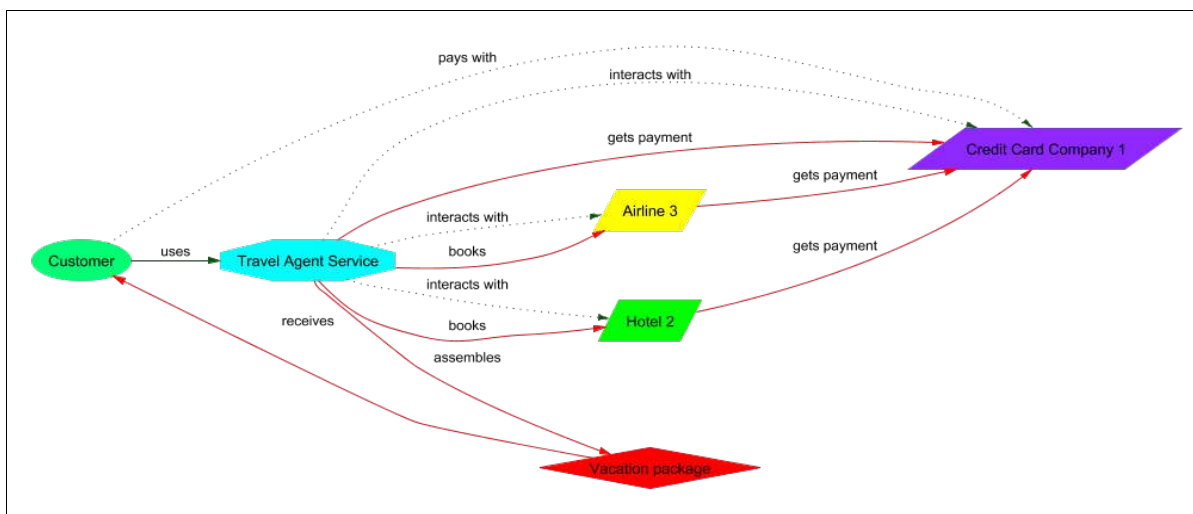


Illustration 6 Book Hotel and Flight

Benefit of Implementing Software Agents for The Web Services

The Web services involve many different hardware/software platforms, applications, and databases. This attribute of Web services makes themselves the best candidate of agents technology. Because agents are practicable for the large scale, complex, heterogeneous, adaptable, and distributed systems that are running in an quite open environment.

Web Services profit from dynamic creation of agents, semantic level of communication, mediation and reasoning capabilities, and flexibility and robustness.

Agents profit from standard description techniques for procedures, ontology descriptions, and connection to web.

In the example of booking vacation, agents are able to bring web services into play at run-time where each component or application becomes a commodity. You will be able to pick and choose from a shopping list of fixed price services as and when you need them. Agents will be able to help make decisions as to which services to use at run-time. They will be able to help make decisions based on various criteria such as price, availability, volume etc. Interestingly, fixed prices are inefficient for both buyers and sellers. They exist only because humans don't want to spend lots of time negotiating. Agents don't have this problem. Even when it comes to small amounts or what we might consider details, agents still have the capacity to negotiate. In an agent oriented world, dynamic pricing will probably become the norm and fixed pricing the exception.

Conclusions

The combination of agents and Web services is a good approach in the Web world. Application developers assembling network/distributed applications using web services have to perform two

distinct sets of activities. The first is to “wire-up” web services such that they can interact with each other. The second is to execute business processes that users want [9]. As web services become more pervasive, the number of agents that collect data on behalf of clients will continue to grow [8]. Once you decide to empower the web service with the responsibility to make decisions based on real-time information, agents are a natural way to achieve this. Add to this the need for a web service to make intelligent decisions about how to protect itself from malicious use and some would argue that the distinction between agents and web services becomes very blurred [6].

References

- [1] Web Service Programming using XML and Java™ Technology, Sun Microsystems Corp., 09/21/2003
- [2] Web Services and Web Service Agents, January 2003, An IPNet Solutions White Paper
- [3] Web Services: agents and roles, W3C slides (<http://www.w3c.it/talks/webday2003-interoperability/slide40-0.htm>)
- [4] Web Services and Software Agents, F. Dignum, Utrecht University
- [5] FIPA - Agents Meet the Semantic Web? Heimo Laamanen, Principal Researcher, Sonera Corporation
- [6] Agents and web services, Ken Woghiren, Lost Wax
- [7] Agent-Based Software Engineering, Michael Wooldridge, Mitsubishi Electric Digital Library Group, 18th Floor, Centre Point, 103 New Oxford Street, London WC1A 1EB, United Kingdom, September 19, 1997
- [8] P3P Position Paper: Agents and P3P, W3C Workshop on the Future of P3P, November 12-13, 2002, Jack Humphrey, Development Manager, Data Acquisition, Coremetrics, Inc.
- [9] Two Level Architecture for Web Service Interactions Position Paper to Workshop on Web Services Sankar Virdhagriswaran, Dr. Gordon Dakin, Dan Leliberte, Crystaliz, Inc.