How a Secure and Open Mobile Agent Framework Suits Electronic Commerce Applications

Paolo Bellavista, Antonio Corradi,
Rebecca Montanari
Dip. di Elettronica, Informatica e Sistemistica
Università di Bologna
Viale Risorgimento 2, 40136 Bologna, Italy
{pbellavista, acorradi, rmontanari}
@deis.unibo.it

Cesare Stefanelli
Dipartimento di Ingegneria
Università di Ferrara
Via Saragat 1, 44100 Ferrara, Italy
cstefanelli@ing.unife.it

Abstract

The Mobile Agent (MA) paradigm seems able to support effectively distributed applications in open and heterogeneous environments, and application areas such as e-commerce appear to be of particular interest. However, MA technology has to answer to the requirements of security and interoperability to achieve wide deployment, especially in e-commerce applications. The paper focuses on security and interoperability, and describes a Secure and Open Mobile Agent (SOMA) programming environment where both requirements are main design objectives. On the one hand, SOMA is based on a thorough security model and provides a wide range of mechanisms and tools to build and enforce flexible security policies. On the other hand, the SOMA framework permits to interoperate with different application components designed with different programming styles. SOMA grants interoperability by closely considering compliance with the OMG CORBA and MASIF standards. In particular, the paper presents a SOMA-based e-marketplace that stresses to the limit the security and interoperability issues and that has served as a testbed for the validation of SOMA security and interoperability support.

1. Introduction

Global distributed systems such as the Internet and the Web have proposed a new framework for application development and have motivated the interest in new and flexible programming paradigms based on dynamically mobile entities. Remote Evaluation, Code On Demand and Mobile Agents (MA) [1] [2] [3] propose the migration not only of data but also of code over the network, to overcome the limits of the traditional client/server (C/S) model. The MA model differs from the others because it allows executing entities to decide autonomously about their migration (with their own code and execution state).

Many application areas, such as electronic commerce, mobile computing, network management and information retrieval can benefit from the application of the MA technology. However, the deployment of MA in these application domains can be accelerated as soon as MA systems can provide solutions that respect the *opening* and *closing* properties. The *opening property* permits to overcome the system boundaries in order to interoperate with any necessary external component and to allow any external recognized usage, while the *closing property* is the possibility of constraining the system in such a way to identify and exclude any malicious intrusion. The *opening* property is granted by *interoperability* considerations and the *closing* property by *security* mechanisms and policies. Because the first MA prototypes fail in satisfying the *opening* and *closing properties*, the recent MA research has focused on the areas of interoperability and security.

Interoperability is an important property for MA systems. In general, MA proposals tend to provide an easy access to users of the Internet and the Web. We claim that full interoperability means not only the ubiquitous accessibility via friendly Web interfaces, but also the capacity of interacting with other applications, either designed according to the same MA style or with very different programming models, and even with legacy systems. The goal of interoperability requires to identify the aspects of the MA technology candidate to become standard. The main recognized effort in the standardization toward interoperability of Object-Oriented components is Common Object Request Broker Architecture (CORBA), promoted by the Object Management Group (OMG) [4]. The OMG works in different specialized

Investigation supported by the Italian "Consiglio Nazionale delle Ricerche" in the framework of the Project "Global Applications in the Internet Area: Models and Programming Environments" and by the University of Bologna (Funds for Selected Research Topics: "An integrated Infrastructure to support Secure Services").

areas, and one of its subgroups has defined the MASIF standard [5]. MASIF integrates the traditional C/S model and the MA paradigm, thus providing CORBA-based interfaces for agent registration, management and transfer. There are also other interesting approaches toward the standardization of many aspects of the MA technology, such as the FIPA proposal that focuses on the definition of general standard languages and protocols for communication and management of heterogeneous agents [6].

Security is a fundamental issue when dealing with mobile agents in the Internet environment. Mobility increases the threat of security violations, due to the dynamic injection of possibly malicious agents by untrusted users that can compromise the resources of the hosting nodes. In addition, mobility introduces new security issues: the hosts responsible for agent execution could try to tamper with agent code and state in order to disclose agent private information, and could gain competitive advantage with respect to other nodes, and could refuse to transfer the agent to successive execution sites. On the one hand, the problem of host protection against malicious agents has been extensively investigated. Technologies such as Java sandboxes and type safe languages seem to effectively ensure that incoming agents cannot access information they are not authorized to act upon, they cannot cause a denial of service to other authorized entities, and they cannot deliberately interfere with agents of other users [7], [8]. On the other hand, the protection of mobile agents against malicious behavior of execution environments is specific to the MA technology and represents an active and challenging research area investigated only by a few proposals [8], [9].

The paper describes an MA framework called Secure and Open Mobile Agents (SOMA, available at `http://lia.deis.unibo.it/Research/SOMA/`) that answers the key issues raised by the adoption of an MA programming environment for the Internet and offers a large variety of policies and mechanisms to achieve proper levels of security and interoperability. SOMA has been exploited to build several applications in different areas, such as network management, multimedia distribution and e-commerce. In particular, the paper shows SOMA in the area of e-commerce: the reported application has served as a testbed for the validation of the security and interoperability modules of the SOMA framework.

2. The SOMA Programming Environment

This section gives a general overview of the architecture of the SOMA programming environment and then presents the design solutions adopted to provide the opening and closing properties.

2.1. The SOMA architecture

SOMA is based on a hierarchy of locality abstractions to model and describe any kind of open and global distributed system, ranging from simple LANs to the Internet. Any node owns at least one *place* that constitutes the agent execution environment. Several places can be grouped into a *domain* abstraction that corresponds to a network locality. In each domain, a *default place* hosts a gateway to perform inter-domain functionality, to maintain domain-specific runtime information and to permit full integration with other components via the interoperability facility.

Places and domains abstract a logical representation of system physical resources, and permit users and administrators to express their needs in terms of mobile agents, and to develop new applications and services based on mobile code. Mobility is an intrinsic feature not only of agents but also of most entities in the SOMA system, e.g. a *mobile place* is suitable for modeling the behavior of a nomadic terminal.

The SOMA programming framework is designed according to a layered architecture, built on top of the Java Virtual Machine (JVM), as depicted in Figure 1. SOMA main features are provided by the facility layer, that contains the principal functionality to fully support MA-based applications. This layer is composed by a set of basic facilities and by two distinguished components in charge of granting the opening and closing properties.

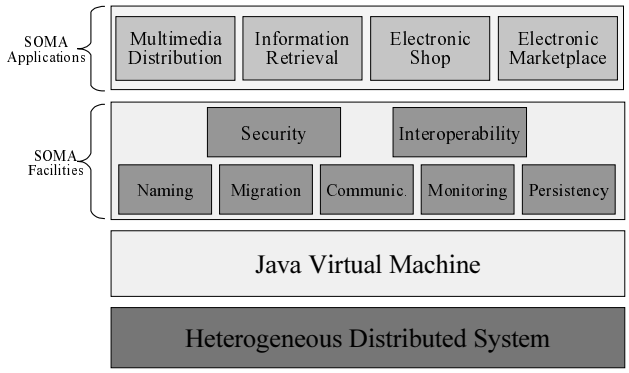

Figure 1. The SOMA layered architecture.

The SOMA basic facilities include:

- the *naming* facility to associate entities with globally unique identifiers and to organize these identifiers in name systems to make possible the tracing of entities even if they move. This facility allows to put together a set of different naming systems (DNS-, CORBA-, and LDAP-compliant), possibly characterized by different resolution policies, and is currently implemented by a coordinated set of dedicated agents;
- the *migration* facility to permit the migration of one entity that should change its allocation. The reallo-

cated entity should be traced also in the new location by any entity in need of its services, and, if it is active, should transparently restart its execution at the new location;

- the *communication* facility to provide tools for coordination and communication between possibly mobile entities. Within the same place, agents interact by means of shared objects, such as blackboards and tuple spaces for tight cooperation. Outside the scope of the place, agents can perform coordinated tasks by exchanging asynchronous messages that are delivered to agents also in case of migration;
- the *monitoring* facility to observe the state of local resources and services and to provide this information to the application level for permitting dynamic adaptive reactions. SOMA can monitor both system indicators (e.g. CPU load, file system occupation, printer status, available network bandwidth and collision rate) and application indicators (e.g. available services, program versioning, local agent states);
- the *persistency* facility to give the possibility to suspend temporarily executing agents and to store them on a persistent medium. The facility allows agents not to waste system resources while they are waiting for external events such as the reconnection of one user or terminal to yield back the results of autonomously performed operations. In addition, it can be also exploited in providing fault-tolerant solutions by organizing and disseminating stored copies of agents [10].

As shown in Figure 1, on top of the basic facilities we have realized two advanced SOMA facilities to provide a rich set of mechanisms for interoperability and security.

2.2. The SOMA interoperability Facility

The large number of recently implemented MA platforms shows the interest of the distributed systems area in the MA programming paradigm. This variety, however, risks to endanger interoperability and to limit the growth of an MA applications industry. The only way to promote both interoperability and system diversity is to standardize some aspects of the MA technology.

In the area of the traditional C/S approach to OO distributed computing, CORBA is the universally adopted standard for object management, apart from the notable exception of Microsoft which has its own Distributed Component Object Model (DCOM) [11]. CORBA puts together objects that can communicate to each other across boundaries such as network, different operating systems, and different programming languages. CORBA provides network transparency, openness and interoperability. In addition to that core functionality, it specifies an extensive set of object services, common facilities and application interfaces.

The MA model embodies a new programming paradigm, distinguished from the C/S one, and, consequently, different from the CORBA programming model under many points of view (e.g. location awareness vs. network transparency). However, we claim that CORBA can play a fundamental role in achieving interoperability also for the MA technology, working as a standard bridge among proprietary implementations.

CORBA compliance imposes some additional costs, especially in terms of run-time performance, but it is worth the trouble because it ensures openness and stability to applications, saving the investments in MA-based programming. Our scenario puts together two models: we use proprietary and efficient solutions for internal operations among SOMA entities, but provide standard CORBA interfaces, both for exploiting the available CORBA services and for making SOMA itself a CORBA application object.

There are three different aspects in providing SOMA compliance with CORBA:

1. an agent may call external CORBA objects (SOMA agents as CORBA clients);
2. an agent may publish its interface on a CORBA ORB (SOMA applications as CORBA servers);
3. any external entity may manage SOMA entities through the standard MASIF interface (interoperability between SOMA and other MASIF-compliant MA platforms).

The first two features are obtained via the *CORBA C/S* module of the SOMA interoperability facility, which provides functionality to simplify the application designer duty in deploying CORBA-compliant components: agents can play the role of CORBA clients or can register themselves as CORBA servers to offer an application access point outside the SOMA system. Even if there is no conceptual problem for a mobile agent to register itself as a CORBA server, we currently grant this possibility only to SOMA agents that do not migrate during their lifetime (*stationary agents*) in order to avoid the burden of de/registering to the CORBA Naming Service at any migration.

The third functionality is more complex an issue, and is addressed by the MASIF standard. MASIF does not suggest standardization of local agent operations such as agent interpretation, serialization, execution and deserialization because these actions are application specific, and there is currently no reason to limit MA system implementations to a single rigid architecture. MASIF proposes a standardization for agent and agent system names, for agent system types and for location syntax. It defines two interfaces (`MAFAgentSystem` and `MAFFinder`) with the typical sets of functionality respectively for agent management and for agent tracking. Agent management allows an external system to control agents of a MASIF-compliant MA system: MASIF de-

fines interfaces for actions such as suspending/resuming/terminating agents or for moving agents from an MA system to another one if the two systems have a compatible type. Agent tracking permits the tracing of agents registered with `MAFFinder`s, which essentially provide an MA name service, since the CORBA Naming Service is not suitable for entities like agents, which are mobile by nature. Agent communication is outside the scope of MASIF, since it is extensively addressed by CORBA as object communication.

The CORBA and MASIF standards recognize the need of security information and its management; all the implementations are required to introduce tools and mechanisms to enforce security when interacting with external components. SOMA also considers the additional security threats introduced by CORBA interoperability. On the one hand, sending/receiving CORBA requests/replies requires security techniques for channel encryption to ensure privacy on message exchange. On the other hand, the possibility for SOMA agents to act as CORBA servers and for SOMA places to host non-SOMA agents call for mechanisms for client/agent authentication, auditing and access control. SOMA takes into account the security problems by providing solutions compliant to both CORBA Security Services (CORBA SSs) [12] and MASIF security features.

2.3. The SOMA Security facility

SOMA addresses, as a distinctive feature, the security issues that emerge in the context of real MA applications running over the Internet. A crucial goal of MA security is to identify the application-specific proper balance between different and sometimes contrasting requirements: flexibility, usability, and efficiency. For this reason, the SOMA programming environment provides a large variety of security mechanisms, policies and tools for the execution sites and the agents while both migrating over insecure networks and executing in malicious hosts, to flexibly answer the different requirements of application designers. Whenever possible, SOMA security model has been implemented by taking into account only the standard security solutions usually employed in distributed systems. In fact, the design and the exploitation of ad hoc security mechanisms could require too great an effort, and, more important, non standard tools are unlikely to be accepted.

With regard to the protection of execution sites, the definition of different locality abstractions allows to enforce layered security policies in which actions are controlled at both domain and place level. The domain defines a global security policy that imposes general authorizations and prohibitions; each place can only apply restrictions to the domain-level set of permissions.

Authentication and authorization are enforced at both domain and place level. Agents entering a domain are authenticated on the basis of their *credentials* which are a series of unforgeable information such as the names of the originating domain and place, and the name of the principal the agent acts on behalf of.

Once authenticated, the agents are authorized to interact with the resources on the basis of the current security policy; each of the resources has its specific role-based access control (RBAC) list for all principals [13]**.** In the RBAC model permissions are assigned to roles rather than to individual users in order to improve manageability: security policies do not have to be changed when users are assigned to new roles. Whenever a SOMA agent is instantiated, it is provided then with a role certificate [14] associating in a secure manner the agent with the correspondent role of its owner. The agent is then authorized to access resources on the basis of its role certificate.

SOMA protects the agents moving in an untrusted environment in terms of both secrecy and integrity. Secrecy is achieved by using encrypted and authenticated channels. The protection of agents against possibly malicious hosting places requires to ensure the integrity of mobile agent, to detect any possible modification of the data collected by agents moving in an untrusted domain. To this purpose, SOMA provides a number of integrity protocols that can suit different application scenarios by balancing application specific requirements and protocol efficiency and scalability [15].

The SOMA security facility provides advanced security services, such as certification and role/policy management services. SOMA users register offline to a SOMA domain in order to be identified and can then acquire their credentials through an online certification process. In addition, the SOMA support provides tools to simplify the dynamic management of policies and roles to authorised administrators. For instance, one administrator can change the system security policy of one domain by using a graphical policy editor and these modifications are propagated automatically to all the places within a domain, with no service suspension. Another graphical tool to manage run-time roles permits administrators to define new roles and update existing ones, associating principals with roles, and automatically generating corresponding place policies.

The wide variety of security mechanisms available in SOMA, and the clear distinction between mechanisms and policies give the possibility to decide a suitable trade-off between security needs and required performances, tailored on specific circumstances. Agents from internally trusted domains can directly access to authorization check, while agents from untrusted ones have to overtake all the secrecy, integrity, authentication and authorization steps.

3 SOMA Mobile Agents for E-commerce

We have already exploited the SOMA framework to design, implement and deploy MA-based applications in several areas, from network and systems management to e-commerce (presented in the following), from adaptive multimedia distribution with Quality of Service requirements to autonomous retrieval of highly heterogeneous information for virtual museum services. Here, we intend to present the experience in the realisation of an e-marketplace environment implemented in the SOMA programming framework. We describe this experience because e-commerce operations stress to the limit the interoperability and security features of SOMA.

3.1 Mobile Agents for E-commerce

The Internet and the agent technology have pointed out the possibility of new business models ranging from e-shops and e-auctions to virtual enterprises and e-marketplaces.

The Internet represents the most suitable global network support for e-commerce applications, where services are available to potential customers without any temporal and spatial limit. The software agent technology seems an attractive paradigm to support e-commerce applications, because agents acting on behalf of end-consumers could reduce the efforts required to complete an e-commerce transaction. Agents are intrinsically autonomous and can be easily personalised to embody end-consumer preferences. In addition, they are adaptive and capable of learning from both past actions and their environment, in order to cope with changing network conditions and evolving user requirements. The above features permit agents to add value to three primary e-commerce dimensions: information filtering, information gathering and retrieval, and dynamic and flexible execution of transactions [16] Agent-based electronic shops, auctions, and marketplaces have demonstrated the potential of software agents [17], [18].

Adding mobility to software agents increases even further the potential of applications in the e-commerce area. To accomplish their goals, mobile agents can not only autonomously act and negotiate on behalf of their creators in an e-commerce service provider, but can also autonomously decide to move from one e-commerce service provider to another.

3.2 A SOMA E-marketplace

An e-marketplace consists of a collection of e-shops ruled by the e-marketplace owner in charge of ensuring the transactions between consumers and e-shop providers and of supporting the necessary technological infrastructure. The SOMA environment can help in the realisation of an e-marketplace. A market is represented by a SOMA domain and any e-shop is associated with a SOMA place. SOMA helps in the automation of e-marketplace activities by providing the required naming, communication and security facilities: all participants can easily find each other through the naming facility, and consumers/sellers can exchange information and negotiate conditions in a secure way.

Each e-marketplace participant can exploit specific goal-driven SOMA mobile agents (see Figure 2):

- *buying* agents represent consumers. SOMA buying agents can exploit mobility to visit different markets to collect information. Mobility permits to perform all the required operations locally to the involved marketplace, without the need of maintaining reliable connections with remote hosts. Consumers can disconnect while their agents roam in the network to reduce connection costs;
- *management* agents can support e-marketplace owners in the setup of e-shop providers and consumers. Management agents can be instantiated to automate the distribution to e-shop providers of information regarding the marketplace, such as shared resources or marketplace general security policies.
- *advertising* agents are a kind of management agents that an e-marketplace owner can exploit to advertise the services offered by the marketplace. When a consumer firstly joins a market, an advertising agent can be sent to the consumer node to configure her buying system with the pertinent list of e-shop and product descriptions.

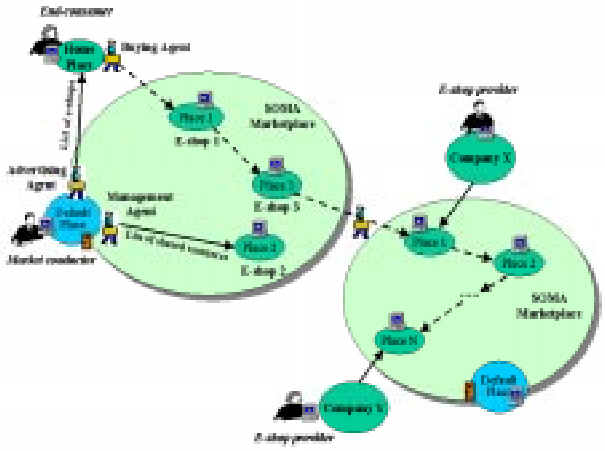

Figure 2. A SOMA based e-marketplace.

Interoperability is a fundamental requirement to provide integration with possibly heterogeneous legacy systems. In fact, e-shop providers are likely to maintain information about their products already stored in existing databases and legacy components. Their service provision via e-markets is significantly accelerated by the

possibility of encapsulating and reusing already available resources and services. E-shops can provide access to their product databases via appropriate CORBA servers, thus making available the information to buying agents.

Security is another basic concern in e-marketplaces: both e-shop and consumer interests should be protected with appropriate security solutions. The SOMA security framework adapt to the security requirements of e-shops and consumers by providing the necessary authentication, authorisation, secrecy and integrity services. Mobile agents are authenticated on the basis of their responsible users and authorised on the basis of corresponding roles. For example, a customer can allow in only the *advertising* agents sent and signed by one trusted marketplace owner, while she can discard any other incoming agent in order to reduce the risk of security threats. It is worth noticing that the flexible choice in SOMA of different security solutions allows any SOMA e-marketplace to specify its suitable level of security and to select the needed security mechanisms accordingly.

4. Conclusions

The MA technology seems to be an elegant and uniform solution to a wide spectrum of application areas, from network and systems management to mobile computing, from distributed information retrieval to electronic commerce. Many MA systems already exist, but the major limit to their use in real complex applications for the Internet stems from their lack of security and interoperability.

SOMA has been designed to provide an integrated approach to security. We have implemented several mechanisms to protect local resources from malicious agents, agents from untrusted hosts, and communication among SOMA entities, in order to obtain a suitable balancing between security and performances.

Interoperability among different MA systems seems the other missing element to leverage the growth of MA applications industry. SOMA faces the issue of CORBA compliance, carefully taking into account its cost in terms of the added system overhead. SOMA agents can play the role of CORBA clients/servers; one place for domain, at least, publishes the MASIF interface to the CORBA ORB, thus permitting the full interoperability with other external and authorized CORBA systems, either MA-based or not.

SOMA makes available a large variety of mechanisms, policies and tools to achieve flexibly proper levels of security and interoperability. These properties make our programming framework particularly suitable for the design, implementation and deployment of distributed services in several application domains, such as e-commerce. In particular, we have extensively worked in the e-commerce area, where agents fulfill the customer and e-shop providers needs by supporting e-commerce transactions on their behalf and by helping in the information gathering, filtering, and negotiation phases.

References

[1] A. Fuggetta, et al., "Understanding Code Mobility", IEEE Transactions on Software Engineering, Vol. 24, No. 5, 1998.

[2] J.W. Stamos, and D.K. Gifford, "Remote Evaluation", ACM Transaction on Programming Languages and Systems, Vol. 12, No. 4, Oct. 1990.

[3] K. Rothermel, and F. Hohl (ed.), 2nd International Workshop on Mobile Agents, Springer-Verlag, Lecture Notes in Computer Science, Vol. 1477, Sep. 1998.

[4] Object Management Group, CORBA/IIOP Rev 2.2, OMG Document formal/98-07-01, http://www.omg.org/library/, Feb. 1998.

[5] GMD FOKUS, and IBM Corp, Mobile Agent Facility Specification, Joint Submission supported by Crystaliz Inc., General Magic Inc., the Open Group, OMG TC Document orbos/97-10-05, ftp://ftp.omg.org/docs/orbos/, 1998.

[6] Foundation for Intelligent Physical Agents – FIPA'99 version 0.2, http://www.fipa.org/spec/.

[7] L. Gong, "Java Security: Present and Near Future", IEEE Micro, Vol.17 N.3, 1997.

[8] G. Vigna (ed.), Mobile Agents and Security, Lecture Notes in Computer Science, Vol. 1419, Springer Verlag, 1998.

[9] B. Yee, "A Sanctuary for Mobile Agents", DARPA Workshop on Foundations for Secure Mobile Code, Monterey (CA), USA, 1997.

[10] F.M. Assis-Silva, et al., "An Approach for Providing Mobile Agent Fault Tolerance", in [3].

[11] Microsoft – DCOM, http://www.microsoft.com/com/tech/DCOM.asp.

[12] Object Management Group, CORBA Security Services, OMG Document formal/98-12-17, http://www.omg.org/cgi-bin/doc?formal/98-12-17, Dec. 1998.

[13] Ravi S. Sandhu, et al., "Role-Based Access Control Models", IEEE Computer, Vol. 29, No. 2, 1996.

[14] N. Nagaratnam, et al., "Role-Based Protection and Delegation for Mobile Object Environments", ECOOP'98 Workshop on " Secure Internet Mobile Computations", Brussels, Belgium, 1998.

[15] A. Corradi, et al., "Mobile Agents Integrity for Electronic Commerce Applications", Information Systems, Vol. 24, N.6, 1999.

[16] R. Guttman, et al., "Agent-mediated Electronic Commerce: A Survey", Knowledge Engineering Review, Vol. 13, N.2, 1998.

[17] A. Chavez, et al., "Kasbah: An agent marketplace for buying and selling goods", PAAM'96, London, Great Britain, 1996.

[18] B. Doorenbos, et al. Etzioni O., and Weld D., "A scalable comparison-shopping agent for the world-wide web", AGENTS'97, ACM, Marina del Ray, USA, 1997.