The Mobile Agents Paradigm

Roberto Silveira Silva Filho
Department of Information and Computer Science
University of California, Irvine
rsilvafi@ics.uci.edu

Abstract

A mobile agent is an object that migrates through many nodes of a heterogeneous network of computers, under its own control, in order to perform tasks using resources of these nodes. The use of this technology represents a change in the distributed programming paradigm. This approach have provides many benefits to the development of distributed applications but introduce new requirements to the engineering of these systems. This paper presents this paradigms with its issues and benefits, discussing its use in the development of distributed applications.

Keywords*: Mobile software agents, Distributed Systems Development, Software Engineering.*

1. Introduction

The development of distributed applications is directly influenced by the choice of an architecture style or paradigm. The requirements of the system as scalability, fault tolerance, response time, support for disconnected operations and so on, are important point to measured and reasoned before the implementation of a system.

This paper gives an overview of the mobile agent paradigm, focusing on the benefits of using this approach in the development of distributed applications. It presents the qualities of this approach with its weaknesses, strengths and requirements. This information can be used in the decision process of adoption of this or other approaches during the specification of a distributed system.

2. Mobile Agents Paradigm

According to Gray et al. [GKRNC96], A **software agent** is "a program that is autonomous enough to act independently, even when the user or application that launched it is not available to provide guidance and handle errors". In another definition, using general terms, a software agent is program that acts in behalf of its owner (agent owner) [GHNCSE97].

A **mobile software agent**, or a software agent for now on, is an object that migrates through many nodes of a heterogeneous network of computers, under its own control, in order to perform tasks using resources of these nodes [IH99; RGK97]. It travels from node to node of a distributed system performing tasks in behalf of its owner. At the end of this process, an agent can return to its home site and report itself to the user who injected this object in the distributed system [KT98].

Mobile agents are used in the development of distributed applications. This paradigm differs form the traditional client/server approach in the following way.

In the client-server paradigm, resource owners (servers) are physically distant of their clients (users). The communication among these parts occurs through a network of computers, being mediated by mechanisms as remote procedure calls, message exchange, sockets and so on. In this paradigm, the reliability of the communication links and the synchronicity of the remote procedure calls are important requirements of the majority of such applications.

On the other hand, in the mobile agent paradigm, the agents migrate to interact locally, at the same host as the resources. By the moving of location, agents can dynamically change the interaction qual-

ity, reducing these costs [CPV97]. Some examples of applications that use the mobile agents paradigm are: The deployment and update of distributed applications, the customization of services, the support for applications evolving mobility, the implementation of fault tolerance polices, workflow management systems, and so on.

3. When to Use the Mobile Agent Paradigm

There are many advantages of the mobile agent paradigm. Some of them include the reduction of the network bandwidth use, distribution of processing and loading through the hosts of the network, support for a more flexible peer-to-peer model, scalability and decentralization of control.

In terms of processing and network bandwidth consumption, the use of the mobile agent paradigm is justified when the cost of the use of some remote resource, using traditional approaches as the client/server paradigm, overcomes the use of the agent.

Mobile agents can be used to overcome the network latency. Consider a distributed environments composed by a big number of machines connected by a slow network, for example, a LAN connected to another LAN using a slow Internet link. Suppose a client on one LAN wants to make a complex query in a Database server in the other LAN. In the mobile agent paradigm, agents can move to the place where the data is stored, realizing queries and filtering relevant information before sending this data to the client. In this context, it is shipper to transport a small agent to the source of the data, than to bring the entire query results back to the node in order to be processed.

In general, during the project of software architectures for distributed systems, the interaction among components is defined in a location independent form. The CORBA middleware [CORBA98], for example, allows the abstraction component's location. In this distributed communication framework, there is no distinction between local or remote interactions. The mobile agent paradigm suggests, though, a new approach to the project and specification of distributed system. This paradigm is usually necessary in cases where the location and mobility of the application need to be considered due to reliability and performance requirements. In problems using the mobile agent paradigm, these requirements are so important, that they affect the conceptual structure of the application in the design phase.

Compared to a client-server centralized system, the use of mobile agents carrying their own data does not reduce the overall traffic of data in the network. In both cases, data or part of the data must be copied locally, in the client hosts. The use of a decentralized model using mobile agents, however, distributes the data traffic over the local network, unloading the central server backbone. The traffic is not client-server centric but peer-to-peer centric. The decentralization of data and control also distributes the server processing and communication among client hosts [SWME00]. Moreover, as the interaction between the agent and the resource (after moving) is performed in the same host, without the transmission of messages through the network, this paradigm is indicated for some kinds of real-time distributed applications.

4. Applications

The literature [CHK94; KT98, RGK97, GKNRC96, LO99] describes many applications that can benefit from the use of the mobile agent paradigm. These are mobile computing, fault tolerance, load balancing, workflow management and electronic commerce. Additionally, new applications as runtime software change and software deployment can also benefit from this technology. Some of these applications, with some examples, are listed as follows.

4.1.1. Mobile computing

In applications evolving mobile devices, the presence of a network connection is intermittent, or has variable and low bandwidth rates [RGK97]. In this context, the independence and autonomy of the mobile agents can be used. Applications can be written as mobile agents that migrate to mobile hosts, perform their activities and move out when the network connection allows to. A mobile agent is independent from his origin. The user or host that lunched the agent in the network does not need to keep a live connection to this object during his lifecycle. The mobile agents are also self-sufficient carrying its code and execution state as it moves. In some cases,

it can also be configured to carry all the application data, which makes it independent form network file system connections, for example.

4.1.2. Fault tolerance and Load Balancing

Tasks and processes in distributed applications can be split in small sub-processes in order to perform their goal. These subtasks can be configured to move form host to host in order to distribute processing load, or also be duplicated (or forked) providing fault tolerance. The agent can operate in the host independently form network connection, allowing temporary absence of it. Notifications are sent to the agent owner in an asynchronous way. In the occasion of a network failure, the agent can wait until the connection is reestablished to migrate or send data back to its owner.

4.1.3. Electronic Commerce

Mobile agents, acting as customers, can be configured to move through different nodes from a network in order to perform commercial transactions on behalf of its owner. In a virtual shopping center scenario, stores offer products with different models and prices. Agents represent the user needs and interests, being equipped with a buying list. The agents can search for some kind of product or service, compare its prices and perform purchases and orders on behalf of its owner.

4.1.4. Distributed System Management

In a distributed system management application, mobile agents can move through hosts in a network, collecting management data (passive management) or reconfiguring nodes in order to implement different management polices (active management), perform specific tasks and apply configurations. A discussion of the potential uses of mobile agents in network management is presented by Bieszczad et al. [BPW98].

4.1.5. Software Deployment

The use of mobile agent paradigm in configuration management, in special, software deployment, is a new field of study. An example of use of this paradigm in software deployment is described by Hall et al. [HHHW97], in the Software Dock system. This application use mobile agents to coordinate the software update process of hosts in the Internet.

4.1.6. Workflow Management System

Workflows are computer interpretable description of activities (or tasks), and their execution order. Workflow Management Systems (WFMS) are used to automate and coordinate the execution of bureaucratic tasks. Tasks can be performed concurrently by many users and automated applications. These tasks can be modeled as autonomous agents that move through the network nodes, carrying the data and controlling the execution of the activities in a WFMS. One example of such approach is the WONDER (Workflow ON Distributed EnviRonment) architecture [SWME00]. This architecture defines a WFMS that addresses, in special, the scalability and availability issues. The architecture is based on the mobile agent paradigm. the case is represented as a mobile agent that migrates from user host to user host, following the process definition. The case (instance of a process described by a workflow) is implemented as a mobile In the WONDER architecture, the control, the storage of data, and the execution of the activities are all distributed over the hosts of an enterprise computer network.

4.1.7. Runtime Change of Software

Software systems can be specially specified and configured to be changed at runtime [OMT98]. In this context, software agents can be deployed conveying updates of modules and software configurations. Its intrinsic capability of conveying data and their ability to execute operations in the current machine can be used to control and coordinate the process of stopping, modifying, and updating a system at runtime.

5. Mobile Agents Systems

A Mobile Agent System (MAS), or Agency, is a computational framework that implements the mobile agent paradigm. It provides services and primitives that help in the use, implementation and execu-

tion of systems developed using the mobile agents paradigm.

This generic framework allows the developers to focus on the logic of the application being implemented, instead of focusing on the implementation details of the mobile agent system.

In order to host he mobile agents, each involved in the distributed application must provide a basic support environment. This environment, called Agency, supports the creation, activation, deactivation and management of agents, which include mechanisms to help in the migration, communication, persistence, failure recovery, management, creation and finalization of agents. Additional services as naming and object persistence can also be provided. This environment must also be safe, in order to protect the resources of the machine from malicious attacks and possible bugs in the implementation of the agent code.

The General Magic MAS, developed together with the Telescript language [White94] in the early 1990, was the first commercial system specially designed to support the development of mobile agents paradigm applications. This system was followed by many others as Tacoma [JRS95] and Agent Tcl [Gray96], in which the agents are described in proprietary script languages. The advent of the Java programming language [Flanagan99], with its support for object serialization and mobile code (applets), fostered the development of new MASs. The IBM Aglets [KLO97], the ObjectSpace Voyager [ObjectSpace97], the Concordia [Concordia97] and the Ajanta [KT98] are some examples. A comparison among these systems is described in [KT98].

The first MAS generation implemented their own migration protocols and mechanisms. The new MAS middleware based on Java, however, use the passage of objects by value, a facility provided by the Java RMI (Remove Method Invocation) API.

Due to the mobility requirement, and the necessity to execute in different operating systems and hardware architectures, the MAS and the agent are generally implemented using interpretable programming languages. The Voyager MAS, for example, uses Java.

MASs support the development of mobile agents implemented in their specific for programming languages as TCL, Java and Telescript. Some MAS as Aglets and Voyager are compatible with the IIOP protocol from OMG (Object Management Coalition) OMA (Object Management Architecture). They allow the communication of agents written to these systems, with CORBA servers. The integration of these systems with CORBA occurs only in this level.

The use of such protocols improves the maintainability and extensibility of the software, which can intemperate and be integrated with non-agent enabled applications more easily.

5.1. Requirements of MAS

In order to facilitate the development of mobile agents distributed applications, and to overcome some problems that arise from this approach, some requirements must be addressed. Systems that support the use of the mobile agent paradigm have to provide a basic set of services and characteristics as follows.

5.1.1. Transportability

A mobile agent must be able to move itself, under its own decision, from one machine to another in a heterogeneous network. It is, the program must be able to suspend its execution in a node, move itself to another node and start its execution from the point it stopped, using its own resources. This migration must happen in an independently of the different hardware or software platform that may compose the network. The transportation of the agent (or its state and code), from one node to the other must be helped by external entities as message services, middleware, or e-mail servers. This is the basic requirement provided by mobile agent middleware as the Object Space Voyager [ObjectSpace97], or IBM Aglets [KLO97].

The migration process of an agent can be implemented in two manners. The agent can create another copy of itself (fork), and follow executing in a different node of the system in an independent way or can suspend its execution and move itself to another node in which the execution is restarted.

Most of the implementation of this mechanism in the literature uses the weak migration. In the weak migration, the agent moves only its execution state. As a consequence of this approach, the code of the agent can come from another site, or be replicated in the sites in which the agent can execute [IH99].

As mobile agents are autonomous, their migration occurs under its own command. Due to this charac-

teristic, more advanced mechanisms, hat allow the capture of the execution state using a fine grain granularity, it is, storing the state of the execution thread stack, are usually not necessary [KT98]. This last approach is known as Strong migration. Telescript is an example of a language that implements this kind of migration [CPV97].

In a third approach, the agent does not carry its own code but only a reference to a code base from which it can be copied on demand.

The choose of one of these approaches depends on the ability of the mobile agent language in dealing with the agent and code state. The selection of one of this approaches is though, will depend on the requirements of the application being built, in special, the programming language used.

5.1.2. Autonomy

The agent must be able to decide where and when to migrate during the accomplishment of its mission. This move can be performed in a reactive way. Some applications may require that this decision be based on dynamic parameters and performance information of the distributed system, as the example of management applications. Other applications as workflow management systems, may require the following of a pre-established plan (not static sequence of resources/nodes that can/have to be visited, together with tasks to be accomplished in each host).

To cope with the agent autonomy characteristic, the communication between the agent and its home site must be avoided. In order to do so, the agent must use resources and mechanisms that allow the decision making related to the migration of the agent. These mechanisms are usually provided by sensors that allow the agent to collect data from its environment, as well as plan interpreters, algorithms and other mechanisms internal to the agents, that provide some degree of autonomy to this object.

5.1.3. Navigability

In order to support the decision making process of the agent (where and when to migrate), the objects must have the knowledge of its objectives and plans, as well as parameters related to its environment. This knowledge of the environment may vary according to the application being implemented. In some cases, the agent can be helped by external services, as the example of traders, naming services, yellow pages and so on.

5.1.4. Security

In a local network, completely isolated, located uniquely in a single organization, it is possible to trust in all the hosts and in the software installed in this distributed system [KT98]. In these systems, agents can freely migrate among hosts. For applications that do not need to communicate with the exterior world, executing in this isolated network, the security is not a big issue.

In applications executing in open networks as the Internet, however, agents can belong to different administrative domains, which cannot be trustable. This characteristic introduces two main problems: agents must be protected from “malicious” hosts; and hosts must be protected from malicious agents, or viruses. An example of the bad use of the mobile agent paradigm was the “I LOVE YOU” virus, that infected thousands of computers in May 2000 [Freedman00; IloveYou00]. In order to deal with these security problems, a MAS must provide the following security mechanisms:

A) Privacy and integrity.

Agents carry their state and data. These data can have sensitive information. For example, in the WONDER distributed workflow architecture [SWME00], a purchase process can carry forms having data of the clients and contracts. The privacy of these data should be ensured. Moreover, nodes of a distributed system may not be equally trustworthy. These matters must considered by the mobile agent developer. Agents have to be programmed in order to apply different levels of access to the information they convey, according to the level of confidentiality of the host.

The MAS must also provide support for the detection of attacks, as the change in the source binary code or the data conveyed by the agents, for example. A way to protect this data is use some cryptography and algorithms that check the integrity of data as CRC (Cyclic Redundancy Check).

B) Authentication of agents and servers.

A MAS have to prevent malicious agents from being confounded as authorized application agents. Is must also avoid that malicious hosts receive authorized agents from the system.

Mechanisms that allow the identification and certification of the server, as well as the agent or the user that the agent represents, have to be supported. This characteristic is usually provided by digital signature schemas (public and private keys for example). These schemas are usually supported by authentication servers that validate the clients.

C) Authorization and access control.

The access for some resources of the system must be restricted/limited in a MAS. Agents can, for example, be configured to respect polices of quotas of occupation in the disk and can have the limited access of write to disks or to create connections in the network. This polices are used in virtual machines like the JVM (Java Virtual Machine). These polices are usually implemented using the access control lists and capabilities (tokens that give to their holders the ability to access a resource).

D) Auditing and Metrics

Agents consume resources as network bandwidth, disk space and CPU during its life. These resources have to be monitored in a way to provide information to the agents and to the administrators of the distributed system.

5.1.5. Fault Tolerance

Agents can execute over many nodes of the distributed system, migrating through many machines, resources and not reliable network connections. The shift from the client-server paradigm to the mobile agent peer-to-peer approach introduces many points of failure in the distributed system. The MAS must provide resources to the agent programmers in order help them in the detection of hardware and software errors. Once an error is detected, the agent can perform the necessary procedures to overcome these errors as, for example, notify the other agents about the failures, move to an alternative resource, wait until a resource become active again, and so on.

5.1.6. Performance

The moving process of an agent must be efficient, in a way to compensate its use, when compared to other paradigms as the client/server.

According to the requirements of the application being developed, the agent need to be small, allowing its fast transfer between nodes of a network. The agent also may have to be able to execute in machine with possible memory and processing restrictions, as the example of mobile computers and handheld computers.

5.1.7. Multi-platform support

The distributed systems in big organizations are usually composed by an heterogeneous set of hardware and software platforms. The ability to execute in these systems usually require that the agents be able no migrate and execute in different operating systems and computer architectures. The SAM must support programming languages that can intemperate and execute in different platforms.

5.1.8. Adaptability

The agent must be sensible to the diverse traffic conditions, connection and topologies of a computer network, as well as to the diversity of resources available in each node. The MAS can provide this information to the agents. This information is usually used in the decision making process related to the migration, fault tolerance, operation mode (connected/disconnected) of the agent and so on. This resource is helped by the use of information sensors.

5.1.9. Communication

The ability to communicate in a localization independent way is another characteristic that must be provided. Agents constantly migrating and do not have a fix address in the network. For such, agents usually need location and tracking mechanisms. Some examples of such services are the message services, forwarding or the actualization of the name service. The communication can be done in an asynchronous way (based on datagrams), synchronously, using remote procedure calls, or using shared files or

resources. The MAS have to cope with this requirement. In some cases, group communication primitives can also be provided.

6. Developing With Mobile Agent Paradigm

In order to get the benefits of the mobile agent paradigm, without incurring in the main problems of this model, the use of a MAS is essential. In the present time, a software engineer can use one of the many MASs available both in the market and in the research projects.

The most important requirements presented in session 5, are implemented in the current mainstream MASs. For example, the Voyager 4.0 [VoyagerRef] implementation support communication among agents using publish/subscribe events, CORBA IIOP and RMI protocols, agent persistence, authentication and authorization of agents, management tools, secure communication and so on.

These systems are built on top of widely used middlewares as CORBA and RMI, providing additional services and facilities to these communication frameworks.

In special the OMG (Object Management Group) a consortium responsible for the standardization of CORBA, also provided his own set of extensions to the OMA architecture. As part of the Common Facilities of the OMA architecture, the OMG defines a Facility for Mobile Agents Systems Interoperability [OMG-MASIF98]. Its main goal is to define a common framework, based on the CORBA middleware, to the interoperability of the MASs. This specification is very generic and does not consider mobile agents as first-class CORBA objects. Moreover, it does not define mechanisms to transport these agents through the ORB. This last requirement is addressed, however, in another OMG specification, the Object by Value RFP [OMG-OBV96]. This specification defines a generic mechanism that allows the implementation of a MAS using CORBA. This facility is available in the CORBA 3.0 standard [Vinoski 98].

7. Discussion

In this paper there were presented the mobile agents paradigm, highlighting its main characteristics and benefits to the development of distributed applications. Some requirements and issues that this paradigm introduces to the distributed applications were pointed. The MAS were presented as a middleware to provide these requirements and make the use of these applications easier. Some examples of the user of this technology were also presented, some of them as workflow management system, runtime change and software deployment are still not very well solved problems, that can benefit for the use of this approach.

There is no “killer” application that can only be implemented with this paradigm, however, there are many benefits in the adoption of this approach, specially in the development of distributed and decentralized applications.

In this sense, Harrison et al. [CHK94] argues that the ability to migrate through distributed systems hosts provide many benefits to the applications that use this paradigm. Among them we can list:

- Local agent-host interaction, reducing the bandwidth use of the network;
- Support for thin clients, with short computational power, or with scarce resources;
- Parallel processing though the distribution of the control and processing.
- Facility to implement semantic routing, as the example of workflow applications;
- Support for scalable applications ; and
- Improvement of fault tolerance to network link failures.

On the other hand, the mobile agent paradigm has some disadvantages, which introduces some extra requirements to the applications that use this approach, as follows:

- The need for secure execution environments, with more severe access restrictions, in order to prevent malicious agents detection (virus);
- Performance limitations due to the use of security polices and interpreted languages;
- The communication and processing overhead associated to the migration of the agents.
- The introduction of many points of failure

However, if considered all positive and negative points of this approach, the mobile agent paradigm

provides an open and generic framework for distributed application development. Even though none of these characteristics is exclusive from the mobile agent paradigm, these aggregate set of benefits are hardly implemented alone by other paradigms as the client-server.

8. References

[BPW98] A. Bieszczad, B. Pagurek, and T. White. Mobile agents for network management. IEEE Communications Surveys, September 1998.

[CHK94] D. Chess, C. Harrison, and A. Kershenbaum. Mobile Agents: are they a good idea?. IBM Research Report, IBM T. J. Watson Research Center, Yourktown Heights, N.Y. RC 19887, December 1994. http://www.research.ibm.com/massdist/mobag.ps

[Concordia97] Mitsubishi Electric. Concordia: An Infrastructure for Collaborating Mobile Agents. In Proceedings of the 1st International Workshop on Mobile Agents (MA '97), April 1997.

[CORBA98] The Common Object Request Broker: Architecture and Specification. Object Management Group, Framingham, MA, 1998.

[CPV97] A. Carzaniga, G. P. Picco and G. Vigna. Designing distributed applications with mobile code paradigms. Proceedings of the 1997 international conference on Software engineering, 1997, pp. 22 – 32.

[DLMM93] M. Day, B. Liskov, U. Maheshwari and A. C. Myers. References to remote mobile objects in Thor. ACM Letters on Programming Languages and Systems 2, 1-4 (Mar. 1993), pp. 115- 126.

[Flanagan99] D. Flanagan. Java in a Nutshell. O'Reilly & Associates, 3rd. Edition. December 1999. ISBN 1565924878.

[Fowler85] R. J. Fowler. Decentralized object finding using forwarding addresses. Tech. Rep. 85-12-1, Dept. of Computer Science, Univ. of Washington, Dec. 1985.

[Freedman00] D. H. Freedman. Attack of the Killer Viruses. The New York Times - Editorial Desk. May 6, 2000, Saturday Edition.

[GKNRC96] Kotz D., R. Gray, D. Rus, S. Nog and G. Cybenko, Mobile Agents for Mobile Computing. In Technical Report PCS-TR96-285, Computer Science Department, Dartmouth College, May 1996.

[Gray96] R. S. Gray. Agent Tcl: A flexible and secure mobile-agent system. In Proceedings of the 4th Annual Tcl/Tk Workshop (TCL '96), July 1996.

[IH99] L. Ismail, D. Hagimont. A Performance Evaluation of the Mobile Agent Paradigm. Proc. Of the 1999 ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications. November 1-5, 1999, Denver, CO USA.

[IloveYou00] I Love You Virus in News. CNET.com News – Enterprise Computing. May 16, 2000. http://news.cnet.com/news/0-1003-201-1826257-0.html?tag=st.cn.sr1.ssr.ne_virus .

[JRS95] D. Johansen, R. van Renesse, and F. B. Schneider. Operating System Support for Mobile Agents. In Proceedings of the 5th IEEE Workshop on Hot Topics in Operating Systems (HotOS-V), pages 42--45, May 1995.

[KLO97] G. Karjoth, D. Lange, and M. Oshima. A seciruty Model for Aglets. IEEE Internet Computing, Vol. 1. No. 4, July-Aug. 1997. pp. 68-77.

[KLO97] G. Karjoth, D. Lange, and M. Oshima. A Security Model for Aglets. IEEE Internet Computing, pages 68--77, July-August 1997.

[KT98] N. Karnik and A. Tripathi. Agent Server Architecture for the Ajanta Mobile-Agent System. In Proceedings of the 1998 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'98), July 1998.

[KT98] N. M. Karnik and A. R. Tripathi. Design Issues in Mobile-Agent Programming Systems. IEEE Concurrency, July-September, 1998.

[LO99] D. B. Lange and M. Oshima. Seven good reasons for mobile agents. Communications of the ACM, 42(3): 88-89, March 1999.

[ObjectSpace97] ObjectSpace: ObjectSpace Voyager Core Package Technical Overview. Tech. Repot. ObjectSpace Inc. Dallas, 1997.

[OMG-MASIF98] OMG doc orbos/98-03-09 - Evaluation Report on the Mobile Agent Facility Joint Submission. Revised Version.

[OMG-OBV96] OMG doc orbos/96-06-14 - Objects-By-Value RFP

[OMT98] P. Oreizy, N. Medvidovic, and R. N. Taylor. Architecture-based runtime software evolution. In Proceedings of the International Conference on Software Engineering 1998 (ICSE'98), Kyoto, Japan, April 1998.

[RGK97] D. Rus, R. Gray, and D. Kotz. Transportable Information Agents. Proceedings of the First ACM International Conference on Autonomous Agents , 1997, pp. 228 – 236.

[SWME00] R. S. Silva Filho, J. Wainer, E. R. M. Madeira, C. Ellis – CORBA Based Architecture for Large Scale Workflow. Special Issue on Autonomous Decentralized Systems of the IEICE Transactions on Communications, Tokyo, Japan, Vol. E83-B, No. 5. May 2000, pp.988-998.

[Vinoski 98] S. Vinoski. New features for CORBA 3.0.

Communications of the ACM, Volume 41, Num. 10. ACM Press. October 1998. pp 44 - 52.

[VoyagerRef] http://www.objectspace.com/products/voyager/

[White94] J.E. White. Telescript Technology: The Foundation for the Electronic Marketplace. Whitepaper by General Magic, Inc, Sunnyvale, CA, USA, 1994.