The Mobile Agent Technology to Support and to Access Museum Information

Paolo Bellavista
Dip. Elettronica, Informatica e Sistemistica – Università di Bologna
Viale Risorgimento, 2
Bologna - Italy
Ph.: +39-051-2093087
pbellavista@deis.unibo.it

Antonio Corradi
Dip. Elettronica, Informatica e Sistemistica - Università di Bologna
Viale Risorgimento, 2
Bologna - Italy
Ph.: +39-051-2093083
acorradi@deis.unibo.it

Andrea Tomasi
Dip. Ingegneria dell'Informazione
Università di Pisa
Via Diotisalvi, 2
Pisa - Italy
Ph.: +39-050-568560
tomasi@iet.unipi.it

ABSTRACT

The global scenario put together by communication networks determines new opportunities towards the realization of Internet-based distributed services in many complex and composite application areas, such as the access to museum information. Solution complexity mainly stems from the heterogeneous representation formats of data, their geographical distribution, the large number of data sources involved, and the user requirements for personal customization and optimization of the accessed services. The paper claims that the realization of flexible museum information access services require a middleware-level approach and the implementation of a distributed support infrastructure. Within the MOSAICO project, we have realized the VM (Virtual Museum) framework on top of the SOMA (Secure and Open Mobile Agent) programming environment. Mobile agents have been chosen for their intrinsic properties of autonomy, asynchronicity, dynamicity of distribution, and adaptability to available system resources. We have designed the VM to accommodate different usage scenarios: VM users can play different roles with different expertise levels; they can ask the VM infrastructure for differently complex features, from simple Web accessibility to user accounting for data resource consumption, from data customization via user profiling to automatic update of subscripted query results. The first prototype, apart from the feasibility of the approach, has already shown the potential and the flexibility of the mobile agent infrastructure to adapt to both different user requirements and different resource availability.

Keywords

Virtual Museums, Internet Services, Mobile Agents, Heterogeneous Data Resources, Web Accessibility, Asynchronicity.

1 INTRODUCTION

The emerging scenario of global communications together with the increasing diffusion of Internet-based services have given a definite impulse to several application areas, by suggesting not only new solution directions, but also new design frameworks for the realization of distributed services. As the most notable example, on the one hand, the Internet communication infrastructure has made possible the cooperation of geographically distributed users via interactive tools, such as the ones to support video conference; on the other hand, the Internet is undergoing continuous evolution to answer the raising user expectations for granting the desired level of Quality of Service (QoS). At the same time, several support areas are still unexplored and attract many proposals and projects, and specific application fields are still in need of attention, such as information retrieval, cooperative work and electronic commerce.

The paper addresses the possibility of realizing a virtual museum by using the Internet as the network infrastructure to make available heterogeneous museum information, from pictures, figures and maps to texts, audio and animated images. The virtuality attribute stems from the impossibility of putting physically together the whole information and from the opportunity of creating a service infrastructure only via the communication layer. In fact, the considered application field for our virtual museum is represented by the collection of information related to artistic and historical objects, as manufactured goods, tissues, paints and documents, which public and private organizations in Italy have recently started to organize. The disseminated sites involved are public and private museums, archives, libraries, either open to visitors or only accessible to researchers. They are characterized by diversity and complexity, as well in heterogeneous data content as in system organization, because they have operated in a completely independent way, without the definition of common standards for neither storing the inventory data and the digital representations of the cultural patrimony, nor providing catalogues and indexing structures at the local sites or at the global level.

The paper defines a framework called **V**irtual **M**useum[*] (**VM**), and presents its design and implementation on the top of a mobile agent programming environment called SOMA (Secure and Open Mobile Agent) [2]. The VM provides the accessibility to the

[*] The VM framework is available at `http://www-lia.deis.unibo.it/Research/SOMA/VM.shtml`.

above collection of digital representations of artistic and historic artifacts, independently of the physical location of the information. Data representations span from simple inventory forms to object 2D/3D visualization, and may even collect Web links to more detailed descriptions, as specialized publications or audiovisual comments. The VM solution offers to users a Web-enabled and simple interface to access the museum information. The interface permits different degrees of expertise to different user typologies, thus covering the main real application scenarios, from simple query services for generic visitors, to more protected and customizable policies for supporting the activity of expert users and managers. The VM provides mechanisms and tools to authenticate users, to associate them with their proper role, and to account their inquiry and processing activity according to the policies adopted by the data providers [16]. Differentiated views and browsing possibilities on the linked museum information can be obtained on the basis of the different user roles and system policies.

The paper claims that the flexibility required in the museum application area (e.g. for integrating heterogeneous and legacy data servers in an open framework, for customizing query results depending on user role and dynamically changing profile information, and for adapting the resource consumption of query services to the currently available QoS) calls for the realization of a distributed support infrastructure. This infrastructure significantly takes advantage of an implementation based on mobile agents: the Mobile Agent (MA) programming paradigm stresses properties that are particularly suitable for the VM application area, such as the MA capacity of independent behavior and execution, of dynamic migration and exploration of open global environments, of run-time adaptation to the hosting sites and their resource availability [9]. In addition, to overcome data heterogeneity, the VM framework integrates MA with the more traditional client/server model of interaction, by exploiting widely diffused database access technologies, such as JDBC and ODBC [22, 20], and standardized solutions for distributed object infrastructures such as CORBA [21].

The paper is structured as follows. Section 2 describes the VM framework requirements and briefly proposes guidelines for solution. Section 3 underlines the properties of mobile agents that claim for the adoption of the MA technology to implement the VM infrastructure. Section 4 presents the architecture of the SOMA programming environment, while Section 5 focuses on the detailed description of the VM components and of their duties in different usage scenarios. In particular, the paper shows that different levels of functionality and complexity are possible depending on user roles and expertise. Directions of evolution, which currently guide our implementation work, and concluding remarks follow.

2 VIRTUAL MUSEUM REQUIREMENTS AND GUIDELINES FOR SOLUTION

The VM framework should be able to connect and collect the whole information related to artistic and historical artifacts located at many different geographically distributed sites. Any artifact belonging to the VM should be described by a dynamically determined inventory form, which contains all the related information available, allows the visualization of the artifact and establishes links to further known documentation about it. Digital representations of artifacts include:

- simple inventory data, with coded or descriptive fields related to physical location, conservation state, time of manufacturing, material and technique used, name of the author, ...;
- photo images;
- 3D coded representation for virtual exhibition;
- miscellaneous descriptive documentation, such as archive documents, historic notes, expert publications and audiovisual material, with/out hypermedia links.

The scenario is complicated by the fact that the inventory form fields may also contain imprecise or uncertain values because of incomplete classification and attribution work, sometimes due to controversy between experts. In addition, the documentation related to one artifact is usually spread to many different sites and, in most cases, neither the document links are present into the inventory form, nor a link chain exists to reach all of them. Public organizations devoted to the maintenance of artistic and historic patrimony have recently started to consider the need for standardization, but the definition and acceptance of a standard for museum data content, and for user/management functionality to be provided by VM services is still long to be achieved.

Moreover, the VM framework should answer very different user requirements. VM users can be roughly divided into three classes, with different authorization rights and processing needs: managers, experts and generic visitors. Managers usually belong to the organization that host the VM data: they are aware of object locality; they need full query services to support patrimony management, exhibition arrangement, digital publishing and some form of electronic commerce. Expert users need to investigate extensively and completely the information related to specified artifacts, usually with no limitation on the VM resource consumption, possibly discovering new data and establishing new relationships between objects; the result of expert work may be new material to be added to the VM repository. Generic visitors are normally interested in examining a single piece, or in on-line navigation from artifact to artifact to collect the available even incomplete information. The VM should allow managers, experts and visitors to see different data views, with different degrees of authorization (maximum for the manager, minimum for the visitor). The accounting policy should be bounded by any local responsible site but, in most cases, the access to the information is free.

The largest part of the existing distributed services in the application area of museum information mainly follows the Client/Server (C/S) programming paradigm. The C/S approach has demonstrated to be a satisfying solution to realize distributed queries that operate on completely known and static information, by applying the traditional techniques of distributed databases and exploiting standard access interfaces such as JDBC and ODBC [22, 20]. In this context, CORBA can play a crucial role to allow and simplify the integration with legacy data systems, which are very frequent in the museum information area and rarely support diffused access interfaces. The traditional C/S model of interaction is also a suitable programming paradigm to provide browsing functionality: users should be able to navigate the VM from site to site, following existing links via standard Web browser solutions. In this case, Java applets [7] can significantly extend the interaction model by dynamically moving code to the client host that can actively take part in service provision.

However, other application-specific issues (e.g. dynamicity - extending service functionality, moving data servers, modifying security policies, at run-time; asynchronicity - pre-fetching the results of a query before the connection of the interested user; adaptability - tailoring data to user profile requirements and current resource availability) stress the necessity of integrated solution frameworks that achieve a degree of flexibility higher than the one available in frameworks built upon the traditional C/S model. We claim that emerging programming paradigms [9], such as mobile agents, which focus on code mobility to emphasize flexibility and dynamicity in service provision, have to be considered an interesting enabling technology for VM services, not in alternative but as a necessary extension to the C/S solution.

Our VM infrastructure has the additional implementation constraint to continue to exploit some software components (e.g., interfaces) belonging to a previous C/S application, well-known by involved museum operators who used it at local sites to manage inventory information of artistic objects. In particular, Figure 1 shows respectively the query form that permits to employ dictionaries and lists of terms, and the inventory form possibly sent as a query result (in Italian). The required "back-compatibility" of the interfaces forces to consider VM solutions capable of integrating existing and heterogeneous components and of simplifying the data exchange between them.

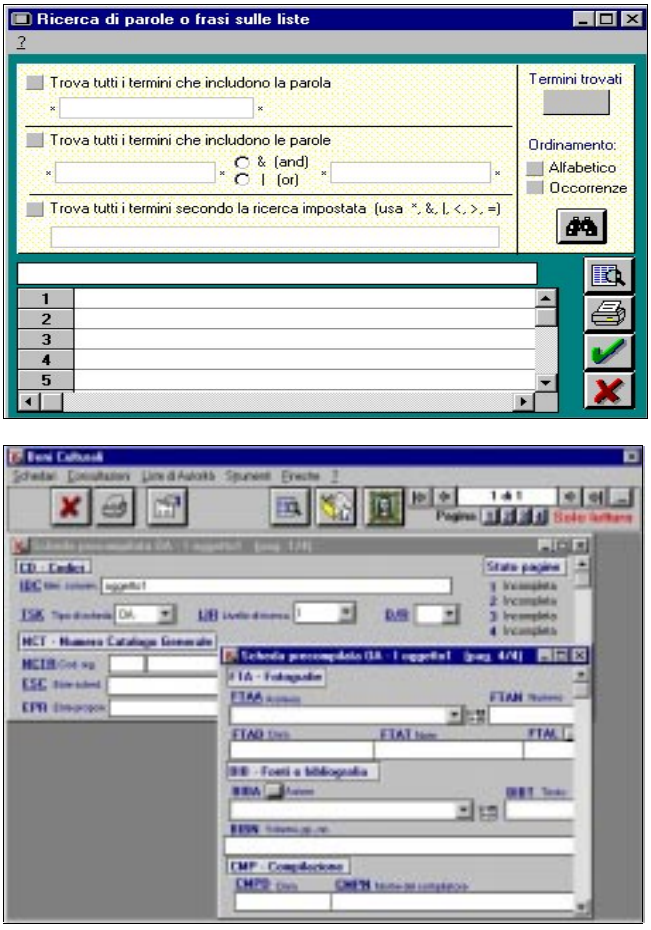

Figure 1. The dictionary-based query form and the inventory form of the VM.

3 MOBILE AGENT INFRASTRUCTURES FOR MUSEUM INFORMATION

The intrinsic characteristics of the information retrieval in the above presented case of geographically distributed heterogeneous museum information require to design a distributed infrastructure that also supports Web accessibility. It is recognized that the design, deployment and management of large-scale and complex distributed services in global environments such as the Internet calls for solutions at the middleware level [5, 11, 18]. In this context, services can be implemented in terms of distributed coordinated components, where any component can concentrate on the specific issues related to service provision because of the possibility of exploiting the general-purpose facilities provided by the middleware infrastructure. In our case, the needed infrastructure has not only to provide Web accessibility to a large amount of highly autonomous, heterogeneous and unstructured distributed information with differentiated views depending on user roles, but, because of the multimedia nature of the interesting data, also to optimize network-bandwidth consumption by exploiting local interrogations on information sources and by realizing the distributed caching of general interest query results.

We claim that the MA technology is particularly suitable for the realization of this infrastructure because mobile agents are autonomous entities with a large capacity of coordination, able to dynamically move where the needed resources are located, and able to operate and adapt to the current system conditions in a completely asynchronous way with regard to their launching user [9]. As a consequence of the adoption of the MA paradigm, a VM infrastructure based on mobile agents can achieve the following properties:

- *distribution of service control*; VM services are realized in terms of cooperating mobile agents, that are placed where they are needed to exploit locality in the management and control of information resources. In addition, agents can autonomously decide to move dynamically to follow possible movements of both users and available resources;
- *enhanced accessibility*; agent mobility makes possible to dynamically migrate service components where needed. In particular, mobile agents can also implement proprietary clients that are capable of dynamically moving to run-time determined access points. In that way, they enhance service accessibility even in case of the compulsory use of proprietary non-standard access interfaces;
- *customizability*; mobile agents provide an effective mechanism to support service customization in response to dynamically specified user requirements. Dedicated agents can be in charge of collecting profile information about the user desiderata, of migrating profile information depending on the user current access point, and of adapting consequently VM information flows to users (data filtering, multimedia format conversion, …);
- *adaptability*; agents make possible to adapt services to the current system situation (*location-aware* MA-based services [17]). For instance, monitoring agents can exchange information and move themselves to obtain a global view of the system state. This knowledge is the shared basis to decide corrective operations that can modify the perceived QoS (re-negotiation, additional communication channels, …), according to a strategy

either required by the VM user in her user profile or decided by VM administrators depending on the user role;

- *security*; the MA paradigm introduces not only specific security mechanisms and policies to deal with the execution of possibly untrusted hosted code, but also integrates widespread and standard solutions for secure service provision at the application level. For instance, agent operations on information and system resources are controlled depending on the permissions recognized to their authenticated principals and associated with their proper role [16]. Based on these security mechanisms, any operation can be permitted, registered and accounted to the requesting user depending on her role in the VM infrastructure;
- *interoperability*; agents work in an open global scenario and should interoperate with existing components, from legacy systems to existing and standard Internet services. Many MA systems achieve interoperability via compliance with diffused standards in the distributed object area, such as CORBA, in the MA-specific research area, such as MASIF (the OMG standard for interoperability between heterogeneous MA platforms [10]) and FIPA (the standard specification proposal for agent communication languages [8]) [13, 2], and in access interfaces for distributed information systems, such as ODBC and JDBC [20, 22].

Several research activities have shown the suitability of the MA technology to provide the above properties [23, 26]. As it is natural whenever a new technology is in its infancy, the largest part of the research efforts has been directed to the implementation of basic mobility mechanisms and MA platforms. Notwithstanding a general agreement on the opportunity of mobile agents to support access, filtering and retrieval of Web distributed information, only a few projects have already reached interesting results in this application domain. They mainly address the specific issues of reducing the communication cost by exploiting the possibility of mobile agents to filter distributed information locally to data servers [25]. Others stress the development of frameworks in the Web databases application area to assess and validate the MA approach in terms of achievable performance [24]. To our knowledge, no specific work has experimented yet the MA approach in the particular arena of virtual museum information.

4 The SOMA Architecture

The main guideline of the SOMA design has been to implement a modular and extensible Distributed Processing Environment (DPE) [14] that permits the rapid prototyping and deployment of distributed services. A particular feature of the SOMA DPE is its *openness*: SOMA has been designed and implemented to permit its full integration with other DPEs, whether MA-based or not, via CORBA compliance. In that way, SOMA service developers can realize service components that are not tied to use only the SOMA facilities. Service components may choose the most suitable implementation of the needed functionality, provided by whichever DPE, depending on the requirements of the specific application domain. SOMA has already demonstrated its suitability for the development of applications in several fields, e.g., user and terminal mobility, multimedia distribution, network and systems management [3, 4]. Usability, accessibility and manageability are the main attractive features of the SOMA framework, that facilitates the use of the environment to first-time users, but can also assist skilled developers in the deployment of complex distributed services.

The SOMA facilities are structured on two levels (Upper Layer Facilities – ULF - and Lower Layer Facilities – LLF; see Figure 2) and offer the fundamental functionality for the realization of any distributed application in open, global and untrusted environments such as the Internet. The SOMA DPE LLF include:

- ***Agent Communication Facility (ACF)***; the ACF provides mechanisms and tools to simplify coordination and communication between possibly mobile entities. Agents in the same execution node interact by means of shared objects, such as blackboards and tuple spaces; local access to resources is regulated by agents controlling authorization and enforcing the corresponding security policy. Agents that need to share resources with other remote agents are forced to migrate; outside the scope of the node, in fact, agents can only coordinate via message passing, and messages are eventually delivered even in case of agent migration.
- ***Agent Naming Facility (ANF)***; the ANF permits to identify dynamically any entity in the system, i.e. (possibly mobile) agents, users, terminals and resources. The ANF is implemented in terms of coordinated agents, and can interoperate with several naming standards, e.g. DNS-, CORBA-, and LDAP-compliant naming services [1, 12, 21]. The ANF is based on the association of a globally unique identifier with any SOMA public entity, making possible its dynamic tracing. Frequently moving entities have fixed care-of agents that keep updated the entity current position information; entities with lower mobility degree are traced by a more traditional and hierarchical register service.
- ***Agent Migration Facility (AMF)***; the AMF gives service designers the possibility to simply reallocate network resources and service components at run-time. Entities capable of reallocation are represented by agents, that can move in the network either via SOMA native migration methods or via standard specifications, such as CORBA Internet Inter-ORB Protocol [21] and MASIF [10].

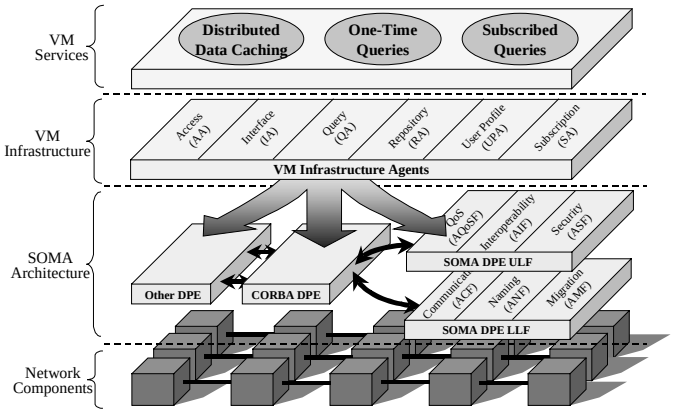

Figure 2. The SOMA architecture for VM services.

The SOMA ULF makes extensive use of the functionality provided by the lower layer of facilities. On this basis, they provide:

- ***Agent Interoperability Facility (AIF)***; the AIF permits interoperation among different resources and service components, by closely considering conformance with the CORBA and MASIF standards. For instance, the AIF simplifies the integration of legacy data resources (independently of the programming style

of their implementation) in the VM infrastructure via encapsulation with CORBA interfaces.

- ***Agent Security Facility (ASF)***; the ASF ensures the protection to any SOMA entity with a wide range of mechanisms and tools for authentication, authorization, privacy, integrity, and fine-grained access control to resources and service components depending on user recognized roles. The security framework is based on standard security providers for cryptographic algorithms (IAIK [15]) and diffused certificate infrastructures (Entrust PKI [6]).
- ***Agent QoS Facility (AQoSF)***; the AQoSF is in charge of application-level QoS monitoring and adaptation functionality. The monitoring module has the duty of observing the interesting properties of single or aggregated resources, e.g., local/domain disk free space and currently available network bandwidth. It exploits platform-dependent mechanisms to obtain the monitored indicators (e.g. the `psapi.dll` on Windows NT and `pstat` on Unix). It invokes this functionality provided by the hosting operating system via the Java Native Interface [7]. Any authorized SOMA component can access the monitored properties to decide a strategy for adapting to the current environment conditions. In addition, the monitored QoS parameters serves in evaluating and registering user resource consumption towards accountability. It is possible to associate different costs to different QoS data resources, and to account users not only depending on the size of the received data but also on the resource consumption to perform the requested queries.

Mobile agents are typically *location-aware* entities [17], and SOMA provides a set of locality abstractions suitable for describing any kind of internetworking scenario. Any node in the system owns at least one *place* that constitutes the agent execution environment. Several places can be grouped into a *domain* abstraction that corresponds to the network locality. In each domain, a *default place* hosts a gateway to perform inter-domain functionality.

Further details about the SOMA programming framework and its implementation are presented elsewhere [2, 3] and are out of the scope of this paper that concentrates on describing the distributed infrastructure for the VM support, which is built on top of these two general facility layers as depicted in Figure 2.

5 THE VM INFRASTRUCTURE

We have designed and implemented a specific MA-based infrastructure to support the provision of VM services. The complexity of the VM infrastructure is completely hidden to its users who operate via friendly interfaces, either provided by a specific type of mobile agents (the *Access Agents*, see the following section) or integrated within standard Web pages. The physical localization of data resources is generally transparent; the only knowledge required to users who have no managing duties is about the logical names of resources. In addition, users can specify profile information at their first registration, and can modify it at any time. Profile data are maintained by *User Profile Agents* (see next section) and can also be replicated in several nodes of the VM infrastructure for increased reliability and efficiency. User profile preferences are the key mechanism to permit user subscriptions and to inform the infrastructure of information pre-fetch and cache opportunity.

In the following, we first present several types of MA-based components that implement the distributed VM infrastructure. Then, we briefly describe a concrete usage scenario that shows how the VM organizes museum information when dealing with different levels of complexity: when the functionality requirements of the scenario increase, the VM infrastructure is capable of offering increasingly complex and powerful features.

5.1 VM Components

The main guidelines in the implementation of the VM infrastructure are:

- to overcome the heterogeneity problems due to the intrinsic open nature of the considered data. To realize a completely open framework, we have followed the principles of data resource encapsulation, and we have pursued the compliance with the most accepted data access standards (JDBC, ODBC, and CORBA access in case of legacy data resources [20-22]);
- to improve the efficiency and the effectiveness by the realization of a persistent information infrastructure that is geographically distributed over the Web and that accommodates user-subscripted queries, i.e. queries that one user is permanently interested in. The VM infrastructure can maintain distributed cached results for subscripted queries and can automatically update this information at predefined intervals, as specified in the user profile information. This can significantly reduce the on-line connection time needed by the user since she has only to receive the already retrieved query results.

The VM infrastructure results from the interaction of several MA-based components. Their specification, design and realization has been simple and rapid because of the modular design of SOMA and its implementation in the Java language [7]. In addition, the object-oriented approach makes simple successive refinements of VM components via sub-classing and allows their reuse in other projects and application areas. The main categories of mobile agents that compose the VM infrastructure are access, interface, repository, query, user profile and subscription agents.

The ***Access Agent*** (AA) is responsible for accepting queries from authorized users. We introduce different types of AA tailored to the recognized different user roles: manager, expert and normal visitor. Normal visitors can only submit simple queries and explore the provided results with no possibility of specifying a required QoS level on reception. Experts can also subscribe for query repetition and are accounted on the basis of both the VM resource consumption and the obtained QoS. In addition to the functionality available to experts, managers can modify the VM data under their responsibility, e.g., to add dynamically a link to a new publication about a particular artifact. AAs are in charge of authenticating the connected user and associating her with the proper role, by exploiting the underlying SOMA security facility. Then, they collect the requested query, decide how to answer the query by coordinating a group of query agents, control the work progress up to its completion, and finally yield back the results to the user according to the corresponding user profile information. The visualization of the query results, which possibly consist of a set of multimedia data encoded in different representation formats, currently exploits both the integration of the existing legacy interfaces shown in Section 2 and standard available visualization tools, e.g., Web browsers with RealPlayer plug-in, that are supposed to be already installed at clients. However, this is not a limiting constraint because SOMA provides specialized agents to automate dynamic software installations [3]. The default system

configuration provides one AA for any client host, but the infrastructure permits to dynamically create new AAs and to migrate them wherever the traffic of service requests justify a multiplicity of agents to overcome possible congestion and bottlenecks. We are testing several AA distribution policies: we are evaluating solutions where one AA can handle from 1 to j different queries, from 1 to k different users (*AA-per-query/user*); we are also measuring the performance under different traffic conditions for a pool of n AAs already running to receive user queries on any client host, if compared with the instantiation of a new AA at any new user query. Any of the aforementioned AA comes in two flavors depending on the fact it implements or not the Web-enabled CORBA server interface (see Figure 3): it can either run on SOMA places active on client hosts, or reside on a Web server and be accessed via a CORBA-client applet integrated in a standard Web page.

The ***Interface Agent*** (IA) is the stationary entity that encapsulates information resources by abstracting from their specific implementation details. It represents the key component to overcome heterogeneity in data resources, and provides a uniform interface to the other components of the VM infrastructure. Its implementation is different depending on the particular characteristics of the encapsulated data resource. We have already implemented two specialized IAs for data resources that own JDBC and ODBC access interfaces (see Figure 3). We are currently working on the encapsulation of legacy data components via CORBA interfaces and on their integration in the VM by exploiting the SOMA interoperability facility. In addition, the IA enforces the security policies for data resource access: it receives user authentication information from the AA, and registers resource consumption for accountability. The IA is always placed locally to the data resource it encapsulates. As a consequence, apart from its initial dynamic code distribution, the IA is mainly a stationary agent; however, it can move to follow the possible migration of the encapsulated data resource (e.g., the transfer of a database on a new server).

The ***Repository Agent*** (RA) is the stationary agent in charge of organizing the data resource name service for the whole system. Typically, we have one RA in each SOMA domain. After receiving a query, any AA interrogates its RA to discover the physical locations of the data resources involved. The RA resolves the logical names of the resources into the corresponding physical locations of the IAs. The information maintained by a single RA does not cover all the data resource names in the system. RAs are hierarchically organized: one RA that is not able to respond to a name resolution request simply forwards it to its higher-level RA in the hierarchy, and waits for an answer in a way that is completely analogous to the resolution of logical IP names in the DNS [1]. In addition, in case of replicated data resources, the RA chooses the IA among a group of equivalent ones according to locality and load-balancing evaluations. At the moment, VM agents interrogate RAs via a SOMA proprietary directory protocol. We are extending this implementation to increase standardization and portability of the repository service without affecting its performance; in particular, we are terminating the integration of an additional interface for our RAs compliant with LDAP [12].

The ***Query Agent*** (QA) is the mobile entity that really performs user-specified queries on data resources. After one user inputs a query, expressed in a subset of the SQL language, the AA parses it and interrogates the RA in its domain to discover which data resources are interested by the interrogation. At that point, on the basis of the query structure and of data localization, the AA decides the number of QAs to generate. Any query is always performed by a group of coordinated QAs that exploit the communication facility provided by the SOMA platform. We have already implemented several QA refinements (see Figure 3). A *master QA* is the query-responsible agent that coordinates the other agents (*slave QAs*), puts together the overall result and yields it back to the interested user. Slave QAs, instead, calculate the results of independent partitions of the original query; slave QAs are normally instantiated and distributed one for each SOMA domain to be explored. Finally, in case of join operations, *slave join QAs* have the opportunity to delegate sub-queries to new slave QAs, generated on-the-fly, and to wait for merging the corresponding sub-results.

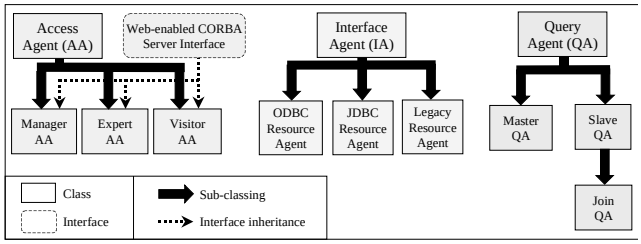

Figure 3. Class and interface relationships for the AA, IA and QA components.

The ***User Profile Agent*** (UPA) is the mobile agent responsible for maintaining the preferences of usual customers, i.e. experts and managers. It collects information about the characteristics of the most commonly used terminals (e.g., graphic resolution), the usual attachment points to the network, and the expert subscripted queries with the related general preferences (e.g., to discard animations from query results if they overcome predefined thresholds). The UPA is generally located at the user home, i.e. the place of the user first registration, but it is able to migrate to follow possible user movements (*personal mobility*). The distributed location of UPAs makes possible a good decentralization of the load of user profiling. The limitation due to the fact that any recognized user has one personal active UPA in the VM infrastructure is not severe because UPA agents normally do not waste system resources, but simply wait in an idle state until they are waked up by the reconnection of the corresponding user.

The ***Subscription Agent*** (SA) automatically maintains and updates query results that are still interesting for VM users, because of subscription or sharing potential. Distributed SAs are the crucial component to provide the distributed caching functionality within the VM infrastructure. At the moment, we have realized a first simple SA prototype: any expert-subscripted query has a dedicated SA that automatically re-executes the needed interrrogations at intervals specified in the subscription; when a user connects to the VM infrastructure, her AA directly interrogates the corresponding SA and simply collects the already retrieved results. We are interested in extending the SA functionality accordingly to two different directions: the former is to automatically trigger query re-execution when involved IAs notify the VM of a a significant data updating; the latter is to accommodate and optimize the case of different users interested in overlapping query results. For instance, the distributed replication of query results in the VM can significantly improve access performance and reduce overall network traffic in case of large volume data objects (e.g.,

high-quality raster images and animations) when the results are supposed to interest a multiplicity of independent VM users working in the same network locality.

5.2 The VM Infrastructure in a Possible Usage Scenario

The flexibility of the VM infrastructure also impacts on the possibility to provide differently complex levels of functionality to users with different degrees of expertise and different expectations from the service. While a normal visitor is likely to perform simple queries, an expert is a usual museum customer and pays for several pending subscriptions.

When a normal visitor accesses the VM infrastructure, the AA authenticates her via standard certificates, requests her profile information to the corresponding UPA at the user home, and interrogates the domain RA to know the location of the data resources involved in the query. In the case of a simple query on the data resources of a single domain, the AA creates one QA and sends it for the whole search. The QA collects the query results and comes back to its launching AA, which communicates the results to the corresponding user.

The effectiveness of the VM framework becomes more and more evident when subscripted queries complicate the usage scenario and make possible to exploit distributed caching (see Figure 4). When one expert user requests a new query subscription, her AA creates a dedicated SA in the local domain. It is the SA that asks the local RA for data resource name resolution, and that creates, coordinates and collects the query results from a pool of QAs (normally one in any interested SOMA domain). In addition, the SA is able to command the repetition of the query, triggered by the corresponding UPA at the time intervals specified. The accounting information on data resource usage is registered at any involved IA, and the overall cost of the subscripted query is maintained by the SA. Any time the expert user of a subscripted query accesses the VM service, the AA directly connects her to the corresponding SA to immediately download the updated query results with no need for the user to wait for the query execution.

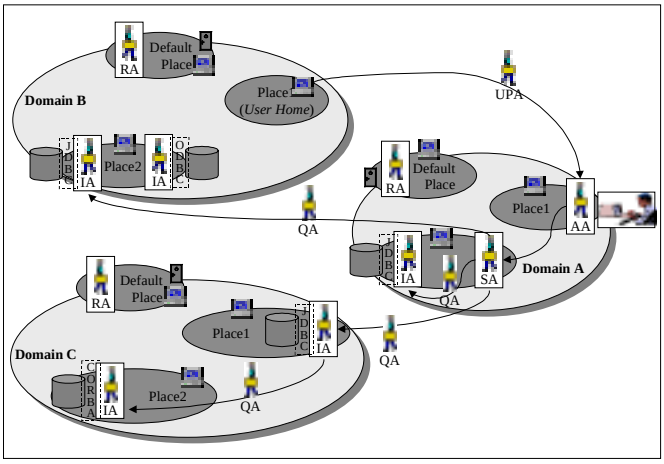

Figure 4. The VM agents in a possible usage scenario.

6 FUTURE WORK

The first experiences in deploying museum information services within the VM framework have shown not only the feasibility of the approach but also its effectiveness: the modular design of both SOMA and VM components have allowed to simply extend and refine the implemented services depending on the feedback from the test-bed experience; the first concrete tests of the infrastructure exhibit significant performance improvements, especially when exploiting the distributed caching infrastructure and when dealing with geographically distributed data servers interconnected by networks that are highly heterogeneous in their bandwidth characteristics.

At the moment, we have extensively tested the VM prototype only on a small subset of the museum information (3 Web sites with about 100000 images and inventory forms) involved in the MOSAICO project (about 300 data servers with ten million objects will be included in the VM before the end of next year). We have now started the integration of the complete test-bed of data servers and we expect other interesting feedback, especially from the point of view of scalability.

In addition, we are working on the extension of the VM infrastructure to accommodate complete mobility of users, terminals and data resources: user mobility is already supported by the current VM prototype via the UPAs; we are implementing the distributed support for the run-time dis/connection of terminals from/to different points of attachment to the network with no need to suspend service provision; data resource mobility is under investigation, and we are collecting VM performance under different network traffic conditions to measure which cases justify the migration of data servers (and of corresponding IAs), e.g., when some databases residing on domain A are accessed very frequently by users of domain B.

Moreover, the current VM prototype still needs implementation work to extend its management functionality and browsing facilities: on the one hand, the management operations are currently realized in a traditional client/server way by exchanging commands between the AA and the IAs and exploiting the proprietary functions of the data servers involved, while also management can significantly take advantage of an MA-based implementation; on the other hand, the VM is under extension to allow QAs autonomous migration to follow dynamically determined Web links.

Finally, we are evaluating the possibility of introducing a uniform data representation that is flexible enough for the open nature of the involved data. Data representation languages that are emerging in archive and museum Web catalogues, such as XML [19], and interoperable standard efforts for agent communication, such as FIPA [8], are under consideration.

7 ACKNOWLEDGMENTS

Work carried out under the financial support of the Italian Ministero dell'Università e della Ricerca Scientifica e Tecnologica (MURST) in the framework of the Project "MOSAICO: Design Methodologies and Tools of High Performance Systems for Distributed Applications".

8 REFERENCES

[1] Albitz, P., and Liu, C. DNS and BIND, 3rd Edition. O'Reilly & Associates, Sep. 1998.

[2] Bellavista, P., Corradi, A., and Stefanelli, C. A Secure and Open Mobile Agent Programming Environment. ISADS99, Tokyo, Japan, Mar. 1999.

[3] Bellavista, P., Corradi, A., and Stefanelli, C. An Open Secure Mobile Agent Framework for Systems Management. Journal of Network and Systems Management, Vol. 7, No. 3, Sep. 1999.
[4] Bellavista, P., Corradi, A., Stefanelli, C., and Tarantino, F. Mobile Agents for Web-based Systems Management. Internet Research, MCB University Press, Vol. 9, No. 5, Nov. 1999.
[5] Bolliger, J., and Gross, T. A Framework-Based Approach to the Development of Network-Aware Applications. IEEE Trans. Software Engineering, Vol. 24, No. 5, May 1998.
[6] Entrust Technologies. Entrust. http://www.entrust.com/.
[7] Flanagan, D. Java Power Reference. O'Reilly & Associates, March 1999.
[8] Foundation for Intelligent Physical Agents. http://www.fipa.org/.
[9] Fuggetta, A., Picco, G.P., Vigna, G. Understanding Code Mobility. IEEE Trans. Software Engineering, Vol. 24, No. 5, May 1998.
[10] 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/pub/docs/orbos/, 1998.
[11] Gribble, S.D., and Brewer, E.A., System Design Issues for Internet Middleware Services: Deductions from a Large Client Trace. USENIX Symp. on Internet Technologies and Systems, USA, 1997.
[12] Howes, T., and Smith, M. LDAP: Programming Directory - Enabled Applications with Lightweight Directory Access Protocol. Macmillan Technical Publishing, Jan. 1997.
[13] IKV++ GmbH. Grasshopper, http://www.ikv.de/products/grasshopper/.
[14] Inoue, Y., Cuha, D., and Berndt, H. The TINA Consortium. IEEE Communications, Vol. 36, No. 10, Sep. 1998.
[15] Institute for Applied Information Processing and Communications. IAIK JCE. http://jcewww.iaik.tu-graz.ac.at/.
[16] Lupu, E.C., and Sloman, M. Towards A Role-based Framework for Distributed Systems Management. Journal of Network and Systems Management, Vol. 5, No. 1, Mar. 1997.
[17] Maass, H. Location-aware Mobile Applications Based on Directory Services. Mobile Networks and Applications, Vol. 3, No. 2, pp.157-173, 1998.
[18] McFall, C. An Object Infrastructure for Internet Middleware. IEEE Internet Computing, Vol. 2, No. 2, pp. 46-51, Mar. 1998.
[19] Michard, A., Pham-Dac, G. Descriptions of Collections and Encyclopaedias on the Web using XML. Archives and Museum Informatics, Vol. 12, 1998.
[20] North, K., Understanding ODBC 3.0 Standards and OLE DB. DBMS, Vol. 9, No. 4, Apr. 1996.
[21] Object Management Group. CORBA/IIOP Rev 2.3. OMG Document formal/98-12-01, http://www.omg.org/library/, Dec. 1998.
[22] Quan, X., Ling, F., and Hongjun, L. Supporting Web-based Database Application Development. 6th IEEE Int. Conf. on Database Systems for Advanced Applications, USA, 1999.
[23] Rothermel, K., and Hohl, F. (eds.). Mobile Agents. 2nd Int. Workshop (MA'98), Springer Verlag, Germany, Sep. 1998.
[24] Samaras, G., Dikaiakos, M., Spyrou, C., and Liverdos, A. Mobile Agent Platforms for Web-Databases: A Qualitative and Quantitative Assessment.
[25] Theilmann, W., and Rothermel, K. Disseminating Mobile Agents for Distributed Information Filtering. 1st IEEE Int. Symp. on Agent Systems and Applications, USA, Oct. 1999.
[26] Vitek, J., and Tschudin, C. (eds.). Mobile Object Systems: Towards the Programmable Internet. Springer Verlag, Berlin, 1997.