Towards a Standardization of Multi-Agent System Frameworks

by Roberto A. Flores-Mendez

Introduction

Agents are everywhere. People encounter intelligent agents, information agents, mobile agents, personal assistant agents, and other types of agents daily. One might wonder if it is possible to make some sense of this apparent anarchy: What is it that makes an agent? Is there something that agents have in common? How can one organize them to carry out tasks?

Fortunately, we are not the first to ask these questions, and researchers have already defined some helpful notions. First, authors have acknowledged that **agents** are entities within an environment, and that they can sense and act (not necessarily in that order). This means that agents are not isolated entities, and that they are able to communicate and collaborate with other entities. Simply put, agents that are not able to work together with other agents (humans included) are destined to become virtually useless.

Once agents are ready for collaboration, they will need to find the other agents they need to collaborate with. Such a task is easy if they know exactly which agents to contact and at which location. However, a static distribution of agents is very unlikely to exist: people are usually on the move and they are not always readily available to interact with others. The same holds true for dynamic multi-agent systems: agents need support to find other agents.

The dynamic nature of agent distribution motivates research by groups working on the standardization of dynamic collaborative multi-agent systems. Some of these groups are the Foundation for Intelligent Physical Agents (FIPA), the Object Management Group (OMG), the Knowledge-able Agent-oriented System (KAoS), and the General Magic group.

This article briefly describes these groups' efforts toward the standardization of multi-agent system architectures, and outlines early work at the University of Calgary to define multi-agent systems architectures. However, the main objective of this article is to give the reader a basic overview of the background and terminology in this exciting area of research.

Background

Although agent researchers come from a variety of backgrounds, the Distributed Artificial Intelligence (DAI) and the Distributed Computing (DC) communities stand out as traditional agent research areas.

During the mid-1970s, researchers from DAI began to formulate some of the basic theories, architectures, and experiments that showed (computationally speaking) how interaction and division of labor could be effectively applied to problem solving [15]. Experiments showed that intelligent, rational behavior is not an attribute of isolated components, but rather an outcome that emerges from the interaction of entities with simpler behaviors [4, 10].

More recently, DC became an active discipline in agent research. DC is challenged to integrate heterogeneous, largely autonomous computer components that span several generations as part of collaborative environments. From this perspective, agents are applied as interaction entities to mediate differences among components, while providing a syntactically uniform and semantically consistent intermediary role [21].

Agents Overview

The term ``agent'' is difficult to define. Agents are often described as entities with attributes considered useful in a particular domain. This is the case with intelligent agents, where agents are seen as entities that emulate mental processes or simulate rational behavior; personal assistant agents, where agents are entities that help users perform a task; mobile agents, where entities that are able to roam networking environments to fulfill their goals; information agents, where agents filter and coherently organize unrelated and scattered data; and autonomous agents, where agents are able to accomplish unsupervised actions.

Attributes versus Attributions

Several researchers have attempted to provide a meaningful classification of the attributes that agents might have. A list of common agent attributes is shown below [2].

- **Adaptivity**: the ability to learn and improve with experience.
- **Autonomy**: goal-directedness, proactive and self-starting behavior.
- **Collaborative behavior**: the ability to work with other agents to achieve a common goal.
- **Inferential capability**: the ability to act on abstract task specifications.
- **``Knowledge-level'' communication ability**: the ability to communicate with other agents with language more resembling human-like ``speech acts'' than typical symbol-level program-to-program protocols.
- **Mobility**: the ability to migrate in a self-directed way from one host platform to another.
- **Personality**: the ability to manifest attributes of a ``believable'' human character.
- **Reactivity**: the ability to selectively sense and act.
- **Temporal continuity**: persistence of identity and state over long periods of time.

According to their attributes, agents could be classified as showing weak or strong notions of agency [24]. The weak notion of agency, which comes from DC and DAI, sees agents as a paradigm of network based cooperative automation. The strong notion of agency, from Artificial Intelligence (AI), leads toward an anthropomorphic view where agents are seen as conscious, cognitive entities that have feelings, perceptions and emotions just like humans [26].

Nevertheless, it has been asserted that agents cannot be characterized solely based on their attributes; they need to be classified based on their attributes as perceived by humans. A simple categorization of behavior can be achieved by applying the following three predictive stances [9]:

- **Physical stance**: predictions are based on physical characteristics and laws.
- **Design stance**: predictions are based on what a system is designed to do.
- **Intentional stance**: predictions are based on assumptions of rational agency (e.g., beliefs, intentions, desires, and so on).

It is interesting to note how human perception of intelligent behavior influences the definition of agency:

> ``[It] helps us understand why coming up with a once-and-for-all definition of agenthood is so difficult: one person's 'intelligent agent' is another person's 'smart object'; and today's 'smart object' is tomorrow's 'dumb program.' The key distinction is in our expectations and our point of view " [2].

What Do We Mean By Agent?

One aspect of agents that is broadly mentioned in the literature is the notion of agents as interactive entities that exist as part of an environment shared with other agents. This definition of an agent is taken from descriptions given by several authors, who describe agents as conceptual entities that perceive and act [4, 33] in a proactive or reactive manner [24] within an environment where other agents exist and interact with each other [34] based on shared knowledge of communication and representation [14].

Objects versus Agents

Agents and objects share many characteristics; this sometimes makes it hard to differentiate between them. For example, agent-oriented programming (AOP) could be considered a specialization of the object-oriented programming (OOP) paradigm [34]. OOP views systems as consisting of objects communicating with one another to perform internal computations, whereas AOP specializes this view to have agents (instead of objects), whose internal computations are based on beliefs, capabilities, and choices, that communicate with each other using messages adopted from speech-act theory.

Although this view allows one to appreciate the similarities between agent and objects, their differences are less obvious. There are three main differences between agents and objects that have been identified:

> ``The first is in the degree to which agents and objects are autonomous. We thus do not think of agents as invoking methods upon one-another, but rather as requesting actions to be performed. In the object-oriented case, the decision lies with the object that invokes the method. In the agent case, the decision lies with the agent that receives the request.
>
> ``The second important distinction ... is with respect to the notion of flexible (reactive, pro-

active, social) autonomous behavior.

``The third important distinction ... is that agents are each considered to have their own thread of control ... in the standard object model, there is a single thread of control in the system'' [23].

Multi-agent Systems Overview

Terminology

As part of their study of agent systems, researchers began to develop (purposely or otherwise) terminology for agents. Some of these terms are described as follows [21]:

- **Agent Architectures** analyze agents as independent reactive/proactive entities. Agent architectures conceptualize agents as being made of perception, action, and reasoning components. The perception component feeds the reasoning component, which governs the agents' actions, including what to perceive next.
- **Agent System Architectures** analyze agents as interacting service provider/consumer entities. System architectures facilitate agent operations and interactions under environmental constraints, and allow them to take advantage of available services and facilities.
- **Agent Frameworks** are programming tools for constructing agents. Examples of these are Voyager [30], Aglets [22], and Odyssey [16].
- **Agent Infrastructures** provide the regulations that agents follow to communicate and to understand each other, thereby enabling knowledge sharing. Agent Infrastructures deal with the following aspects:
    - **Ontologies**: allow agents to agree about the meaning of concepts.
    - **Communication Protocols**: describe languages for agent communication.
    - **Communication Infrastructures**: specify channels for agent communication.
    - **Interaction Protocols**: describe conventions for agent interactions.

What is a Multi-agent System?

Various definitions from different disciplines have been proposed for the term **multi-agent system (MAS)**. As seen from DAI, a multi-agent system is a loosely coupled network of problem-solver entities that work together to find answers to problems that are beyond the individual capabilities or knowledge of each entity [11]. More recently, the term multi-agent system has been given a more general meaning, and it is now used for all types of systems composed of multiple autonomous components showing the following characteristics [23]:

- each agent has incomplete capabilities to solve a problem
- there is no global system control
- data is decentralized
- computation is asynchronous

One of the current factors (and arguably one of the more important ones) fostering MAS development is the increasing popularity of the Internet, which provides the basis for an open environment where agents interact with each other to reach their individual or shared goals. To interact in such an environment, agents need to overcome two problems: they must be able to find each other (since agents might appear, disappear, or move at any time); and they must be able to interact [23].

Finding Agents

There is a need for mechanisms for advertising, finding, fusing, using, presenting, managing, and updating agent services and information. To address these issues, the notion of *middle agents* [7] was proposed. Middle agents are entities to which other agents advertise their capabilities, and which are neither requesters nor providers from the standpoint of the transaction under consideration. The advantage of middle agents is that they allow a MAS to operate robustly when confronted with agent appearance, disappearance, and mobility.

There are several types of agents that fall under the definition of middle agents. Note that these types of agents, which are described below, are defined so vaguely that sometimes it is difficult to make a clear differentiation between them.

- **Facilitators**: agents to which other agents surrender their autonomy in exchange for the facilitator's services [2]. Facilitators can coordinate agents' activities and can satisfy requests on behalf of their subordinated agents.
- **Mediators**: agents that exploit encoded knowledge to create services for a higher level of applications [37].
- **Brokers**: agents that receive requests and perform actions using services from other agents in conjunction with their own resources [8].
- **Matchmakers** and **yellow pages**: agents that assist service requesters to find service provider agents based on advertised capabilities [3, 8].
- **Blackboards**: repository agents that receive and hold requests for other agents to process [28, 6].

Agent Interaction

Interaction is one of the most important features of an agent [29]. In other words, agents recurrently interact to share information and to perform tasks to achieve their goals. Researchers investigating agent communication languages mention three key elements to achieve multi-agent interaction [14, 21, 32]:

- A common agent communication language and protocol
- A common format for the content of communication
- A shared ontology

Agent Communication Language

There are two main approaches to designing an agent communication language [17]. The first approach is **procedural**, where communication is based on executable content. This could be accomplished using programming languages such as Java [1] or Tcl [30]. The second approach is **declarative**, where communication is based on declarative statements, such as definitions, assumptions, and the like. Because of the limitations on procedural approaches (e.g., executable content is difficult to control, coordinate, and merge), declarative languages have been preferred for the design of agent communication languages. Most declarative language implementations are based on illocutionary acts, such as requesting or commanding; such actions are commonly called **performatives**. One of the more popular declarative agent languages is KQML.

KQML

KQML, which is an acronym for **Knowledge Query and Manipulation Language** [18], was conceived both as a message format and a message handling protocol to support run-time knowledge sharing among agents [14]. This language can be thought of as consisting of three layers: a communication layer (which describes low level communication parameters, such as sender, recipient, and communication identifiers); a message layer (which contains a performative and indicates the protocol of interpretation); and a content layer (which contains information pertaining to the performative submitted).

```
(register
      :sender        agentA
      :receiver      agentB
      :reply-with    message2
      :language      common_language
      :ontology      common_ontology
      :content       ``(ServiceProvision Manufacturing:TaskDecomposition)''
)
```

Figure 1. Example of a KQML message.

The format of a KQML message is shown in Figure 1. The message in the example starts with the word ``register'', which is the action (performative) intended for the message. The remainder of the message contains keywords needed for the message and communication layers. Keywords used in KQML messages are defined as follows [25]:

- **sender**: agent sending the message.
- **receiver**: agent receiving the message.
- **from**: original sender; used when a message is sent using intermediary agents.
- **to**: final recipient; used when a message is sent using intermediary agents.
- **in-reply-to**: identifier of the message that triggered this message submission.
- **reply-with**: identifier to be used by a message replying to this message.
- **language**: language for interpreting the information in the content field of this message.
- **ontology**: identifies the ontology to interpret the information in the content field of this message.
- **content**: context-specific information describing the specifics of this message.

Ontologies

Ontologies are defined as specification schemes for describing concepts and their relationships in a domain of discourse [14]. It is important that agents not only have ontologies to conceptualize a domain, but also that they have ontologies with similar constructions. Such ontologies, when they exist, are called **common ontologies**. Once interacting agents have committed to a common ontology, it is expected that they will use this ontology to interpret communication interactions, thereby leading to mutual understanding and (ultimately) to predictable behaviors. **Ontolingua** [20] is often mentioned in the literature as a system that provides a vocabulary for the definition of reusable, portable and shareable ontologies. Ontolingua definitions are described using syntax and semantics similar to those of the **Knowledge Interchange Format** [19], also known as KIF, which is a format to standardize knowledge representation schemes based on first-order logic.

MAS Architectures Review

The design of computer programs as multi-agent systems presents a useful software engineering paradigm where systems are described as individual problem-solving agents pursuing high-level goals. Although having such an abstraction seems promising, its widespread adoption among system designers has not materialized yet. One reason is that MAS development is a technically difficult task. Efforts are challenged not only by known distributed programming issues, but also by the complexities associated with supporting agent collaboration.

If the agent-oriented paradigm is to succeed, systematic methodologies will be required for specifying and structuring applications as multi-agent systems. Once these methodologies are agreed upon, it will only be a matter of time until commercial MAS development toolkits emerge, and agent technology becomes accessible to a wide variety of software developers.

MAS Architectures Standardization

Although agent research started more than two decades ago, few efforts have been directed toward a definition of an acceptable MAS architecture. It is possible that one of the reasons for such an absence of consensus might be the common misconception in research circles that MAS architectures and frameworks need to be designed from first principles to match project requirements [39]. Although this approach might prove time efficient for individual projects, it certainly creates incompatible systems that are difficult to reuse from project to project. Therefore, one might expect that the widespread adoption of MAS technology will only begin after the formalization and standardization of architectures, mechanisms, and protocols supporting distributed interoperation of agents [35].

Recently, several independent industrial and research groups started to pursue the standardization of multi-agent technology. Prominent efforts, such as those of the Object Manager Group (OMG), the Foundation for Physical Agents (FIPA), the Knowledge-able Agent-oriented System (KAoS) group, and the General Magic group are briefly described below.

OMG's Model

The OMG group proposes a reference model as a guideline for the development of agent technologies [35]. This model outlines the characteristics of an agent environment composed of agents (i.e., components) and agencies (i.e., places) as entities that collaborate using general patterns and policies of interaction. Under this model, agents are characterized by their capabilities (e.g., inferencing, planning, and so on), type of interactions (e.g., synchronous, asynchronous), and mobility (e.g., static, movable with or without state). Agencies, on the other hand, support concurrent agent execution, security and agent mobility, among others.

FIPA's Model

The Foundation for Intelligent Physical Agents (FIPA) is a multi-disciplinary group pursuing the standardization of agent technology. This organization has made available a series of specifications to direct the development of multi-agent systems. Of particular importance are their Agent Management [12] and Agent Communication Language [13] specifications. FIPA's approach to MAS development is based on a ``minimal framework for the management of agents in an open environment.'' This framework is described using a reference model (which specifies the normative environment within which agents exist and operate), and an agent platform (which specifies an infrastructure for the deployment and interaction of agents).

KAoS' Model

Another important standardization effort is pursued by researchers of the Knowledge-able Agent-oriented System [3] architecture. This system, which is also known as KAoS, is described as ``an open distributed architecture for software agents.'' The KAoS architecture describes agent implementations (starting from the notion of a simple generic agent, to role-oriented agents such as mediators and matchmakers), and elaborates on the interactive dynamics of agent-to-agent messaging communication by using conversation policies.

General Magic's Model

General Magic is a commercial endeavor researching mobile agent technology for electronic commerce. Conceptually, this technology models a MAS as an electronic marketplace that lets providers and consumers of goods and services find one another and transact business. This marketplace is modeled as a network of computers supporting a collection of places that offer services to mobile agents. Mobile agents, which are entities that reside in one particular place at a time, have the following capabilities [36]:

- they can **travel**, to move from one place to another
- they can **meet** other agents, which allows them to call one another agent's procedures
- they can create **connections**, to allow an agent to communicate with another agent in a different place
- they have **authority**, which indicates the real-world individual or organization that the agent represents
- they have **permits** to indicate the capabilities of agents

Design of a MAS Architecture

The Computer Science and Mechanical Engineering departments at the University of Calgary have recently assembled a research group to study multi-agent systems. This group, called the Collaborative Agents Group [5], has adopted notions from the models described in the preceding sections to define a dynamic collaborative multi-agent systems architecture. This architecture will be tested by mapping higher-level multi-agent manufacturing architectures, such as MetaMorph [27], into a dynamic collaborative agent environment.

The architecture models open environments composed of logically distributed areas where agents exist. The basic agents in this architecture are minimal agents, local area coordinators, yellow page servers, and cooperation domain servers. These agents are described as:

- **Minimal agents**: They are the abstract common denominator of all agents in the architecture. They implement the basic communicational functionality needed in agent-to-agent interactions.
- **Local area coordinators**: There is one local area coordinator per area. Local area coordinators represent agents in their area and help them initiate agent-to-agent interactions. They also provide a ``white pages'' directory service of agents in an area.
- **Yellow page servers**: They are responsible for storing and making available information about services advertised by agents.
- **Cooperation domain servers**: They provide virtual environments supporting multi-agent communications and information sharing. These environments, called cooperation domains, allow agents to subscribe, exchange messages, and access shared information.

Simply described, a MAS is an environment consisting of areas. Areas are required to have exactly one local area coordinator, which is an agent that acts as a facilitator for other agents within its area. Agents can be identified as being inside an area if they have registered with the area's local area coordinator. Agents will use the services of local area coordinators to access other agents in the system. Agents can advertise services and find out about other agents' services by means of yellow page servers. Agents requiring data sharing with other agents can join virtual environments called cooperation domains, which are supported by cooperation domain server agents.

Conclusions

The multi-agent system paradigm promises to be a valuable software engineering abstraction for the development of computer systems. In addition, the wide adoption of the Internet as an open environment and the increasing popularity of machine-independent programming languages, such as Java, make the widespread adoption of multi-agent technology a feasible goal.

Consequently, it might pay off to invest some time to read and understand key concepts on this area. Most serious readings about agents are compilations of research papers previously published in specialized conferences and workshops. The technical nature of these articles makes it difficult for the uninitiated reader to coherently integrate the information presented to form a high-level framework. Therefore, the motivation for this article was to provide readers with a global perspective on the research literature on multi-agent systems.

In addition, this article briefly introduced some basic notions in the design of a multi-agent systems architecture being developed at the University of Calgary, an effort that promises to become a fruitful contribution for the area of agent research.

Acknowledgments

I would like to thank the Collaborative Agents Group (CAG) at University of Calgary, especially my supervisors (Rob Kremer, Niek Wijngaards and Douglas Norrie) for their invaluable comments, suggestions and support, which helped me to understand better the intricacies of multi-agent research. Partial support for this work was provided by a grant from Smart Technologies, Inc.

References

1

Arnold, K. and Gosling, J. *The Java Programming Language.* Addison-Wesley Publishing Co., second edition. 1998.

2

Bradshaw, J.M. An Introduction to Software Agents. In: *Software Agents*, J.M. Bradshaw (Ed.), Menlo Park, Calif., AAAI Press, 1997, pages 3-46.

3

Bradshaw, J.M., Dutfield, S., Benoit, P. and Woolley, J.D. KAoS: Toward An Industrial-Strength Open Agent Architecture. In: *Software Agents*, J.M. Bradshaw (Ed.), Menlo Park, Calif., AAAI Press, 1997, pages 375-418.

4

Brooks, R. A. *Intelligence Without Reason.* Massachusetts Institute of Technology, Artificial Intelligence Laboratory, A.I. Memo Number 1293, April, 1991.

5

CAG *Collaborative Agents Group.* 1998. Web site: http://sern.ucalgary.ca/cag/ (publications forthcoming)

6

Cohen, P.R., Cheyer, A., Wang, M., and Baeg, S.C. An open agent architecture. In: *Proceedings of the AAAI Spring Symposium*. 1994.

7

Decker, K., Sycara, K. and Williamson, M. Middle-Agents for the Internet. In: *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI-97)*, January, 1997.

8

Decker, K., Williamson, M. and Sycara, K. Matchmaking and Brokering. In: *Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS-96)*, December, 1996.

9

Dennett, D.C. *The Intentional Stance.* MIT Press, Cambridge, Mass. 1987.

10

Durfee, E.H. What Your Computer Really Needs to Know, You Learned in Kindergarten. In: *Proceedings of the Tenth National Conference on Artificial Intelligence*, July 1992, pages 858-864.

11

Durfee, E.H., Lesser, V.R. and Corkill, D.D. Trends in Cooperative Distributed Problem Solving. In: *IEEE Transactions on Knowledge and Data Engineering*, March 1989, KDE-1(1), pages 63-83.

12

FIPA *FIPA 97 Specification, Part 1: Agent Management.* Foundation for Intelligent Physical Agents, Version 1.2, October 10, 1997.

13

FIPA *FIPA 97 Specification, Part 2: Agent Communication Language.* Foundation for Intelligent Physical Agents, Version 1.2, October 10, 1997.

14

Finin, T., Labrou, Y. and Mayfield, J. KQML as an Agent Communication Language. In: *Software Agents*, J.M. Bradshaw (Ed.), Menlo Park, Calif., AAAI Press, 1997, pages 291-316.

15

Gasser, L. Foreword. In: *Readings in Agents*, Huhns, M.N. and Singh, M.P. (Eds.), San Francisco, Calif., Morgan Kaufmann Publishers, 1998, pages v-vi.

16

General Magic *Odyssey*. 1998. Web site: http://www.genmagic.com/technology/odyssey.html

17

Genesereth, M. An Agent-based Framework for Interoperability. In: *Software Agents*, J.M. Bradshaw (Ed.), Menlo Park, Calif., AAAI Press, 1997, pages 317-345.

18

Genesereth, M. and Fikes, R. *Knowledge Interchange Format, Version 3.0 Reference Manual*, Technical Report, Computer Science Department, Stanford University, USA., 1992.

19

Ginsberg, M. The Knowledge Interchange Format: The KIF of Death. In: *AAAI Magazine*, Fall 1991, Volume 12, Issue 3, pages 57-63.

20

Gruber, T.R. A Translation Approach to Portable Ontology Specifications. In: *Proceedings of the Knowledge Acquisition for Knowledge-Based Systems (KAW'93)*, Gaines, B.R. and Musen, M. (Eds.), Banff, Canada, 1993, pages 199-220.

21

Huhns, M.N. and Singh, M.P. Agents and Multi-agent Systems: Themes, Approaches, and Challenges. In: *Readings in Agents*, Huhns, M.N. and Singh, M.P. (Eds.), San Francisco, Calif., Morgan Kaufmann Publishers, 1998, pages 1-23.

22

IBM *Aglets.* 1998. Web page: http://www.trl.ibm.co.jp/aglets/

23

Jennings, N.R., Sycara, K. and Wooldridge, M. A Roadmap of Agent Research and Development. In: *Autonomous Agents and Multi-Agent Systems Journal*, N.R. Jennings, K. Sycara and M. Georgeff (Eds.), Kluwer Academic Publishers, Boston, 1998, Volume 1, Issue 1, pages 7-38.

24

Jennings, N.R. and Wooldridge, M. Intelligent Agents: Theory and Practice. In: *The Knowledge Engineering Review*, 1995, Volume 10, Number 2, pages 115-152.

25

Labrou, Y. and Finin, T. *A Proposal for a new KQML Specification.* TR CS-97-03, Computer Science and Electrical Engineering Department, University of Maryland Baltimore County, Baltimore, February 1997.

26

Mamdani, A. The Social Impact of Software Agents. In: *Proceedings of the Workshop on The Impact of Agents on Communications and Ethics: What do and don't we know?*, Program presentation, Foundation for Intelligent Physical Agents (FIPA), Dublin, July 15, 1998.

27

Maturana, F., Shen, W. and Norrie, D.H. MetaMorph: An Adaptive Agent-Based Architecture for Intelligent Manufacturing. In: *International Journal of Production Research*, 1998. (in press)

28

Nii, H.P. Blackboard Systems. In: *The Handbook of Artificial Intelligence*, A. Barr, P.R. Cohen and E. A. Feingenbaum (Eds.), Addison-Wesley, New York, 1989, Volume IV, chapter XVI, pages 1-82.

29

Nwana, H.S. Software Agents: An Overview. In: *The Knowledge Engineering Review*, October/ November 1996, Volume 11, Number 3, pages 205-244.

30

ObjectSpace *Voyager 2.0, User Guide*. 1998. Web page: http://www.objectspace.com/developers/voyager/

31

Ousterhout J.K. Tcl: An Embedded Command Language. In: *Proceedings of the USENIX Conference*, Winter 1990, pages 133-146.

32

Peng, Y., Finin, T., Labrou, Y., Chu, B., Long, J., Tolone, W.J. and Boughannam, A. A Multi-Agent System for Enterprise Integration. In: *Proceedings of the Third International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agent Technology*, H.S. Nwana and D.T. Ndumu (Eds.), London, UK, March, 1998, pages 155-169.

33

Russell, S.J. and Norvik, P. *Artificial Intelligence: A Modern Approach*, Prentice Hall, Englewood Cliffs, N.J., 1995.

34

Shoham, Y. An Overview on Agent-oriented Programming. In: *Software Agents,* J.M. Bradshaw (Ed.), Menlo Park, Calif., AAAI Press, 1997, pages 271-290.

35

Virdhagriswaran, S., Osisek, D. and O'Connor, P. Standardizing Agent Technology. In: *ACM StandardView*, 1995, Volume 3, Number 3, pages 96-101.

36

White, J.E. Mobile Agents. In: *Software Agents*, J.M. Bradshaw (Ed.), Menlo Park, Calif., AAAI Press, 1997, pages 437-472.

37

Wiederhold, G. Mediators in the Architecture of Future Information Systems. In: *IEEE Computer*, March 1992, pages 38-49.

38

Wiederhold, G. and Genesereth, M. Basis for Mediation. In: *Proceedings of the Third International Conference on Cooperative Information Systems (COOPIS'95)*, Vienna, Austria, May 1995, pages 138-155.

39

Wooldridge, M.J. and Jennings, N.R. Pitfalls of Agent-Oriented Development. In: *Proceedings of the Second International Conference on Autonomous Agents (Agents '98)*, K. P. Sycara and M. Wooldridge (Eds.), ACM Press, May 1998.

Roberto A. Flores-Mendez is a student at the University of Calgary, Canada, where he is pursuing a Ph.D. in Computer Science. His interests include software agents, the design of multi-agent systems, and issues in distributed computing. Some of his previous achievements include a B.Sc. in Computer Systems Engineering from the Instituto Tecnologico y de Estudios Superiores de Monterrey (Mexico) and an M.Sc. in Software Engineering from the University of Calgary (Canada). In 1997, Roberto was awarded a first prize in the ACM/IBM Quest for Java contest for his jKSImapper concept mapping tool. He can be contacted by email at robertof@cpsc.ucalgary.ca