Web-Assisted Visits to Cultural Heritage

Giacomo Cabri, Letizia Leonardi, Franco Zambonelli
Dipartimento di Scienze dell’Ingegneria – Università di Modena
Via Vignolese, 905 – 41100 Modena – ITALY
E-mail: {giacomo.cabri, letizia.leonardi, franco.zambonelli}@unimo.it

Abstract
The wide access to cultural heritage is gaining more and more interest from the information technology community. Museums, exhibitions and cultural places are becoming enterprises that supply new services to access cultural information, and this calls for suitable infrastructures and tools. In such a context, this paper proposes a distributed application to enable virtual visits, based on an infrastructure exploiting innovative technologies such as active proxy servers and mobile agents. Such application is integrated within the Web and permits to build up virtual visits that can be attended by groups of people interested in the same subject. Such visits are tailored on the basis of users’ profiles and devices, and can be enhanced by the availability of distributed services. This grants a high degree of flexibility and autonomy for our application.

1 Introduction
Cultural heritage represents an estate that must be preserved and exploited in the best way. In particular, Europe holds by far the largest volume of objects and information on our cultural heritage in the world, and for most European countries tourism is one of the most important sources of richness.

The management of cultural places calls for new efforts to fruitfully exploit the existent estate. New industries will arise in this area carrying new employment opportunities. Museums, exhibitions and cultural places in general can be thought as enterprises that have to promote and sell their contents and their services. Besides the management of real visits, they must face also other media and channels to spread culture, adding value to the simple digitalization of images, sculptures, monuments and places.

Currently, the divulged cultural patrimony is fragmented, not very structured, and managed by single institutions in a non-collaborative way. It is important therefore to create synergy and activate thematic discourses among the various institutions that are the promoting actors. Then, collaboration among the different sources providing cultural information is needed. Such collaboration must be supported by technologies and tools that exploit the existing standards and have a high degree of flexibility.

The Web is one of the candidates to spread the access to cultural heritage. While few years ago people surfed the Web only to search for information, now they are more and more becoming aware of the capabilities of the Web in terms of offered services. Many people discover how the use of the Web permits to save time even in performing simple tasks. Moreover, the Web gives a world-scale spectrum of possibilities among which to choose the preferred one, letting people compare and make the best decision.

The aim of this paper is to present an application that enables virtual visits to cultural heritage via the Web, collecting information from different sources and permitting group visits. Distributed approaches in the area of cultural heritage have already been proposed [16] and focused on the capability of using a standard visualization protocol to display images and animation of cultural elements in Web pages. We propose a deeper integration with the Web and more flexibility that leads to the capability of customization of the accesses on the basis of different criteria.

To this purpose, we implement this application exploiting PROOF v2 [7], a modular infrastructure that permits heterogeneous interactions and collaborations on the Web [1, 8]. It is based on the concept of active proxy server, which permits to enhance Web interactions by adding computation, without requiring the modification of existing Web servers and browsers. Moreover, it includes the capability of hosting (possible mobile) agents, which can rely on an advanced coordination means to coordinate their activities. All these features of PROOF v2 are exploited to provide users with flexible collaborative capabilities.
2 The PROOF Infrastructure

The main aim of PROOF (simply PROOF in the following) [7] is to provide a powerful and flexible means to enrich the Web with computational capabilities for CSCW (Computer Supported Cooperative Work) without requiring significant modifications to current servers and clients. To these purposes PROOF exploits active proxy servers, mobile agents and coordination technology (see Figure 1).

While traditional proxy servers are mainly used to provide cache functions, PROOF is much more flexible and can embody several different functionalities. For example, PROOF can visualize on-line the current and/or past activities of the workgroup, and can integrate a chatting system or other kinds of inter-workgroup communication facilities. Synchronous interactions between the clients and the proxy server can be enabled by letting PROOF insert specific applets into the pages that it provides to browsers, thus enabling communication among people in the workgroup via the proxy server.

![Diagram of the PROOF infrastructure]

**Figure 1. The PROOF infrastructure**

Distributed applications can rely on several PROOF proxy servers, which can coordinate each other and also constitute a federation of proxies.

PROOF is based on a modular infrastructure, composed of a framework and several application modules. The framework provides the basic proxy functionalities (such as the connections with the client browsers and with the Web servers, the user identification and authentication) to be enhanced by installing specific application modules. The framework can also host the execution of mobile agents [14, 15] and integrates an advanced coordination architecture based on programmable tuple spaces: this permits to define reactions to adapt the tuple space behavior to specific application needs [10].

PROOF is written in pure Java, and this permits a high degree of portability. It exploits the Java mechanisms of interfaces and reflection to implement the framework and to permit an easy and dynamic installation of application modules into the framework.

Each application module implements the behavior of (a part of) one specific application. PROOF enables the installation of several modules at the same time, which permits both one single proxy server to be exploited by several clients (browsers or agents) for different applications and the decomposition of one application in smaller modules. Programmers can also use coordinated mobile agents and reactions in their applications. In addition, mobile agents, on behalf of users, can install specific modules to give the proxy server an application-dependent behavior. Since agents are autonomous, the user can give them a possibly high-level task to carry out, and they can proactively search for the needed module(s), perhaps by mean of negotiation with module providers; once they have found the module(s), they can search for the most appropriate – also in terms of costs – PROOF proxy server where to install the module(s). The user is then notified of the PROOF proxy server (s)he have to use in order to exploit the needed functionalities. As a further advantage, the use of mobile agents permits to give other entities – besides people – the access to collaborative Web applications. Users can think of not participating directly to an interactive application that requires repetitive actions; instead, they can rely on one or more agents that act in behalf of them. In this context, the exploitation of mobile agents can make the PROOF infrastructure very open and flexible. Moreover, the mobility capability is recognized as very useful in the implementation of distributed applications, because of saving bandwidth, of letting agents move locally to the resources they need and of dealing with non-continuous network connection.

The coordination architecture embodied in PROOF is based on programmable tuple spaces, which are powerful coordination means to suit Internet applications. The concrete coordination system is MARS (Mobile Agent Reactive Spaces) [6], which is based on an enhanced Linda-like model [13]. MARS is JavaSpaces [12] compliant and implements tuples as Java objects that represent ordered sets of typed fields. MARS defines five

---

1 PROOF v2 denotes the second version of PROOF; the first one [4] did not have mobile-agent and coordination capabilities.
Linda-like operations to read, write and extract tuple from/to the tuple space. MARS overcomes the limits of the static data-oriented Linda model, by adding programmable reactivity, which means that specific actions can be programmed in response to the accesses performed by agents on the tuple space [5].

3 Assisted Virtual Visits

The PROOF infrastructure is very general and it is not tightly bound to any specific application. Different application modules can be implemented with a limited coding effort and easily installed within the proxy-framework.

This section presents a specific distributed application that we have implemented to enable virtual visits to cultural heritage. In particular, the application consists of a module, called Virtual Visit Module (VVM), which can be loaded in different PROOF proxy servers, as well as of different mobile agents classes to exploit in information search and in proxy coordination.

3.1 The Application Architecture

The architecture on which our application is based is very general, and can be exploited by a huge class of distributed and collaborative applications, which have requirements similar to the ones presented here.

In our application scenario we assume that each cultural place – intended as a place that contains cultural heritage elements – is provided with a standard Web server that makes information available. Such information can be of different kinds, and in particular related to the description of the elements that can be visited. As explained in the next subsection, we assume that the information is stored using an appropriate standard – the Dublin Core – that permits to describe several historical and artistic issues, to allow detailed visits. Moreover, the Web servers should make available also graphical information that is used to present images and animations to the user.

The application can exploit the presence of the VVM in several PROOF proxy servers, which constitute a federation of proxies, to distribute the load and to make special-purpose services available. For the load-balancing aim, a trivial solution is to provide several “points of access” in the form of several PROOF proxy servers, delegating the choice of the preferred one to the user. In a more sophisticated solution, a well-defined PROOF proxy server can be in charge of accepting requests, finding the best proxy server, and then setting the user’s browser to use such PROOF proxy server, for example by exploiting agents. With regard to special-purpose services, we suppose that the PROOF proxy servers differentiate each other by the available services, and agents are in charge of searching the PROOF proxy server that has all the services required by the user or has the best trade-off between costs and provided services.

![Figure 2. The application architecture](image)

The Virtual Visit application works as shown in Figure 2: users configure their browsers to use a PROOF proxy server where a module for the virtual visit has been loaded (step a in Figure 2). As soon as a user makes a request, a login page is provided to identify and authenticate her/him. At the first connection, the user can select some options to configure her/his profile. Such options relate to the degree of experience of the user, the language in which the information will be presented (when multilingual information is available), the connection speed and the required details for information. In addition, the browser is checked about the availability of plug-ins, to know which formats of information are supported. In future connections, the profile of the user is searched and retrieved in the federated proxies.

Moreover, the user is asked if (s)he wants to join a group, to perform a visit together with other people. The simplest case is when the people that want to constitute a group connect to the same PROOF proxy server, as the two clients shown in the top-left corner of Figure 2. A workgroup is created locally to a server and a session is started; all users belonging to this workgroup make the same virtual visit, which happens as described in Subsection 3.3.

A more complex case is when users are connected to different PROOF servers. This may happen for several
reasons: for instance, users could be geographically distribute and each one may choose the PROOF server nearest to her/him, in order to achieve more efficiency; otherwise, each user could have her/his profile on a given server, and wants to connect via such server; finally, users could not be aware of the distributed nature of the application, and they simply connect to one PROOF server. The application provides its services in a transparent way, so no matter where the users connect, they can create a workgroup and start virtual visits; this implies coordination issues, in particular the need of exchanging information about the status of the visit and the performed actions. We exploit mobile agents to coordinate users belonging to the same workgroup on different PROOF servers. At the beginning, each PROOF server sends one agent to other servers of the federation, in order to find out the PROOF servers to which the other users of the workgroup are connected. Then, each PROOF server keeps a table with the addresses of the other PROOF servers that are involved in the workgroup. Such table can be dynamically updated as users join or leave the workgroup. When a change in the status of a server occurs, one or more agents are sent to the servers listed in the table, to inform them about the occurred changes (step c in Figure 2).

An advantage of exploiting agents is that they can be instructed to monitor and report changes in a smart way. First of all, they must report only those pieces of information that may affect the navigation of other users (for example, the information about the graphical capability of a user's hardware does not influence the other users, so it must not be reported). Then, they can negotiate with the destination servers the interesting changes (e.g., new hosts with interesting information) and can deal with possible conflicts.

3.2 Data Representation

We suppose that the art works and their digital reproductions are described using Dublin Core records. The Dublin Core Metadata standard (DC in the following) is a fifteen-element metadata set, originally conceived for author-generated description of Web resources [11]. Recently it has gained the attention of formal resource description communities such as museums and libraries due to the fact that: (i) it is useful to describe artifacts and associated information resources in the museum community and (ii) it is particularly simple to learn and easy to implement (using a basic XML syntax) [9].

Cultural heritage information available through the Internet includes mainly multimedia data like texts, images, video and audio files that describe art works in permanent collections and temporary exhibitions [18]. Moreover the advent of virtual reality Web technologies (like Quick Time VR and Virtual Reality Modeling Language) enables the user not only to access multimedia data, but also to visualize and interact with 3D objects reproducing art works, galleries, museums, churches and other cultural monuments and to do virtual walkthroughs. This can be considered very interesting in order to visualize sculptures, buildings, and archaeological finds in which concepts like real volume and interactivity with the user are very important [3].

DC enables to describe in a standard way all these different types of cultural resources like buildings, paintings, sculptures, and art works in general [2]. In our case, the use of the DC standard grants a high degree of interoperability and the fact that it describes the content of resources enables the VVM to allow the user to obtain cultural heritage information tailored on the basis of her/his own interests:

- enabling the creation of a collection of information coming from different sites;
- suggesting cultural paths to explore cultural heritage contents distributed on the Web;
- providing the exploitation of related services.

The DC information retrieved during a visit is locally stored as MARS tuples; this grants high flexibility and uncoupling in the management of the retrieved information.

3.3 User Interface and Resource Displaying

After the login, the VVM inserts the control applet shown in the box B of Figure 3 in all the pages sent to the user’s browser, and takes control over them. The applet interface is quite similar to a search engine, and this makes familiar to the user the request for information. However, there are several differences with regard to a search engine:

1. the VVM does not simply report (possibly thousands of) found links to the user, but it builds up a visit, also taking into consideration the user’s profiles;
2. the search is based on the DC Metadata standard, which permits searches on the basis of the content of documents;
3. the interface for the navigation is the same for all the found resources, so all the collected pieces of information are presented in a uniform way;
4. the application does not require continuous network connections, since it relies on mobile agents.

The control applet presents several options (selectable via the buttons in the top of the applet) to configure the user profile, the visualization of the resources, the hosts where to search, and so on.

Mobile agents are exploited to delegate the operation
of retrieving interesting information. When the user presses the START button on the bottom-right of the applet, the VVM sends mobile agents to the sites that are likely to contain information related to the user's request (step b in Figure 2). Agents can report the pieces of information in an asynchronous way, thanks to the uncoupling provided by the tuple space; this lets the user start the visit as soon as some information is available, and increasing the available information collected by agents returned to the server.

By means of our application, the user can look for art works related to specific keywords, historical periods, geographic locations or authors (all these data are specifically described by DC metadata elements). The VVM enables the user to customize also the presentation of original or surrogate resources descriptions, visualizing only a selection of the fifteen DC metadata elements. The result of these interactions is a tailored collection of information that constitutes the virtual visit where the required information can be found, also on the basis of further elaborations.

In order to provide the described features, the VVM:
1. selects the DC records reported by mobile agents from different sites on the basis of the user's interests and choices;
2. builds customized XML files on the basis of the DC subset selected by the user;
3. visualizes them using appropriate XSL style-sheets.

The VVM collects the required information and generates a page for each retrieved resource, and sends it to the user's browser (see Figure 3). Three frames compose this page.

The upper frame (box A in Figure 3) contains the description of the visualized resource, adapting the related DC document to the user profile. This description is supplied as XML document, if the browser supports it; otherwise, it is converted in an HTML page by the VVM via an appropriate XSL style-sheet. This frame contains also a list of Web sites related to the shown resource, specified in the Relation attribute of the DC standard.

In the middle of the page there is the main frame (box B in Figure 3), the one containing the control applet and the found resources. If a resource is not an HTML page, the module dynamically builds a frame to "wrap" the control applet and the resource itself.

The bottom frame (box C in Figure 3) contains two functionalities. The former one permits the user to express a feedback about the current virtual visited resource; this influences the rest of the virtual visit, in terms of both the content and the quality of the rest of displayed resources. The latter one can be exploited to ask for commercial information about both goods (such as books, catalogues and gadgets) and services (such as booking, accommodation and transports); these requests can be tailored also on the base of an available budget. Then, the requested commercial information is sent to the user off-line via email.

During the virtual visit, the users belonging to a group can communicate exploiting the chat capability built-in in PROOF. It permits to send a message to one user or to all users belonging to the same group. Users can exchange information, personal opinion, and so on, in order to simulate a visit all together even if they are spread over the world.

Another PROOF capability that can be useful in virtual visits is master-slave browsing. The super-user of the workgroup assumes the role of master. When (s)he browses, all the other users (slaves) connected to the workgroup are forced to visit the same pages. The control applet inserted in Web pages gets notified of page changes in the master’s browser. When a master’s change occurs, the control applet forces each slave browser to load the same page of the master, without requiring any intervention from the slave users. This is particularly useful when there is a guide that illustrates the visited places. This can happen for instance in a classroom.
where a guide teaches the students, which can follow the
guided tour from their own browsers.

4 Conclusions and Future Work

In this paper we have presented a distributed
application that enables virtual visits to cultural heritage.
This application is based on the PROOF infrastructure,
exploiting mobile agents to grant a high degree of
flexibility and collaboration. The collaboration can occur
at three different levels: institutions can agree about
standards and make their information available to create
virtual visits; PROOF proxy servers can federate to
balance the load or to provide peculiar services; users
can constitute workgroup to perform virtual visits
together with other people. We think that the concepts
and the mechanisms exploited in this application can be
extended to other kinds of collaborative applications that
have similar requirements.

For future work, we outline some directions.
Currently, the VVM records the profile of the users,
but does not keep track of the visited places or pictures;
recording also this information could be useful to propose
cross-references during visits and to permit users to
disconnect and reconnect later to resume a visit.

It should be taken into consideration that the PROOF
proxy modules consume resources of the server where
they are hosted. We can figure out that such services are
charged to the user, on the basis of the quality of service
the servers grant; moreover, the QoS can be negotiated, for
instance via mobile agents that attend auctions [17].

Another interesting field to be explored relates to
mobile devices. We envision that users will be provided
with personal devices such as PDAs, palmtops, or even
mobile phones, which help them in the visit to cultural
heritage. This scenario calls for modeling visit
applications that take into account the help these devices
can provide to user, also in connection with the virtual
visits performed via Web.

Acknowledgements

Work supported by Italian Ministry for University and
Science and Technology Research (MURST) within the project
“MUSIQUE: Infrastructure for QoS in Web Multimedia
Services with Heterogeneous Access” and by the University of
Modena and Reggio Emilia with a fund for young researchers.

References

Web as enabling technology for CSCW: The case of
BSCW”, in Computer-Supported Cooperative Work:
Special issue on CSCW and the Web, Vol. 6, Kluwer

Melotti; “Tailored virtual tours in Cultural Heritage

forimproving the knowledge of the history of a City”, in
Virtual Reality in Archaeology, (J.A. Barcelo, M.Forte,

Framework to Support Synchronous Cooperation on the
Web”, Software, Practice and Experience, Vol. 29, No. 14,

Coordination Models for Internet Applications”, IEEE

Programmable Coordination Architecture for Mobile
Agents”, IEEE Internet Computing, Vol. 4, No. 4, pp. 26-35,
July-August 2000.

Infrastructure for People and Agent Interaction and
Collaboration”, 9th IEEE International Workshops on
Enabling Technologies: Infrastructure for Collaborative
Enterprises (WETICE), NIST (USA), June 2000.

Knoche, “Coordinating Multi-Agents Applications on the
WWW: a Reference Architecture”, IEEE TSE, Vol. 24,

Information (CIMI), Guide to Best Practice: Dublin Core
(DC 1.0 = RFC 2413), final Version, 12 August 1999.

of a Language for Programmable Coordination Media”,

[11] Dublin Core Metadata Element Set, Dublin Core home

[12] E. Freeman, S. Hupfer, K. Arnold, JavaSpaces:

[13] D. Gelernter, N. Carriego, “Coordination Languages and
Their Significance”, Communications of the ACM,

Foundations, Applications, and Markets”, Springer-
Verlag, March 1998.

Agent Programming Systems”, IEEE Concurrency, Vol. 6,
No. 3, pp. 52-61, July-September 1998.

[16] W. E. Moen, “Accessing Distributed Cultural Heritage
Information”, Communication of the ACM, Vol. 41,
No. 4, pp. 45-48, April 1998.

for an Internet-Based Auction House”, IEEE Internet
Computing, Special issue on Agent Technology and the

Heritage and Knowledge”, The Global Village

361