A WSA-Based Architecture for Building Multi-Agent Systems

Aluizio Haendchen Filho  
UNIASSELVI–FAMESUL  
Centro Universitário Leonardo da Vinci  
Rio do Sul, SC, Brazil  
+55 (47) 3521-9299  
aluizio.famesul@uniasselvi.com.br

Hércules Antônio do Prado  
Embrapa Food Technology  
Unicarioca – Centro Universitário Carioca  
Rio de Janeiro, RJ, Brazil  
+55 (21) 2410-9557  
hercules@ctaa.embrapa.br, unicarioca.br

Carlos José Pereira de Lucena  
PUC-RIO  
Pontifícia Universidade Católica do RJ  
Rio de Janeiro, RJ, Brazil  
+55 (21) 2540-6915  
lucena@inf.puc-rio.br

ABSTRACT
This paper discusses some advantages of applying the SOA paradigm for the MAS development, showing a framework whose architecture follows the WSA reference model. We believe that service-oriented paradigm can simplify the MAS development because it demands a much simpler coordination level than the traditional approaches focused on the message-oriented model.

Keywords
Agent Oriented Software Engineering, Agent Architecture

1. INTRODUCTION
The rapid evolution of Internet has opened a new era in the distributed systems development: the bigger part of the information systems currently developed is focussed in Web applications. Typically, the information resources in Web systems are dynamic, distributed, and heterogeneous. Since these computing environments are opened, information resources can be connected or disconnected at any time. This ubiquity and its distributed and interconnected characteristics represent a natural field for Multi-Agent Systems (MAS).

Currently, enterprises are strongly interested in the strategic advantages of adopting distributed infrastructures that are designed to be dynamic, flexible, adaptable, and interoperable. In this context, an increasing demand for agent-based applications has required tools to simplify its development. The current agents frameworks need to be adapted to cope with specific requirements for Web architectures and emerging technologies [8, 10]. Some authors [4, 5] have pointed the relative immaturity of the agent's technology to interact with business applications as one of the causes that inhibit large-scale MAS. MAS' technology did not reach its potential yet and will not be largely used until adequate development tools have been made available [1, 4, 5].

2. WSA SPECIFICATIONS
A service-oriented architecture (SOA) plays an important role in the Web context. It defines an architectural style to build loosely coupled software applications that need services available

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS’07, May 14–18, 2007, Honolulu, Hawai’i, USA.
Copyright 2007 IFAAMAS.

978-81-904262-7-5 (RPS) ©2007 IFAAMAS

in the Web. A Web Service [8] is an instance of SOA whose interfaces and bindings are capable of being described and discovered by XML artifacts supporting direct interactions with other software applications using XML based messages via Internet protocols.

The standards for Web, governed by W3C consortium [2] affect strongly the development of a great variety of applications, including the MAS development. The increasing expansion of Web applications led the W3C to propose the WSA (Web Services Architecture) as a standard for services-based architectures. The first version was released in 2003 (W3C Working Draft 8 August 2003) [3], being updated in further versions.

Following the SOA paradigm, WSA introduces a set of concepts and abstractions for Web architectures. It aims to provide a common Web Service definition and its location inside a wider architecture in order to guide service implementers, authors of services specification and Web application developers. WSA defines different Web Service views, described upon the common core of agents, messages and services.

Five models compose the reference architecture [11]: Message-Oriented Model (MOM): dedicated to the architectural issues related to the structure and transport of messages; Service-Oriented Model (SOM): focussed in the services and actions executed by requesters and providers; Resource-Oriented Model (ROM): focussed in the architectural aspects related to the resources handling; Management Model (MGT): devoted to the management of messages, services, resources and its lifecycle; Policy Model: defines the core of concepts to describe the quality policy and Web services security.

Built upon a set of reusable roles, WSA represents the natural evolution from message to service-oriented architectures [8]. The structure forms modular and tight building blocks, simplifying the development, understanding and evolution of the architecture.

3. MIDAS
MIDAS [9] is a software framework designed to simplify the MAS development. Adopting the WSA specifications, the architecture is based on the coexistence of several containers, each one executing a JVM (Java Virtual Machine). It provides a complete execution environment, where agents can execute concurrently in the same host. Figure 1 shows the MIDAS generic architecture, detailing the Front-End Server (FES) and Agent Container (AC) concepts.
The Sixth Intl. Joint Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS 07) 907

Figure 1. The main concepts of the MIDAS

FES is responsible for the platform integration rules, synchronizing the containers and interoperating with external applications. It contains two gateways: an HTTP for intra-platform communication and a SOAP to interact with external applications. It is also responsible for automatic generation of Web Services, creating the JWS (Java Web Services) files, WSDL descriptors and UDDI registers. AC is a Web container where agent-based applications can be instantiated. Its architecture is composed by two basic structures: one represented by middleware agents and other represented by application agents.

Placed in the AC and FES, the middleware agents (SOM, MOM, ROM, MGT) provide infrastructure services, playing in a collaborative and pro-active way the roles defined by the reference architecture. The introduction of the agent concept to play these roles complies with the current tendency and non-functional requirements for Web architectures: flexibility, dynamic behavior, pro-activity, and adaptability. They abstract completely the generic characteristics, such as communication, concurrency, lifecycle management, services discovery and interoperability, enabling the developer to focus the development only in specific characteristics of the applications.

Application agents and components are only placed in the AC containers. Besides infrastructure services, AC provides a structure composed by abstract classes (Agent and Component) and a blackboard. The application agents (or components) are instantiated in organizations and developed by extending the abstract classes, from which specific application behavior can be implemented. Different from agents, components are purely reactive entities, normally used to encapsulate specific rules of the application domain, such as business processes and data access objects. The blackboard [6] offers a powerful mechanism to support the development of cognitive agents.

The platform includes resources representations, which can be stored in XML files, relational databases or documents in general. The General Cache Descriptor element placed in the FES represents the platform-consolidated structure of resources, kept in cache memory to make faster the access to the services specification. In the AC container, the LXS element represents the services specification described in XML. As well as in FES, the resources structure is also kept in cache memory, being represented by the LCS element. LDB represents local databases, which can be handled and/or accessed by agents and components.

The MOM agent focuses on the architectural aspects related to the message transport: send/receive, pack/unpack, and managing exceptions. When placed in FES, it is responsible by the bi-directional interoperability with Web Services. The external SOAP requests arriving to the FES are received by the SOAP gateway, which works with AXIS and Tomcat servers, both embedded in the platform. In AC, external Web services are registered and encapsulated in wrapper components. When an agent needs to invoke an external Web Service, the request is directed from AC to FES that translates it to the SOAP format. Inside the platform, the messages interchanged between FES and ACs move on HTTP, being the communication assured by means of servlets.

The ROM agent plays the roles defined by the resource-oriented model, focusing the relevant aspects related to the resources concept. A resource description is a machine runnable metadata representation that makes possible for a human or software program to locate services. Considering that a service is an abstract representation, it needs a structure of resources to have a concrete representation. Described in XML file, the structure form a hierarchy, where the root node is the AC and the leaves are the services and its specification. The ROM agent keeps the structure, playing similar roles to the FIPA DF [1] and other agents, such as Matchers and Yellow Pages.

The MGT is the most complex agent in the architecture, playing the roles defined at the management and policy levels. It involves a set of tasks that enable the control over the platform, such as the life cycle management, checking activities, statistics, QoS (Quality of Services) reporting, and GUI (Graphical User Interface) wizards. According to Booth [3], Web architectures become manageable when it holds a set of operations to support management ability, i.e., messages, services, resources, and security.

The roles defined by the service-oriented model are played by the SOM agent, which focuses on the architectural aspects related to the messages processing. It acts as a service provider representative, being responsible by the dynamic configuration and creation of instances. Dynamic configuration focuses on the capacity of an agent, running in the JVM, to be changed at runtime by another modified agent. It is also responsible by the messages verification and authentication, consulting rules that can be defined by the developer. There is no SOM in the FES, since it is not related to application services.

4.DISCUSSION

In this section we compare our approach to JADE [1] for two main reasons. First, because JADE and MIDAS use the same basic principles: both are frameworks that employ middleware agents to provide infrastructure services. Second, both approaches work based on established architectures. While JADE uses the FIPA (Foundation for Intelligent Physical Agents) [7] specification, MIDAS adopts the WSA standard.

The reference models supply the basis for the comparison. The FIPA specification uses three different categories of middleware agents: DF (Directory Facilitator), ACC (Agent Communication Model) and AMS (Agent Management System). They can be translated to the MOM, ROM and MGT agents of MIDAS, respectively, as shown in Figure 2.
Following the FIPA specification, JADE does not provide an abstraction for services layer. The inexistence of a service layer requires the creation of excessive behaviors, for each service in the agents, making the process extremely laborious to the developer. Moreover, as there is no abstraction for services, a special agent needs to be programmed to play this role. The implementation is forced to simulate a proxy between layers, carrying out all redirections manually, resulting in a significant increase of development time and chance of error occurrence.

While in JADE the agents must handle request objects directly, in MIDAS there is a middleware agent playing this role. The SOM agent provides location transparency, eliminating the complexities to identify services providers and to re-direct requests. It is enough to know the desired service name, and the architecture takes care of the remaining procedures. It is possible to insert new services only registering them in the XML file. There is a request abstraction, standardizing the parameters being handled which are passed along with a request. The entities are invoked through an only interface, instead of implementing different classes for each service, reducing cost and development time.

We believe that SOA paradigm can simplify the MAS development because it demands a much simpler coordination level than traditional approaches focused on the message model. SOA enable software components, including application functions, objects, and system processes to be exposed as services, becoming it easily accessible and making possible easy interoperability among heterogeneous applications. It allows designing more configurable, adaptable, flexible and reusable applications. On the other hand, strongly coupled architectures limit the reuse, flexibility, extensibility, adaptability of the architecture and the applications generated by them. Besides a bigger effort and development time, they can bring additional complexity and are hardly adaptive when the distributed points need to change at runtime.

Our approach has made two contributions, extending the current WSA model for MAS development and showing how it can be designed using the reference architecture. The contributions can be summarized as follows:

1. The introduction of the abstract agent to represent application agents extends the WSA concepts, enabling then for the MAS development. The abstract agent provides a way of factorizing the common properties to all agents into one super-class, providing a half-ready implementation of an application agent. The introduction of the blackboard solves a deficiency found in architectures that uses the SOA paradigm. Using the blackboard, it is possible to simulate a communication model through which agents can send messages for an only agent, a group of agents or for all the agents.

2. The detailed architecture [9] can be used as a guide for MAS building under the WSA reference model. Currently, there are no works in the literature showing how MAS can be built by employing the WSA reference architecture. The structures may serve as basis to lead developers to a better understanding of architectures that employ the WSA specifications.

The introduction of software agents performing infrastructure services as encapsulations of dynamic behavior highlights their participation in the proposed architecture. Moreover, by considering middleware and application agents as different entities, the architecture enables a clearer and more coherent view of the agents' roles and its participation in the architecture.

5. REFERENCES


The Sixth Intl. Joint Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS 07)