A Multi-paradigm Approach for Mobile Agents Development

Edgardo A. Belloni[1]

Abstract

Mobile agent systems have received important attention in the last years as a new programming paradigm for widely distributed and heterogeneous systems. In this article, a multi-paradigm approach for the development of intelligent mobile agents is presented. It integrates both object-oriented and logic paradigms. The rationale for this approach comes from the fact that although the object-oriented programming paradigm has relevant features for mobile agent development it presents deficiencies dealing with agents' mental attitudes. These deficiencies are solved by the use of logic programming.

Keywords: Mobile agents; Multi-paradigm languages; Software architecture; Socialware.

1 Introduction

In a broad sense, an agent is any computer program that acts on behalf of a (human) user. In this context, a mobile agent is a computer program, which represents a user in a computer network, and it is capable of migrating autonomously from node to node, to perform some computation on behalf of the user [Karnik and Tripathi 98].

Mobile agents are an emerging technology attracting interest from the fields of distributed systems, information retrieval, distributed artificial intelligence and electronic commerce. This stems from the capabilities of the mobile agents to reduce network usage, execute asynchronously and autonomously and be fault-tolerant among another ones.

At present, Java is the chosen (object-oriented) programming language for the development of mobile agent systems since it has many relevant features that directly support implementation of mobile agents [Wong et al 99]. However, in order to develop intelligent agents, the object-oriented paradigm generally does not allow complex mental attitudes to be represented [Amandi et al 99]. The logic-programming paradigm is an evident support to represent and infer relationships among mental attitudes such as intentions, goals and beliefs.

In this article, a multi-paradigm approach for the development of intelligent mobile agents is presented. It integrates both object-oriented and logic paradigms for the design and programming of mobile agents. This approach is essentially materialized by a software architecture based on JavaLog [Amandi et al 99][Zunino et al 99], a programming language that integrates both Java and Prolog.

The article is structured as follows. Section 2 introduces mobile agents paradigm discussing its basic concepts and comparing it with earlier paradigms for distributed computing from which it has evolved. Some applications, which clearly benefit from using mobile agents, are also introduced. Section 3 analyses the rationale and motivations for the proposed approach, and Section 4 introduces a basic software architecture necessary to materialize it. The several components of this architecture are described, and how the mobile agents integrate both the logical and object-oriented programming paradigms is discussed. Section 5 provides a summary of ongoing experimental works in order to evaluate the effectiveness of the approach. Finally, in section 6, some concluding remarks are delineated and options for future work are discussed.

[1] Instituto de Sistemas Tandil (ISISTAN) - Facultad de Ciencias Exactas - UNCPBA.

Campus Universitario - Paraje Arroyo Seco. (B7001BBO) Tandil, Bs. As., Argentina.

E-mail: ebelloni@exa.unicen.edu.ar

2 Agents and Mobile Agents

There are many definitions of agents. Agents can be defined by a set of their attributes: active, autonomous, goal-driven, and typically acting on behalf of a user or another agent. Agents are not a new paradigm, as they have been researched in the area of Distributed Artificial Intelligence for a number of years.

The widespread use of computers and their connectivity, particularly the web and the Java object-oriented programming language have provided a new influx in the research, development, and deployment of agents. A particular motivation for the use of agents is the huge amount of information available on the Internet. Agents have a significant potential looking for information, filtering it, and extracting it from the source. The ability to represent and act on behalf of the user represents a crucial capability of agents and provides enormous potential for their deployment.

There are at least two communities currently pursuing agent research: the multi-agent system community and the mobile agents' community. The multi-agent and intelligent system community deals mostly with stationary agents distributed on the network, which communicate one another in order to pursue a common goal. The other community deals with mobile agents as a paradigm for widely distributed and heterogeneous systems. People working on system research, with background in operating systems, distributed systems and object-oriented research primarily represent this community.

In this context, to develop more powerful approaches, regarding a mobile agent can also be intelligent, or belong to a multi-agent group, becomes increasingly important.

Mobile Agents: a paradigm for widely distributed and heterogeneous systems

The basic concepts of mobile agent systems are places and agents [White 97]. A mobile agent system involves a number of places - also called nodes or servers - where computation can take place and where various services and resources are provided. In this context, mobile agents are active entities, which can move from place to place to meet other agents or to access to services and resources provided there.

Essentially, the mobile agent paradigm has evolved from two antecedents: client-server and remote evaluation paradigms [Karnik and Tripathi 98]. Traditionally, distributed applications have relied on the client-server paradigm in which client and server processes communicate either through message-passing or remote procedure calls (RPC). This communication form is synchronous, i.e., the client suspends itself after sending a request to the server, waiting for the results of the call. In the early nineties, Remote evaluation (REV) was proposed as an alternative paradigm. In REV, the client rather than invoking a remote procedure sends its own procedure code to a server, and requests the server to execute it and return the results.

In RPC, data is transmitted between the client and server in both directions. In REV, a code is sent from the client to the server, and data is returned. In contrast, a mobile agent is a program (encapsulating code, data and execution context) sent by a client to a server. Unlike a procedure call, it does not have to return its results to the client. It could migrate to other servers, transmit information back to its origin, or go back to the client if appropriate.

Mobile agents can be also considered as the last point of the incremental evolution of mobile abstractions such as mobile code, objects, and process [White 98]. A transferred state of mobile code such as applets consists only of code. A mobile object state consists of code and data. A mobile process state in addition includes the thread state. The mobile agent state consists of code, data, thread and authority of its owner.

According to the movement pattern [White 98], mobile agents differ from applets (downloaded from the server to the client), and from the servlets (uploaded from client to server) in that they can have multiple hops, and that they can be detached from the client. A mobile agent autonomously visits places without interacting continuously with its originating place. This originating place gets involved only when the agent is sent to the first visiting place, and when the agent returns from the last visited place.

Benefits and Applications of Mobile Agents

Several advantages of mobile agent systems, in comparison with other approaches for distributed systems have been identified. Major advantages of mobile agents are seen in the possibility of reducing expensive global communication costs by moving the computation to the data and in the possibility to distribute complex computations onto several, possibly heterogeneous hosts [Harrison et al 95].

Mobile agents can be useful for many applications, e.g., electronic commerce, distributed information retrieval, personal assistance and workflow applications. These applications clearly benefit from using mobile agents because they: reduce the network load; overcome network latency; execute asynchronously and autonomously; are fault-tolerant and can sense their execution environment and react dynamically to changes [Lange 99].

3 Intelligent Mobile Agents: a multi-paradigm approach

Object-oriented languages have characteristics that satisfy part of the programming requirements of both agents and mobile agents.

Shoham [Shoham 93] has introduced the term agent-oriented programming. As a form of object-oriented programming, an agent's state consists of beliefs, capabilities, choices, and similar notions, and the computation consists of interactions, such as informing, offering, accepting, rejecting and competing [Shoham 97]. In this context, an object can model an agent. The object's methods represent the agent's abilities (the agent's behavioral capabilities) and the object's variables of instance represent the agent's mental state (the agent's knowledge).

At present, Java is the most frequent programming language for the development of mobile agent systems. Aglets [Lange and Oshima 98], Voyager [ObjectSpace Corp. 98] and Odissey [General Magic Corp. 98] are examples of Java-based mobile agent systems. Multi-platform support and the ubiquity of the Java virtual machine make Java particularly well suited for mobile agent technology. Networking support of Java includes: sockets; URL communication and a distributed object protocol called remote method invocation (RMI). These features smooth the progress of dissemination of mobile agents throughout the Internet. Furthermore, Java has other features not found in any other language that directly support implementation of mobile agents [Wong et al 99]. For instance, Java facilitates migration of agent's code and state via its class-loading and object serialization mechanisms.

In spite of the benefits that object-oriented languages represent for the agent-oriented programming in general, and particularly the Java language for the mobile agent programming, these languages have limitations to develop intelligent mobile agents. These limitations are demonstrated at the time of dealing with the agents' mental attitudes. In this case, different algorithms of inference, applied on the instance variables that represent the knowledge of the agent, must be implemented. These algorithms are, in general, expensive to implement and they also provide little scope of flexibility facing changes [Amandi et al 99][Zunino et al 99].

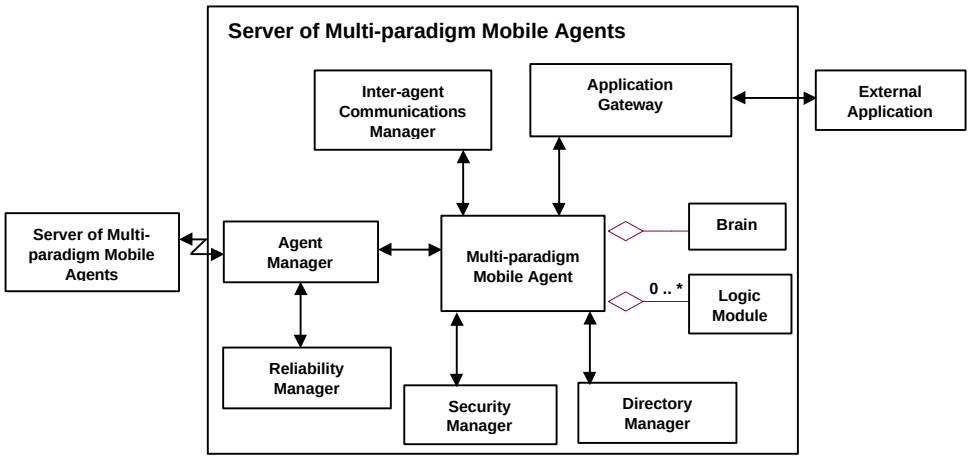
It is widely accepted that the logic-programming paradigm represents an appropriate alternative for managing mental attitudes due to its evident support to represent and infer relationships among mental attitudes such as intentions, goals and beliefs. Logic languages enable us to represent mental attitudes in declarative form by logic clauses. Deductive algorithms interpret these clauses that, in the context of the agent-oriented programming, give origin to agents' knowledge dependent reasoning.

Therefore, it can be concluded that a multi-paradigm approach, integrating both object-oriented and logic programming, represents a better approach to support the development of intelligent mobile agent systems. It aims to solve the limitations mentioned above, enabling a mobile agent to manage complex mental attitudes.

The fundamental idea of this work is the materialization of a generic software architecture to support mobile agents based on a multi-paradigm language. This language integrates both Java and Prolog programming languages.

4 A Software Architecture Supporting Multi-paradigm Mobile Agents

The following figure shows the necessary software architecture to materialize the proposed approach. This is essentially founded on the generic architecture of Java-based mobile agents described in [Wong et al 99]. The diagram uses UML notation [Rumbaugh et al 99].



This architecture includes the following major components: an agent manager, an inter-agent communications manager, a security manager, an application gateway, and a directory manager.

Responsibilities of the components

The agent manager receives agents for execution on the local host and sends agents to remote hosts. Previous to transport, the agent manager serializes the agent and its state. It then passes the serialized form to its counterpart on the destination host. In a highly reliable architecture, it actually passes the agent off to the reliability manager, which ensures that the agent manager on the remote host receives the agent. Upon receipt of an agent, the agent manager reconstructs the agent and the objects it references, and then creates its execution context.

The security manager authenticates the agent before it is allowed to execute. Thereafter, the Java virtual machine automatically invokes the security manager to authorize any operations using system resources.

The inter-agent communication manager facilitates communication between mobile agents dispersed throughout a network. Agents may use the directory manager to identify the location of an application server and then migrate to the host on which the server is located. The application gateway serves as a secure entry point through which agents can interact with application servers. An arriving mobile agent accesses to resident servers such as database servers through this gateway.

This architecture prescribes that an agent is a composite Java object that includes mobility, persistence, and can communicate with other agents. Each mobile agent runs into its own thread of execution and it is implemented through a programming style involving a call-back model based on the Java event delegation model. During its life cycle, a mobile agent receives various kinds of events in response to its actions. For instance, if it moves to another host, a mobility event occurs just before and after the migration and corresponding call-back methods are invoked. In this way, each event gives to the agent the opportunity to determine how to react. A programmer implements a mobile agent filling its call-back methods, such as *beforeDispatch* and *afterArrival*, as appropriate.

Mobile agents integrating both object-oriented and logic paradigms

The architecture also prescribes that the mobile agents are implemented with JavaLog [Amandi et al 99][Zunino et al 99], a multi-paradigm language developed at the ISISTAN, which integrates both the Java and Prolog programming languages.

JavaLog uses objects in order to model the agents' abilities and logic clauses to model the agents' mental state. The agents' abilities can be represented by capabilities of action as well as by capabilities of communication with other agents, whereas the mental state can appear in form of beliefs, intentions and commitments.

JavaLog defines a module as its basic concept of manipulation. In this sense, both an object and a method from the object-oriented paradigm are considered modules. An object is a module that encapsulates data and a method is a module that encapsulates behavior. The elements manipulated by the logic paradigm are also mapped to modules. Logic modules are defined as a set of Horn clauses and two algebraic operators, union and overriding union, are applied to combine logic modules.

Each agent implemented with JavaLog encapsulates a complex object called *brain*. This object is an instance of an interpreter of logic modules using Prolog clauses. This interpreter is implemented in Java and it enables us to use objects within logic clauses, as well as to embed logic modules within Java code. In this way, each agent is an instance of a class that can define part of its methods in Java and part in Prolog. The definition of a class in JavaLog can include several logic modules defined within methods as well as referenced by instance variables.

5 Experimental Work

The proposed approach represents one of the current lines into an ongoing research and development project, at ISISTAN. This project aims to develop intelligent agents in the socialware[2] domain [Trilnik et al 00]. For this purpose, some applications of information systems, multi-agent and integrated to the World Wide Web, are being developed in order to assist in various social activities on virtual communities. These activities include, for instance: electronic commerce for solidarity; planning of meetings; edition, publishing and collecting of community news; and calls or offers of volunteers to campaigns of solidarity. All of them are based on personal preferences.

It is our objective to demonstrate the strengths and weaknesses of our multi-paradigm approach for mobile agents' development applying it in this context. Applications, in this project, can be benefited from using mobile agents representing and acting on behalf of the different members of a virtual community when they are developing social activities on remote communities. These agents could transport the preferences and restrictions of the users they represent, encapsulating these mental attitudes in declarative form.

6 Conclusions and Future Work

In this article, a multi-paradigm approach for the development of mobile agents has been introduced. This approach is materialized by a software architecture integrating both logic and object-oriented programming paradigms in order to support the development of intelligent mobile agents.

The effectiveness of the approach is being evaluated applying it to the development of multi-agent systems in the socialware domain.

An extension of JavaLog's logic modules, which migrate to another host using a *strong* mobility model, is also being explored. It aims to enable an executing unit to move as a whole by retaining its executing state - i.e. retaining control and internal data associated to the executing unit, e.g., the program counter or the call stack - across migration. In this case, migration is transparent, in that the executing unit resumes execution on the new host right after the instruction that triggered the migration.

[2] Socialware: multi-agent systems developed in order to assist in various social activities on network communities [Hattori et al 99].

Acknowledgments

This article is a reviewed version of [Belloni and Campo 00]. We are grateful to the participants of the WICC 2000 workshop, who gave us useful feedback.

References

[Amandi et al 99] A. Amandi, A. Zunino and R. Iturregui. Multi-paradigm Languages Supporting Multi-agent Development. In Multi-Agent System Engineering. F. J. Garijo and M. Boman (Eds.). Lecture Notes in Computer Science, Vol. 1647, pp. 128–139. Springer-Verlag, Berlin - Heidelberg - New York, 1999.

[Belloni and Campo 00] E. Belloni, M. Campo. Una Arquitectura de Software para Soportar Agentes Móviles Inteligentes. Actas del Workshop de Investigadores en Ciencias de la Computación WICC 2000, pp. 65-67. La Plata, Bs. As., Argentina. Mayo de 2000.

[General Magic Corp. 98] Odissey White Paper. General Magic Corp. Cupertino, Calif., 1998.

[Harrison et al 95] C. G. Harrison, D. M. Chess and A. Kershenbaum. Mobile Agents: Are they a good idea? Technical report, IBM Research Division, T.J. Watson Research Center, March 1995. Available at URL http://www.research.ibm.com/massdist/mobag.ps

[Hattori et al 99] F. Hattori, T. Ohguro, M. Yokoo, S. Matsubara and S. Yoshida. Socialware: Multiagent Systems for Supporting Network Communities. Communications of the ACM. Vol. 42, No. 3, pp. 55-61. March 1999.

[Zunino et al 99] A. Zunino, A. Amandi, R. Iturregui. JavaLog: una integración de objetos y lógica para la programación de agentes. V Congreso Argentino de Ciencias de la Computación. Tandil, Bs. As., Argentina. Octubre de 1999.

[Karnik and Tripathi 98] N. Karnik and A. R. Tripathi. Design Issues in Mobile Agent Programming Systems. IEEE Concurrency, Vol. 6, No. 3, July-September 1998.

[Lange 99] D. Lange. Seven Good Reasons for Mobile Agents. Communications of the ACM. Vol. 42, No. 3, pp. 88-90. March 1999.

[Lange and Oshima 98] D. Lange and M. Oshima. Programming and Deploying Java Mobile Agents with Aglets. Addisson-Wesley Longman, Reading Mass., 1998.

[ObjectSpace Corp. 98] Voyager White Paper. ObjectSpace Corp. Dallas, Texas, 1998.

[Rumbaugh et al 99] J. Rumbaugh, I. Jacobson and G. Booch. The Unified Modeling Language Reference Manual. Reading, Addison-Wesley, 1999.

[Shoham 93] Y. Shoham. Agent-Oriented Programming. Artificial Intelligence, 60(1), pp. 51-92, March 1993.

[Shoham 97] Y. Shoham. An Overview of Agent-Oriented Programming. In Software Agents, J. M. Bradshaw (Ed.), pp. 271-290. MIT Press, 1997.

[Trilnik et al 00] F. Trilnik, D. Cordero, E. Belloni, M. Campo, A. Amandi. Agentes Inteligentes Aplicados a Análisis de Sociedades. Actas del Workshop de Investigadores en Ciencias de la Computación WICC 2000, pp. 78-80. La Plata, Bs. As., Argentina. Mayo de 2000.

[White 97] J. White. Mobile Agents. In Software Agents, J. M. Bradshaw (Ed.), pp. 437-472. MIT Press, 1997.

[White 98] J. White. Personal communication, 1998. In Mobility: Process, Computers and Agents. D. Milojièiæ, F. Douglis and R. Wheeler (Eds.). Addison-Wesley. 1999

[Wong et al 99] D. Wong, N. Paciorek and D. Moore. Java-based Mobile Agents. Communications of the ACM. Vol. 42 - No. 3, pp. 92–102, March 1999.