An Agent-Oriented Framework for Constructing Mobile Agent Systems

Marthie Schoeman
Unisa, South-Africa
schoema@unisa.ac.za

Lucas Venter
Unisa, South-Africa
ventelm@unisa.ac.za

Elsabé Cloete

Abstract

Agents and mobile agents hold significant benefits for current trends in computing. Despite some industries actively exploring mobile agent applications the promised deployment has not taken place as expected. Possible incompatibility between the general view of a mobile agent and the diverse definitions of agents has been raised as one of the potential reasons why industry has not yet adopted agent and mobile technology as widely as expected. Developing agent systems requires specialized skills and knowledge in various areas. Accordingly, regardless of a novice mobile agent programmer's computing background, he/she usually has to assimilate new knowledge in order to implement mobile agents. Furthermore, to realise the full potential of mobile agents, they should be implemented according to proper agent programming principles. This paper describes a framework for constructing a mobile agent system that takes agent orientation into account to provide a programming model for novice mobile agent programmers.

1. Introduction

Agents and mobile agents hold significant benefits for current trends in computing such as pervasiveness, distributed systems interconnected via networks, increasing complexity, delegation or automating processes by giving control to computer systems, and human orientation, where computers are increasingly personalized in terms of human characteristics [13; 25; 26; 44]. Both the Semantic Web and Web Services are also important developments.

Several of the trends mentioned are Internet-based applications. Many of them can be implemented by using software agents. For example: agents provide the autonomy and ability to interact that allows *delegation* to remote computer systems or automating processes. Autonomy and interaction also allow *personalizing* computers in terms of human characteristics. Mobile agents' autonomous, asynchronous nature, coupled with the ability to carry their know-how with them, are especially suited to distributed computing, Internet-based computing, ubiquitous (pervasive) and mobile computing applications.

Despite some industries, notably the telecommunications and mobile computing industries, actively exploring mobile agents for a range of applications [1; 9; 27; 35; 36], the debate on the usefulness and viability of mobile agents since the mid-nineties continues [9; 12; 17; 20; 22; 28; 32; 33; 34; 36; 40; 46], and the promised deployment has not taken place. This can be ascribed, among other reasons, to factors such as a lack of a programming model for agent-based applications, lack of re-use, doubts about security, incompatibility between different mobile agent systems and the difficulty in designing and implementing mobile agents. These problems are aggravated by the fact that more than one community are active in the mobile agent field, with differing views on mobile agents and their abilities. The possible incompatibility between the general view of a mobile agent and the diverse definitions of agents [23; 25] has indeed been raised as one of the potential reasons why industry has not yet adopted agent and mobile technology as widely as could be expected [34]. Furthermore, developing agent systems requires specialized skills and knowledge in various areas [2; 3; 6]. Accordingly, no matter what computing background a novice mobile agent programmer may have, (s)he usually has to assimilate new knowledge in order to implement mobile agents. Novices with a distributed systems background, for example, typically lack the artificial intelligence (AI) training to incorporate intelligence in their agents, while novices from an AI background, on the other hand, lack the distributed systems experience. Even when using agent construction toolkits to implement agent systems, mobile agent programmers need to be aware of a substantial number of concepts at various levels, such as at the level of individual agents and the agent system level. Furthermore, to realise the full potential of mobile agents in the programming environment sketched above, they should be implemented according to the proper principles involved in agent programming.

The purpose of this paper is therefore to describe a framework for constructing a mobile agent system that takes agent orientation into account in order to provide a programming model to introduce novices to mobile agent programming. Section 2 sketches the background in terms of agent orientation, mobile agents and mobile agent systems. Section 3 describes the proposed architectural model, section 4 discusses its significance and section 5 provides the conclusion.

2. Background

2.1 Agent orientation

Agent orientation refers to analyzing, designing and implementing complex software systems as a collection of interacting, autonomous agents. These agents exhibit proactive and intelligent behaviour, and interact with one another through high-level protocols and languages [4; 15; 25; 45]. In an agent-oriented software-engineering approach a problem will be decomposed into multiple autonomous agents, embedded in an environment, that can act and interact in flexible ways to achieve their goals [42]. Three key abstractions are employed in the agent-oriented paradigm: agents, interaction and organization [4].

An *agent* is autonomous, continuously perceives its environment, reacts to the perceived information and interacts proactively with its environment as well as with other agents in order to achieve its goals. Agents usually act on behalf of individuals/companies or as part of some wider problem-solving effort. *Interaction* between agents therefore typically occurs in the presence of some *organizational context*, namely in the context of some relationship between the agents concerned, which in turn affects the behaviour of the agents. These relationships have to be represented explicitly in the system. This is done by means of protocols that enable organizational groupings to be formed and disbanded, mechanisms that ensure groupings act together in a coherent fashion, and structures that characterize the macro-behaviour of collectives.

In an agent-oriented view of the world, most problems require or involve multiple agents to represent the decentralized nature of the problem, the multiple loci of control, the multiple perspectives or competing interests. Agents in such a multi-agent system need to interact with one another in order to achieve their individual goals or to manage the complexities of being situated in a common environment. There are two points that qualitatively differentiate agent interactions from those that occur in other software engineering paradigms:

- Agent-oriented interactions occur through a high-level agent communication language (ACL) so that interactions are conducted at the knowledge level, in terms of which goals should be followed at what time, and by whom. Method invocation or function calls in contrast operate at a purely syntactic level.
- As agents are flexible problem solvers, operating in an environment in which they have only partial control and observability, interactions need to be handled in a similarly flexible manner. Agents therefore need the computational apparatus to make context-dependent decisions about the nature and scope of their interactions and to initiate actions that were not foreseen at the time they were designed [15; 16].

Decomposing a system into autonomous agents introduces a tool for conceptual grouping that comes with the well-defined bounds of an agent, since all components that compose the control-flow of a particular agent are grouped as an entity. This makes it easier to identify larger units in the system that belong together semantically [23]. As a result agents can be grouped or organized in order to be managed as a unit and to create various hierarchies in the system. The organization of agents endows a multi-agent system with more than the knowledge, competence and abilities provided by all of its individual agents. Frequently a multi-agent system as a whole, therefore, achieves more than the sum of its component agents' goals. This is termed 'emergence' [15; 45]. For example, in a system composed of multiple autonomous agents, each looking for computational resources to utilize, a global load balancing of the activities can be achieved, without any one agent specifically attempting to manage the load balancing. Consequently the organizational structures, or societies of agents, should be analyzed, designed and implemented as first-class entities in the system [45].

Figure 1 provides an agent-oriented view of a system illustrating some concepts in agent orientation. The system represents a scaled-down version of an examination situation in a school involving a society of teachers, learners and a supervisor. Agents represent each of the teachers, learners and the supervisor. The school itself is the environment in which the agents operate. The teachers (T1 and T2) are organized into a group. Each teacher has an individual goal, namely to mark the examination scripts for the subject he/she is offering. Each teacher also works towards a global goal, namely to determine a final mark for each learner. The manner in which teachers interact with each other, differ from the manner in which they interact with a learner, and thus use different interaction protocols.

Learners (L1 to L5) are organized in groups according to the subjects they are taking. Similar to the group of teachers, individual goals, common goals, and different interaction protocols, etc. can be identified for the learners as well as for the supervisor S, who supervises during the examination.

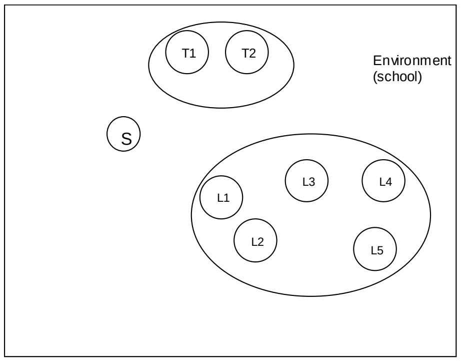

Figure 1. The examination situation in a school represented by a society of agents

2.2. Mobile agents

A mobile agent is considered to be an active entity that can migrate autonomously through a computer network and resume execution at a remote host. Autonomy and mobility are the two most distinctive attributes of a mobile agent.

The concept of a mobile agent developed from advances in distributed computing by two communities, the distributed artificial intelligence (DAI) community and the distributed systems computing community. At present these research coincide in the area of distributed agent computing. The DAI community considers mobility to be an orthogonal property of agents, while the distributed systems community focuses on mobility and considers a mobile agent to be code or an object. Nevertheless, a mobile agent remains essentially an agent with varying degrees of intelligence, autonomy and interaction, depending on the application in which it is applied.

2.3. Essential features in mobile agent systems

A mobile agent system provides the environment mobile agents need to exist and function. This environment consists of agent clients and agent servers. An agent client provides the application programming interface (API) that allows a user to create and control agents. Agent servers provide the runtime environment for agents to execute in as well as several other functions in support of the mobile agent. To achieve functionality mobile agent systems should exhibit the following essential features [37; 39]:

2.3.1. Agent mobility. Not all of the instructions in a mobile agent have to be executed on the same node or even within the same network. Mobility allows the transfer or migration of a mobile agent to another host, as well as the resumption of execution at the new host. Agent migration can be implemented as weak mobility or strong mobility.

2.3.2. Naming services. A global naming scheme and name service are needed to locate resources, specify agent servers for migration and establish inter-agent communication [39], while globally unique agent names are used for identification, controlling and locating agents [29].

2.3.3. Communication. Mobile agents need to communicate with each other, their owners, their hosts and sometimes with other non-agent services and resources, in order to fulfill their social ability.

2.3.4. Access to local resources. Access control to local resources includes controlling the mobile agent's access to application-defined resources and system level resources such as files, I/O devices and network ports, as well as limiting resource consumption, for example CPU time, disk storage, number of threads, network bandwidth, etc. Various mobile agent systems handle this in different ways.

2.3.5. Fault-tolerance/Persistence. While improved fault-tolerance is cited as one of the benefits of mobile agents, this is only true if agent migration itself is fault-tolerant with proper mechanisms for local recovery in place [31].

Various situations, such as breakdown of connections or hosts, destruction of the agent or network errors causing the agent to get lost, can prevent an agent from migrating successfully [14; 41]. In order to ensure a fault-tolerant system, some form of failure detection and recovery has to be provided to ensure persistence.

2.3.6. Interoperability. Interoperability refers to both the interoperability between different mobile agent platforms and integration with existing applications. The two main sets of standards compiled for mobile agents; MASIF from the OMG [29] and the FIPA Agent Management Support for Mobility Specification

[8], attempt to address these issues. Though both groups are currently inactive, and the specifications incomplete, they still offer valuable guidelines for novice mobile agent programmers.

2.3.7. Agent management and control. Agent management and control should allow tracking of an agent in order to control its lifetime and goals, as well as a mechanism for recalling or terminating remote agents.

2.3.8. Security. The two most important problems in this area are mutual protection of the host and the mobile agent from each other. The host needs to be protected against attacks by unauthorised or malicious mobile agent code, while an agent needs to be protected from the host that is executing it [20; 31; 39]. Apart from protecting the agents and hosts from each other, the agent system must also be able to handle many other security considerations, including: control resource consumption, protect agents and the data they collected during transmission, as well as during execution, protect the naming services and provide secure remote communication [24].

3. Proposed framework

In order to establish an architectural basis for the proposed framework, in section 3.1 relevant architectural components are extracted from the available standards for mobile agent systems as specified by MASIF and FIPA, and combined with features and characteristics that are embedded in a number of existing mobile agent systems. In section 3.2 these are integrated with design and development guidelines, in conjunction with agent orientation, to provide a comprehensive model representing the essential features of a mobile agent system.

3.1 Existing models

Four mobile agent platforms were selected to include/assist in establishing this architectural basis, namely SMART (Scalable Mobile and Reliable Technology) [43], D'Agents [10], Grasshopper [30] and Aglets [21]. All are publicly available. Other systems studied include [5; 7; 11; 18; 19; 39].

The essential features in mobile agent systems as listed in section 2.3, the architectural components identified in each of the four platforms indicated above, and the mobile agent standards provided by FIPA and MASIF are compared in Table 1. An x indicates that a particular feature is addressed in the platform or standard. However, some platforms offer features that can apply to more than one of the essential features indicated in section 2.3. These are grouped towards the end of the table and marked with an *. All the essential features indicated in section 2.3 that apply to one or more platforms are given.

As can be seen in Table 1, none of the four mobile agent platforms completely satisfies the features required by a mobile agent system as indicated in section 2.3. Grasshopper's architecture satisfies most requirements, but components are frequently located within several layers of other components. Furthermore, the Grasshopper architecture consists of two layers that are mapped onto each other. This makes it difficult for a novice to form a broad overview, as well as to recognize individual components. The FIPA and MASIF standards also do not satisfy all the requirements listed.

The proposed generic mobile agent system architectural model described below includes all the components listed in Table 1, with the exception of an agent model and explicit reference to interoperability. As mentioned before, interoperability refers to interoperability between different mobile agent platforms and integration with existing applications. As can be seen from the MASIF standard, this covers a number of areas, namely a standard way of managing agents, a common mobility infrastructure to allow agents to communicate and visit other systems, a standardised syntax and semantics for naming services and a standardised location syntax for finding agents. Accordingly, the relevant components in the proposed architectural model handle these aspects. The proposed architectural model thus includes the requirements listed in section 2.3 and all the components available in the platforms examined, as well as those identified by FIPA and MASIF. This provides an extensive overview of the environment in which a mobile agent exists and operates. However, in agent orientation a multi-agent system focuses on addressing the organization and coordination of agents. The proposed architectural model therefore enhances the environment derived from Table 1 by adding a social component. The task of the social component is to control the organization and coordination of agents, as well as to enforce the resulting global social laws and conventions in a multi-agent system.

Though Grasshopper does offer a facility to organize mobile agents in groups by means of places, agencies and regions, no provision is made for explicitly coordinating the mobile agents, as is required in a multi-agent system. SMART uses a region administrator to enforce security policies on a set of agents, which hints at applying global laws, but does not provide a mechanism to organize agents into groups. The proposed architectural model thus portrays

Features required by mobile agent systems	**SMART**	**D'Agents**	**Grasshopper**	**Aglets**	**MASIF**	**FIPA**
Agent client to provide agent management and control	x		x	x	x	
An environment that acts as interface between agent and server	x	x	x	x	x	x
Constrained access to local resources	x	x	x	x	x	
Agent mobility		x	x	x		x
Communication			x			
Interoperability					x	
Fault-tolerance and persistence			x	x		
Security	x	x	x	x	x	
Naming services	x		x	x	x	
*Agent environment / constrained access			x			
*Mobility / Persistence				x	x	
*Life cycle				x		x
*Agent mobility		x	x	x		

Table 1. Features required by mobile agent systems

a complete view of the architectural components required to implement a mobile agent system in agent orientation.

The proposed generic mobile agent system architectural model is depicted in Figure 2. A layered approach, organized according to the life cycle of a mobile agent, is followed. The layered approach isolates the individual components that each forms an essential part of the mobile agent system. In this way, while providing a broad overview of the important system elements during design, the approach also enables developers to focus on the tasks pertaining to a specific layer, which forms a functional unit.

Since the layers in the architectural model are organized according to the life cycle of a mobile agent, in section 3.2 this is used to describe the purpose of each layer in the architectural model.

3.1. Layers in proposed architectural model

At the local host (see Figure 2), an authority (the person or organisation for whom an agent acts) uses the *authority API* to create one or more agents in the *agency*. The agency provides an environment where agents can be created, suspended, activated and destroyed. Such an agent moves through the *execution layer* to the *social layer* where the global social tasks of the agent society are added to the mobile agent. In the *mobility* and *communication aspects* inside the social layer, standard mechanisms to ensure mobility and effective agent communication are added. Fault tolerance mechanisms are added at the *persistence aspect* in the social layer. At the *mobile agent management services layer*, the agent is serialised, before being encrypted at the *agent security layer* for protection against malicious hosts and during transfer. At the *naming server layer*, the agent is named and registered for future reference. The location of hosts to be visited will also be determined at the naming server layer. The agent receives credentials for access to other hosts at the *host security layer* before it is encoded in a suitable transport protocol at the *network layer* for transport through the network. On arrival at a remote host, the network layer removes the transport protocol envelope and sends the agent to the *host security layer* where the agent seeks host access by submitting its credentials. If it is accepted, the host server registers the agent at the *naming server layer*, before the agent is decrypted at the *agent security layer*. The *mobile agent management services layer* is responsible for deserialising the agent and performing any conversions that may be necessary. The agent moves through the *social layer* to be executed at a place in the *execution layer*. The execution layer provides constrained execution environments (places) where agents can execute and meet other agents. The social layer enforces the social laws in the agent society during interaction between agents, the environment and other entities such as Web Services, in order to allow coordination, cooperation and negotiation. During execution, the agent will interact with the various aspects within the social layer: with the *persistence aspect* for storing information, with the *communication aspect* for communicating with other agents or entities, and with the *mobility aspect* for future migration requests. Interaction with the persistence aspect, the communication aspect and the mobility aspect are all subject to the social laws enforced by the social layer.

Once executed, the agent follows the path downwards through the architecture in a similar

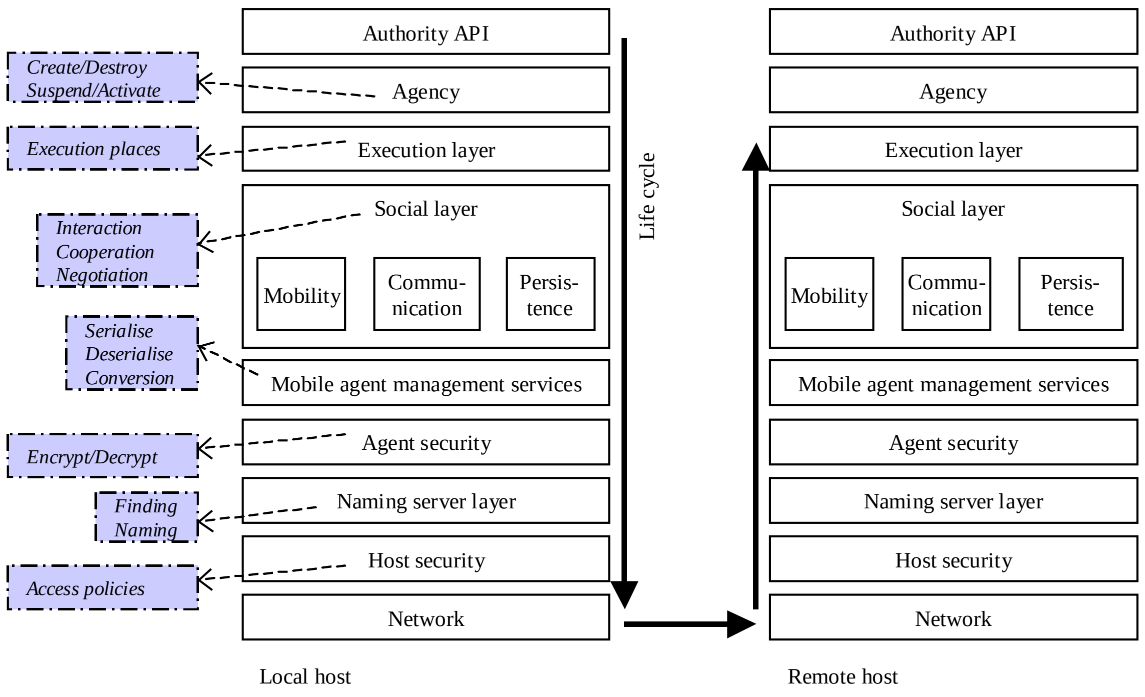

Figure 2. Proposed architectural model for mobile agent system development

fashion as at the local host. On arrival back at the agency in the local host, the owner of the mobile agent may use the authority API to dispose of the agent.

4. Significance of the proposed model

The proposed generic mobile agent system architectural model provides a comprehensive view of the architectural components required to implement a mobile agent system in agent orientation. This is a revision of a previous version of the model that did not include the social layer [38]. The social layer represents the social characteristics required by multi-agent systems.

Though most of the platforms used to identify the essential architectural components follow a layered approach, none of them includes all the layers in the proposed architectural model. Grasshopper is the most comprehensive of the platforms investigated. Its architecture though contains two aspects that make it difficult for a novice to comprehend. It consists of two layers, which appear to be mapped upon each other. Within these two layers, components are layered within several other components. The architectural components in the proposed architectural model, in contrast, represent the essential features of a mobile agent system in a simplified, general but clear way. A clear understanding of the architectural components that are involved makes it much easier for a novice to form a broad overview of the important system elements during design. It also enables programmers to identify and include different mobile agent characteristics, with maximum reuse of available technologies and architectures.

The layered approach isolates the individual architectural components that each form an essential part of the mobile agent system. This enables developers to focus on the tasks that apply to a specific layer during its development, since each layer forms a functional unit. A layered model also allows the independent development of layers while implementing each layer separately and supports program maintenance, debugging and upgrading [5; 19; 38].

Furthermore, the social layer, in representing the social character of a multi-agent system, reminds the novice to implement mobile agent systems according to the proper principles in agent orientation in order to realize the full potential of such systems.

The proposed architectural model thus provides a constructive tool to both programmers and researchers who enter the field for the first time. The architectural model could act as a first step on the way to establishing a standard that can be used during the design and development of mobile agent systems. It also provides a point of reference for researchers and developers to evaluate the capabilities of mobile agent systems created with commercial tool kits.

5. Conclusion

This paper proposed a generic mobile agent system architectural model to guide novice mobile agent programmers in developing a mobile agent system. The model provides a comprehensive overview of the different architectural components for a mobile agent system in the agent orientation paradigm. The proposed architectural model includes more layers than any of the platforms used as a basis to identify components, and thus addresses more aspects and provides more assistance to novices.

6. References

[1] *2004 IEEE International Conference on Mobile Data Management (MDM'04)*. Available at: http://www.cs.duke.edu/mdm2004/. [Accessed 25/5/2005].

[2] *AgentBuilder*.2004. Available at: http://www.agentbuilder.com/Documentation/whyAgents.html. [Accessed 2/11/2004].

[3] Badjonski, M., Ivanovic, M. & Budimac, Z. 2005. Adaptable Java Agents (AJA) - A Tool for Programming of Multi-Agent Systems. *ACM SIGPLAN Notices*. vol. 40. no. 2: pp.17-26.

[4] Burmeister, B. 1996. Models and Methodology for Agent-Oriented Analysis and Design. in K.Fischer, (ed.) *Working Notes of the KI '96 Workshop on Agent-Orientated Programming and Distributed Systems*. DFKI.

[5] Dale, J. 1997. *A Mobile Agent Architecture for Distributed Information Management*. Ph D. University of Southampton. Southampton.

[6] Dastani, M. & Gomez-Sanz, J. J.2005. *Programming Multi-Agent Systems A report of the technical forum meeting.* Available at: http://www.cs.uu.nl/%7Emehdi/tfg/al3files/report.pdf . [Accessed 3/8/2005].

[7] Feridun, M. & Krause, J. 2001. A framework for distributed management with mobile components. *Computer networks*. vol. 35. pp.25-38.

[8] *FIPA00087 - FIPA 98 Part 11 Version 1.0: Agent Management Support for Mobility Specification*.1998. Available at: http://www.fipa.org/specs/fipa00087/. [Accessed 23/5/2005].

[9] Gray, R. S. 2004. Mobile Agents: Overcoming Early Hype and a Bad Name. in *2004 IEEE International Conference on Mobile Data Management (MDM'04)*. Los Alamitos, Calif., : IEEE Computer Society. Berkeley, California. pp. 302-303.

[10] Gray, R. S., Cybenko, G., Kotz, D., Peterson, R. A. & Rus, D. 2002. D'Agents: Applications and performance of a mobile agent system. *SOFTWARE: PRACTICE AND EXPERIENCE*. vol. 35. no. 6: pp.543-573.

[11] Gschwind, T., Feridun, M. & Pleisch, S. 1999. ADK - Building Mobile Agents for Network and Systems Management from Reuseable Components. *Proceedings of First International Symposium on Agent Systems and Applications*. pp.13-21.

[12] Harrison, C. G., Chess, D. M. & Kershenbaum, A. 1995. *Mobile Agents: Are they a good idea?* IBM Research Report. IBM T. J. Watson Research Center.

[13] Hayes-Roth, R. & Amor, D. 2003. *Radical Simplicity transforming Computers Into Me-Centric Appliances*. 1st edn. Prentice-Hall, Inc. New Jersey.

[14] Horvat, H., Cvetkovic, D., Milutinovic , D., Kocovic, P. & Kovacevic , V. 2000. Mobile Agents and Java Mobile Agent Toolkits. in *Proceedings of the 33rd Hawaii International Conference on System Sciences*. IEEE Computing Society, Los Alamos, CA. Maui, Hawai'i, USA. p. 10.

[15] Jennings, N. R. 1999. Agent-Based Computing: Promise and Perils. in *16th Joint Conference on Artificial Intelligence (IJCAI-99)*. pp. 1429-1436.

[16] Jennings, N. R. 2001. An Agent-Based Approach for Building Complex Software Systems. *Communcations of the ACM*. vol. 44. no. 4: pp.35-39.

[17] Johansen, D. 2004. Mobile Agents: Right Concept, Wrong Approach. in *2004 IEEE International Conference on Mobile Data Management (MDM'04)*. Los Alamitos, Calif., : IEEE Computer Society. Berkeley, California. pp. 300-301.

[18] Johansen, D., Lauvset, K. J. & Marzullo, K. 2002. An extensible software architecture for Mobile Components. in *Ninth Annual IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (EECBS'02)*. pp. 231- 237.

[19] Kendall, E. A., Krishna, P. V., Suresh, C. B. & Pathak, C. V. 2000. An Application Framework for Intelligent and Mobile Agents. *ACM Computing Surveys*. vol. 32. no. 1es.

[20] Kotz, D., Gray, R. & Rus, D.2002. *Future Directions for Mobile Agent Research.* Available at: http://dsonline.computer.org/0208/f/kot.htm. [Accessed 15/10/2002].

[21] Lange, D. B.1997. *Java Aglet Application Programming Interface White Paper - Draft 2.* Available at: http://www.trl.ibm.co.jp/aglets. [Accessed 12/10/2005].

[22] Lange, D. B. & Oshima, M. 1999. Seven Good Reasons for Mobile Agents. *Communications of the ACM*. vol. 42. no. 3: pp.88-89.
[23] Lind, J. 2001. Issues in Agent-Oriented Software Engineering. in P. Ciancarini & M. J. Wooldridge, (eds.). *Agent-Oriented Software Engineering - Proceedings of 1st International Workshop AOSE-2000*. vol. 1957. Springer-Verlag. Berlin. pp. 45-58.
[24] Lingnau, A., Drobnik, O. & Dömel, P.1995. *An HTTP-Based Infrastructure for Mobile Agents.* Available at: http://citeseer.nj.nec.com/lingnau95httpbased.html. [Accessed 9/03/2000].
[25] Luck, M., Ashri, R. & d'Inverno, M. 2004. *Agent-based Software Development*. 1st edn. Artech House. Norwood MA.
[26] Luck, M., McBurney, P., Shehory, O., Willmott, S. & Community, A.2005. *Agent Technology Roadmap.* Available at: http://www.agentlink.org. [Accessed 17/8/2005].
[27] Manvi, S. S. & Venkataram, P. 2004. Applications of agent technology in communications: a review. *Computer Communications*. vol. 27. pp.1493-1508.
[28] Milojicic, D. 1999. Trend Wars Mobile Agent Applications. *IEEE Concurrency*. vol. 7. no. 3: pp.80-90.
[29] Milojicic, D., Breugst, M., Busse, I., Campbell, J., Covaci, S., Friedman, B., Kosaka, K., Lange, D., Ono, K., Oshima, M., Tham, C., Virdhagriswaran, S. & White, J. 1998. MASIF: The OMG Mobile Agent System Interoperability Facility. *Personal Technologies*. vol. 2. no. 2: pp.117-128.
[30] Pavlou, G.2000. *The Grasshopper Mobile Agent Platform.* Available at: http://www.ee.surrey.ac.uk/CSSR/ACTS/Miami/grasshopper.html. [Accessed 12/10/2005].
[31] Picco, G. P. 2001. Mobile agents: an introduction. *Microprocessors and Microsystems*. vol. 2. no. 2: pp.65-74.
[32] Roth, V. 2004. Obstacles to the Adoption of Mobile Agents. in *Proceedings of the 2004 IEEE International Conference on Mobile Data Management (MDM'04)*. Los Alamitos, Calif.,: IEEE Computer Society. Berkeley, California. pp. 296-297.
[33] Roth, V., Braun, P. & Rossak, W. 2002. Conference Session and Workshop on Performance, Interoperability and Applications of Mobile Agent Systems. in *Ninth Annual IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS'02).*
[34] Samaras, G. 2004. Mobile Agents: What about Them? Did They Deliver what They Promised? Are They Here to Stay? in *Proceedings of the 2004 IEEE International Conference on Mobile Data Management (MDM'04)*. Los Alamitos, Calif.: IEEE Computer Society. Berkeley, Calif. pp. 294-295.
[35] Satoh, I. 2004. Linking Physical Worlds to Logical Worlds with Mobile Agents. in *Proceedings of the 2004 IEEE International Conference on Mobile Data Management (MDM'04)*. pp. 332-345.
[36] Satoh, I. 2005. Mobile Applications in Ubiquitous Computing Environments. *IEICE Transactions on Communications*. vol. E88B. no. 3: pp.1026-1033.
[37] Schoeman, M. 2005. *An Approach to Facilitating the Training of Mobile Agent Programmers and Encouraging the Progression to an Agent-Oriented Paradigm*. MSc. Unisa. Pretoria.
[38] Schoeman, M. & Cloete, E. 2004. Design Concepts for Mobile Agents. *South African Computer Journal*. no. 32: pp.13-24.
[39] Tripathi, A. R., Ahmed, T. & Karnik, N. M. 2001. Experiences and future challenges in mobile agent programming. *Microprocessors and Microsystems*. vol. 25. pp.121-129.
[40] Vigna, G. 2004. Mobile Agents: Ten Reasons For Failure. in *2004 IEEE International Conference on Mobile Data Management (MDM'04)*. Los Alamitos, Calif.,: IEEE Computer Society. Berkeley, California.
[41] Vogler, H., Kunkelmann, T. & Moschgath, M. L. 1997. Distributed Transaction Processing as a Reliability Concept for Mobile Agents. *Proceedings of the 6th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS '97).*
[42] Weyns, D., Helleboogh, A., Steegmans, E., Wolf, T. D., Mertens, K., Boucke, N. & Holvoet, T.2004. *Agents are not part of the problem, agents can solve the problem.* Available at: http://www.open.org.au/Conferences/oopsla2004/PaperAO/. [Accessed 20/6/2005].
[43] Wong, J., Helmer, G., Naganathan, V., Polavarapu, S., Honovar, V. & Miller, L. 2001. SMART mobile agent facility. *The Journal of Systems and Software*. vol. 56. pp.9-22.
[44] Wooldridge, M. 2002. *An Introduction to Multiagent Systems*. 1st edn. John Wiley & Sons Ltd. Chichester.
[45] Zambonelli, F., Jennings, N. R., Omicini, A. & Wooldridge, M. 2001. Agent-Oriented Software Engineering for Internet Applications. in A. Omicini, (ed.) *Coordination of Internet Agents*. Springer-Verlag.
[46] Zaslavsky, A. 2004. Mobile Agents: Can They Assist with Context Awareness? in *2004 IEEE International Conference on Mobile Data Management (MDM'04)*. Los Alamitos, Calif.,: IEEE Computer Society. Berkeley, California. pp. 304-305.