Integration of Software Agent Technologies and Web Services

Mohammed Ketel
School of Information Technology
University of Baltimore
Baltimore, MD 21201
mketel@ubalt.edu

ABSTRACT

Web Services technology enables the automation of service discovery, invocation, and composition. On the other hand, Software Agents provide a distinctive capability in mediating user goals to determine service invocations. Software Agents are autonomous entities that can discover, invoke, compose, and monitor services without user's intervention. Moreover, agents possess the ability to handle the dynamism of the Web Services environments. Web Services and agent technologies have different problems that limit their functionality when applied separately. The major reason is that agents are not compatible with the widely accepted standards of Web Services. This paper presents a framework that provides an integration of Web Services and Software Agents technologies by making use of a middleware to facilitate their interoperation.

Keywords: Web Services, SOA, Software Agents, Gateway.

1. INTRODUCTION

The Web service paradigm [1] provides the feature richness, flexibility and scalability needed by enterprises to manage the Service-Oriented Architecture (SOA) challenges [12, 16]. One of the essential characteristics of SOA is the idea of loose coupling between services and clients. Loose coupling requires that the service have a well defined interface which is separate from the implementation [2]. In addition, loosely coupled services provide flexibility and scalability necessary for extended application life and reduced maintenance costs. Thus Web services enable improved coordination amongst multiple computing platforms, applications, and Business partners [2, 3].

Today's complex systems can be addressed by exploiting software agents. Software agents represent a useful paradigm in the development of complex distributed systems. The feature of sociability of agents enables building systems composed of several agents, where they interact to achieve a common goal (cooperative agents) [4].

The W3C Web Services Architecture specification defines software agents as, the running programs that drive web services, both to implement them and to access them as computational resources that act on behalf of a person or organization [5]. This definition of an agent identifies one of the primary motivations for implementing Multi Agent Systems (MAS). Agents are primarily responsible for mediating between users' goals, and the available strategies and plans [6].

Although web services and software agents both provide a means for encapsulating business or application knowledge, they differ. Agents offer multiple services that can be processed concurrently and activated according to specified goals [7]. Unlike Web services which provide functionality through simple executable methods, agents that act intelligently use knowledge to react to and act on their environment autonomously and proactively.

This paper presents the integration of software agents and Web services technologies. Combining both technologies provide ease of use and reliability for any user. Some challenges are involved in this integration. Both technologies use different service registries, service description languages and communication protocols. References [8, 9] proposed a Gateway middleware that provides appropriate transformation mechanisms without disturbing the existing specifications of both technologies. The importance of this approach is that it enables integration of Software Agents and Web services without changing their existing specifications at the cost of time taken for translations which is negligible as compared to a transaction.

2. WEB SERVICE PARADIGM

Web Services technologies have been endorsed by many companies as a strategic direction in line with the general IT Industry acceptance of service-oriented architecture (SOA). Vendors of all the major application servers, EAI software, packaged applications and development environments have provided basic integrated support for Web Services [10].
Web Services are encapsulated, self-descriptive, modular, internet applications that may be accessible by the users via the network. Their basic purpose is to enable standardized, uniform access to heterogeneous, distributed software, running on different software/hardware platforms.

A service-oriented architecture (SOA) is a contractual architecture to offer and consume software as services. There are three entities that make up SOA [2, 12]: (1) service providers, (2) service requestors (also known as service consumers) and (3) service broker (registry).
- Service providers are the owners that offer services. They define descriptions of their services and publish them in the service registry.
- Service requestors use a find operation to locate services of interest. The registry returns the description of each relevant service. The requestor uses this description to invoke the corresponding service.
– Service registry is a searchable registry providing service descriptions. It implements a set of mechanisms to facilitate service providers to publish their service descriptions. Meanwhile, it also enables service clients to locate services and get the binding information.
Interactions with a Web service take place in three modes [12]:
– Service publication is to make the service description available in the registry so that the service client can find it.
– Service lookup is to query the registry for a certain type of service and then retrieve the service description.
– Service binding is to locate, contact, and invoke the service based on the binding information in the service description.

Currently, Web services technology implements SOA by means of standard XML-based initiatives. Three initiatives are used in order to support interactions among Web services:
- Simple Object Access Protocol (SOAP), which enables communication among Web services,
- Universal Description, Discovery and Integration (UDDI), which is a registry of Web services descriptions,
- Web services Description Language (WSDL), which provides a formal, computer-readable description of Web services. Figure 1 illustrates the SOA paradigm graphically.

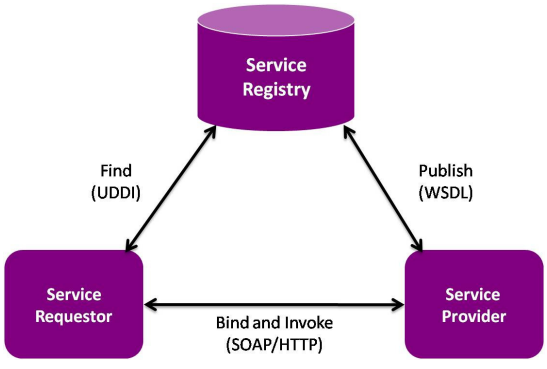

Figure 1: Standards to Define, Publish and Use Web Services

3. SOFTWARE AGENT TECHNOLOGIES

Like Web services, software agent technologies are another important enabler of dynamic product development. Agent-based systems are able to solve problems that are too large for a single-resource limited system; provide solutions where the information is distributed; and enhance system speed, reliability, and extensibility. There are a lot of definitions for software agents but none universally accepted so far. In general, a software agent can be defined as a computational entity, which acts on behalf of others (humans or agents). From different definitions it is possible to summarize a list of attributes common to software agents. Typical attributes of a software agent are [11]:
- Autonomous: Agents control both their individual state and behavior.
- Reactive: Agents are able to perceive changes in their environment and respond in a timely manner.
- Proactive/Goal-oriented: Agents demonstrate goal-oriented behavior by taking initiative to meet objectives.
- Social Ability/Communication: Agents interact with other agents (and possibly humans) via some kind of agent communication language.
- Cooperation: An agent realizes that some goals can only be achieved by cooperating with others.
- Learning/Adaptivity: Agents' ability to learn from history and to adapt to changes means flexibility and improved performance over time.
- Rationality: Is the assumption that an agent will act in order to achieve its goals.
- Mobility: It is sometimes desirable for an agent to change its physical position in the network. Such an agent can optimize its communication with another agent by migrating to the vicinity of that other agent, which might reduce costs and speed up interaction.

Software agents are best suited for applications that are modular, decentralized, changeable, ill-structured and complex. It is the application and goal in question that determine which ones of these attributes dominate in each case. In a typical agent several attributes together make up the behavior of an agent.

Software agents need organizational structures that constitute the multi-agent system [13]. They define the comprising agents and the communication channels. This requires services for agent registration and deregistration, a defined address space, and location services (see section 4 for details).

4. AGENT PLATFORM

The Agent Platform is depicted in Figure 2.

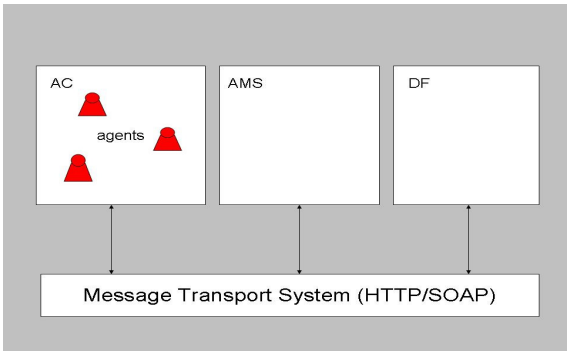

Figure 2: Agent Platform

4.1. Agent Platform Structure

As proposed by FIPA Agent Management Specification [13], the agent platform consists of the following entities:
(1) The AMS Service: the AMS (Agent Management System) represents the authority entity in the platform. The AMS component has the ability to perform various actions on the agents' life-cycle (e.g. create or kill agents) and also provides the naming service. Each agent platform necessarily contains one single instance of the AMS component.
(2) The DF Service: the DF (Directory Facilitator) provides a yellow pages directory service to agents. Any agent can publish its functionality and its services offered by registering to the DF, while an agent that needs to use a specific service can query the DF in order to discover an agent that provides this service.
(3) The AC Service: the AC (Agent-Container) component is the actual run-time environment for an agent. The Agent-Container provides functionality regarding agents' life-cycle management, which can be used from the AMS to manage the agent system.
(4) The Message Transport System: it constitutes the communication bus of the platform. The communication between agents is achieved by exchanging ACL (Agent Communication Language) messages. This kind of communication is obviously a rather abstract way of communication and the actual exchange of messages has to be made via means of a concrete communication protocol. The message transport system of the platform makes use of the SOAP [14] protocol; each ACL message is encapsulated into a SOAP message and sent over HTTP.

All platform components are implemented as stateful web services conforming to WSRF standards. Their functionality is exposed by standard WSDL interfaces and the invocation of the exposed operations is made by standard SOAP requests and responses. Additionally, all components publish resource properties that are representative of their state. The AMS Service publishes two kinds of resources: (a) the platform name and (b) the structure of the platform, which is a list of the URIs of the instances of the Agent-Container Service that have registered with the specific AMS, as well as the identifiers of the agents that reside in each instance. The resource properties of the Agent-Container Service are (a) the name of the Agent-Container and (b) a list of the agent identifiers that currently reside in the Agent-Container. Finally, the DF Service exposes as a single resource property the list of agents that have been registered with it and a description of the services they provide.

4.2. Gateways between Agents and Web Services

There is a need of a middleware for required integration of Software Agents and Web Services. The technological challenge of combining conventional agents and web services with gateways has been studied in [8]. Both technologies have different specifications as follows [8]:
(1) Agents and Web Services use different communication protocol, i.e. Agents use Agent Communication Language (ACL) whereas Web Services use Simple Object Access Protocol (SOAP).
(2) Agents and Web Services use different service description languages, i.e. Agents use ontology named Directory Facilitator Agent Description (DF-Agent-Description) whereas Web Services use Web Services Description Languages (WSDL).
(3) Agents and Web Services use different service registries, i.e. Agents have Directory Facilitator (DF) based on FIPA specifications, whereas Web Services use Universal Description Discovery and Integration (UDDI) which is based on W3C specifications.

A situation in the middle happens when there is a gateway that permits bidirectional interaction between agents and web services [8]. In this context, a gateway is a software program that acts as mediator between two software systems that use two different technologies, WSDL+SOAP+UDDI and FIPA in this case. With gateways, software agents and web services can remain as independent elements and use each other transparently at the cost of time taken for translations which is negligible as compared to a transaction.

The Gateway [7, 8, 9] is shown in Figure 3 (see [8] for details):
(1) The Service Discovery converter enables agents and Web services to search for one another. Software agents can discover Web services via UDDI registries and conversely, Web service clients can perform searches for agents and agent services from agent registries such as the Agent Platform' DF.
(2) The Service Description converter enables service publishing among Software Agents and Web services.

Software Agents can publish services in Web Services registries such as UDDI and Web Services can be published in Multi Agent Systems service registries such as the Agent Platform DF.

(3) The Communication Protocol converter component enables service invocation among software agents and Web services. Software agents can invoke Web services and Web service clients can invoke software agents in Multi Agent Systems. Specifically, the SOAP to ACL converter' enables Web service clients to invoke Software Agents, and the ACL to SOAP converter enables Software Agents to invoke Web services.

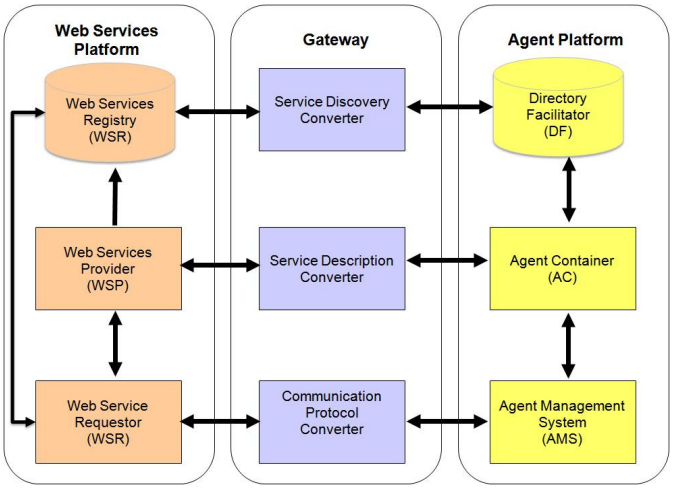

Figure 3: Middleware Conversion Service

In many situations, resources are aggregated into virtual organizations (e.g., a site, or one or more companies that collaborate and share resources). By placing the gateway function in a separate middleware service, the gateway is able to monitor resource usage on a virtual organization-scale rather than on a consumer basis

5. AGENT IMPLEMENTATIONS OF WEB SERVICES

Like web services, agent technologies are another important enabler of dynamic product development. Agent-based systems are able to solve problems that are large for a single-resource limited system; facilitate the interconnecting and interoperation of multiple existing legacy systems; provide solutions where the expertise and information is distributed; and enhance system speed, reliability, and extensibility.

Agents extend Web services in several important ways [15]:

- A Web service knows only about itself but not about its users/clients/customers. Agents are often self-aware and can gain awareness of other agents and their capabilities as interactions among the agents occur.

-Web services, unlike agents, are not designed to use and reconcile ontologies. If the client and provider of the service happen to use different ontologies, then the result of invoking the Web service would be incomprehensible to the client.

-Agents are inherently communicative, whereas Web services are passive until invoked.

- A Web service is not autonomous. Autonomy is a characteristic of agents.

- Agents are cooperative and, by forming teams and coalitions, can provide higher-level and more comprehensive services.

The architecture presented here is divided into three layers: user application layer, service coordination layer (middle agent layer), and Web Services layer, as depicted in Figure 4. The user application layer is responsible for organizing agents to actually perform useful activities for users. Agent-based middle layer is required for scalable, intelligent, dynamic service composition. Agents make use of the semantic annotation of services capabilities to automatically discover, compose, invoke and monitor Web services.

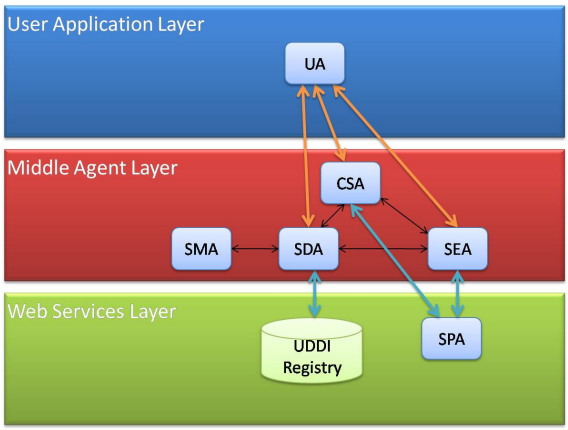

Figure 4: Agents in a Cooperative System

To support the architecture in which heterogeneous components can interoperate and appear homogeneous, a variety of agents are needed [15, 17, 18]. Agents are grouped in two main categories [17]: agents that act on behalf of service owners and agents that act on behalf of service consumers. Those acting on behalf of service owners manage the access to services. On the other side, the agents that act on behalf of service consumers have to locate services, and receive and present results. Different agents are needed for each of the different components:

- User (Requestor) Agents (UA): run on the top of users' devices whether fixed or mobile. They help the user formulate and customize his information requests. They also plan appropriate interactions with other agents (in the lower layer) on the user's behalf.

- Service Discovery Agent (SDA): it is in charge of searching in the Semantic Web Services repository for the

service or set of services (i.e. composition) that satisfy the requisites established by the users.
- Composite Service Agents (CSA): their role is to trigger the specification of the composite services and monitor their deployment. A composite-service-agent ensures that the appropriate component services are involved and collaborating according to a specification.
- Broker Agent (BA): it is responsible for solving the interoperability issues. Brokers might also function as communication aides by managing communications among various agents and application programs in an environment.
- Execution/Mediator agents (SEA): supervise query execution, monitor and execute workflows. A mediator agent is a specialized execution agent. Mediators work with brokers to determine which resources might have relevant information. They also decompose queries to be handled by multiple agents, combine the partial responses obtained from multiple resources, and translate between ontologies.
- Service Matchmaking Agents (SMA): Agent-based matchmaking approach works by accepting requests from requestors/ agents and returning a set of services that matches these requests with additional information as the degree of match for each service.
- Service Provider Agents (SPA): it acts as a service provider representative. The entities set their preferences regarding service execution and these are taken into account during the negotiation process with the service consumers.

In general, software agents and Web services are two independent computational concepts. Software agents are autonomous, cooperative, aware, and embody diverse knowledge and reasoning approaches. These characteristics make agent oriented systems an ideal mechanism for handling the today's ever growing distributed environment [16]. This diversity is sometimes essential in many applications. However, problems arise through unnecessary heterogeneity in agent construction. A practical way is to apply agents in the conventional roles outlined above [16]. It will be easier to keep the agents conceptually separate from each other (could upgrade each agent independently). On the other hand, Web services are good at integrating and managing Internet-based enterprise interoperation, and wrapping application logic. However they are weak in supporting the quality of service such as autonomy and flexibility. So, the capabilities of agents and Web services are complementary.

6. AGENT COOPERATION

Interoperability could be achieved by the cooperation of the software agents at different layers as depicted in Figure 4. The semantic service discovery functionality is realized by two different types of agents: Service Discovery Agents and Service Matchmaking Agents. This was done for reasons of efficiency and flexibility, as in some application domains the matchmaking functionality may not be necessary. Similarly, the service coordination functionality is realized by Service Composition Agents (SCA) and Service Execution Agents (SEA). Whenever a User Agent needs the provision of a service, it asks the Service Discovery Agent (SDA) for service providers that match its request. The SDA accesses the Web Services Discovery and retrieves adequate service descriptions. Then the SDA uses the SMA to achieve a finer-grained, semantic correspondence between query and service profile. If there is no provider of the requested service, the PA invokes the SCA to create a composite service. This service is then forwarded to the SEA, which is in charge of orchestrating its execution. The composite plan may include some abstract services, so the SEA may ask the SDA on-the-fly for adequate services [18].

The SDA is usually a form of a Mobile Agent (MA). In general, a MA is capable of roaming, finding, executing services and delivering results to the user/agent. A typical structure of a MA includes the following components: data state, migration policies, communication and code. The data state component contains the information carried by the MA during migrations; the migration policies component specifies the autonomous behavior of the MA. The communication component is responsible for the MA's communication with other agents or network entities and finally the code implements the MA's autonomous behavior with the support of the other three components. Each component is configured according to the MA's task. Moreover, it should be noted that the MAS platform provides a basic communication frame for the agents as well as general classes with the basic functionality needed either by stationary or by mobile agents [19].

The MA may follow several WS invocation alternatives and these are listed below [20]:
- Poll the servers where the services are located to check their availability
- Try to invoke the services from remote and not migrate to the service provider.
- Migrate to the Web Service Provider (WSP) and collaborate with the Service provider Agent (SPA).
- Migrate to the WSP and directly invoke the Web Service (WS).
- Finally, to send clones to each WSP, instead of migrating serially to each one. This scenario results to a parallel invocation of WSs where each MA clone invokes one WS.
When the MA clones have been used for service invocation, they return and deliver service results to the father MA. After this interaction the MA clones are destroyed. Consequently, the father MA delivers the services results to the user/agent.

The Composite Service Agents (CSA) can benefit from the above mentioned WS invocation used by the MA. Service composition is an important role of the Composite Service Agents (CSA). Web service composition is the ability to

take existing services and combine them to form new services. Web service composition can either be static or dynamic. In static composition, the services are predetermined during the design of a Web process. In a dynamic composition, the Web service that is to be deployed is decided at run-time. Dynamic composition is more suitable if the process has to adapt dynamically to unpredictable changes in the environment. A composite Web service is an aggregation of elementary and composite Web services, which interact with each other according to a process model [3, 21, 22]. From a user perspective, it is important to make sure that all these operations are carried out in a transparent way. Therefore, software agents are deemed appropriate to achieve this transparency.

7. CONCLUSION

In Web services environments with semantically annotated services, software agents are important entities that facilitate user's tasks in a transparent manner. This paper presented a variety of software agents that cooperate at different layers to ease/automate the users (humans or agents) tasks. It is also known that there exists a communication gap between software agents and Web services. This paper presented a middleware solution for integrating both technologies without changing their existing specifications and implementations.

8. REFERENCES

[1] G. Alonso, F. Casati, Web Services: Concepts, Architectures and Applications, Springer-Verlag, 2004.

[2] B. Benatallah and F. Casati (Guest Editors), "Special Issue on Web Services," Distributed and Parallel Databases, Kluwer Academic Publishers, 12(2-3), September 2002.

[3] Z. Maamar, Q.Z. Sheng, B. Benatallah, "On Composite Web Services Provisioning in an Environment of Fixed and Mobile Computing Resources" Information Technology and Management, Kluwer Academic Publishers, Vol 5, No 3, 2004.

[4] P. Davidsson, "Categories of Artificial Societies", Engineering Societies in the Agents World II, LNAI 2203, Springer-Verlag, 2001.

[5] W3C Web Service Architecture Working Group. http://www.w3.org/

[6] Wooldridge, Michael and Ian Dickinson. "Agents are not (Just) Web Services: Considering BDI Agents and Web Services." In Proceeding of the 2005 Workshop on Service-oriented Computing and Agent-based Engineering (SOCABE 2005), Jul 2005.

[7] Greenwood, Dominic and Monique Calisti. "Engineering Web Service-Agent Integration" In IEEE Systems, Cybernetics and Man Conference, October 2004.

[8] M. Shafiq, Y. Ding, and D. Fensel. "Bridging multi agent systems and web services: Towards interoperability between Software Agents and Semantic Web Services," In Proceedings of the 10th IEEE International Conference on Enterprise Distributed Object Computing (EDOC'06), Oct. 2006.

[9] D. Greenwood, M. Lyell, A. Mallya, and H. Suguri. "The IEEE FIPA Approach to Integrating Software Agents and Web Services," AAMAS'07, May 2007.

[10] M. Adacal and A. B. Benner, "Mobile Web Services: A New Agent-Based Framework," IEEE Internet Computing, May 2006.

[11] M. Wooldridge, An Introduction to Multiagent Systems, John Wiley & Sons, 2002.

[12] Q. Yu, X. Liu, A. Bouguettaya, and B. Medjahed, "Deploying and managing Web services: issues, solutions, and directions," The Very Large Data Bases (VLDB) Journal, Volume 17 , Issue 3, May 2008, pp. 537 – 572.

[13] FIPA Agent Management Specification http://www.fipa.org/specs/fipa00023

[14] Simple Object Access Protocol (SOAP), http:// www.w3.org/TR/soap/

[15] Michael N. Huhns and Larry M. Stephens, "Multiagent Systems for Internet Applications," in Practical Handbook of Internet Computing, Chapter 18, Chapman Hall and CRC Press, 2004.

[16] M. Singh, M. Huhns, Service-Oriented Computing Semantics, Processes, Agents. Wiley, New York, 2005.

[17] Francisco García Sánchez, Rodrigo Martínez-Béjar, Rafael Valencia-García, Jesualdo Tomás Fernández-Breis, "Knowledge Technologies-Based Multi-Agent System for Semantic Web Services Environments," ICAISC 2008, pp.1222-1233.

[18] César Cáceres, Alberto Fernández, Sascha Ossowski, and Matteo Vasirani, "An Abstract Architecture for Semantic Service Coordination in Agent-Based Intelligent Peer-to-Peer Environments," Proceedings of the 2006 ACM symposium on Applied computing SAC' 06, pp. 447-448.

[19] V. Baousis, E. Zavitsanos, V. Spiliopoulos, S. Hadjiefthymiades, L. Merakos, and G. Veronis, "Wireless Web Services Using Mobile Agents and Ontologies," ACS/IEEE International Conference on Pervasive Services, June 2006.

[20] V. Baousis, V. Spiliopoulos, E. Zavitsanos, and S. Hadjiefthymiades, "Semantic Web Services and Mobile Agents integration for efficient Mobile Services," International Journal of Semantic Web and Information Systems, January 2008, pp 1-19.

[21] L. Zeng et al, "Policy-Driven Exception-Management for Composite Web Services," Proceedings of the Seventh IEEE International Conference on E-Commerce Technology (CEC'05), 2005.

[22] M. C. Jaeger, G. Rojec-Goldmann, and G. Muhl. "QoS Aggregation for Service Composition Using Workflow Patterns," In Proc. of the 8th Int. Enterprise Distributed Object Computing (EDOC2004), September 2004.