Distributed Multimedia Workstation For Medieval Manuscripts

Sylvie Calabretto and Béatrice Rumpler
LISI - INSA de LYON
20, avenue Albert Einstein
69621 Villeurbanne Cedex
France
E-mail : {cala, rum }@if.insa-lyon.fr

Abstract

This paper presents the Web version of the philological workstation BAMBI (Better Access to Manuscript and Browsing of Images) and its design. The BAMBI Web workstation allows historians, and more particularly philologists, to work on manuscripts (transcription, annotations, indexing, etc.) alongside the Internet network. Our approach for the Web implementation of the philological workstation is based on the reuse of the local BAMBI software. We have used the formalism of UML (Unified Modelling Language) for the distributed software design, and especially the concepts based on the diagrams of collaboration.The implementation is based on the ActiveX approach.

1. Introduction

The BAMBI[1] project (BAMBI for Better Access to Manuscripts and Browsing of Images) for libraries aimed at designing a modern interactive tool for consulting and working on manuscripts [BON 97], [BOZ 97].
BAMBI is an electronic desktop on which it is possible to deal contextually with the original handwritten document and its transcription (see Figure 1). This contextuality means that students always have all the data they need to examine at their fingertips: the document together with its interpretation.
The manuscript and its transcription are 'linked' in BAMBI at the level of the page or, more frequently, the word, through an automatic system for segmentation of the image and searching for matches: a double click on the transcribed word is all that is required to find the original word on the manuscript, and *vice versa.*
The electronic form of the data processed by BAMBI offers many other advantages: the digitized image of the manuscript can be enlarged or reduced to the desired level of detail; it is possible to arrange documents on the screen in a flexible way; the transcription made does not remain an inert text of mere output but pilots the generation of indexes. (The index verborum contains all the words appearing in the transcription and locorum and the index

[1] The different European partners of the BAMBI project are *ACTA S.p.a* (computer society of Florence), *CNR* (Consiglio Nazionale della Ricerche - Istituto di Linguistice Computazionale di Pisa), *BNR* (Biblioteca Nazionale Centrale V.E.II di Roma), *MPI* (Max Planck Institüt für Rechtsgeschichte (Münich)), *CPR* (Consorzio Pisa Ricerche), and *LISI.*

locorum allows to display the positions in which each word occurs in the manuscript.) The indexes can be examined on line and directly in the open document.
Thus the data acquired can be examined in detail in the ambit of the manuscript or evaluated through other manuscripts by means of the sophisticated relational database on which BAMBI relies. The same database also drives the search and cataloguing of manuscripts on the basis of various criteria, saves and classifies the annotations that the user adds to the transcription, supports 'browsing' between pages and between documents, manages rights of access to the application, exports information on manuscripts (stored in the database) in a SGML/HyTime format [CAL 97], etc.

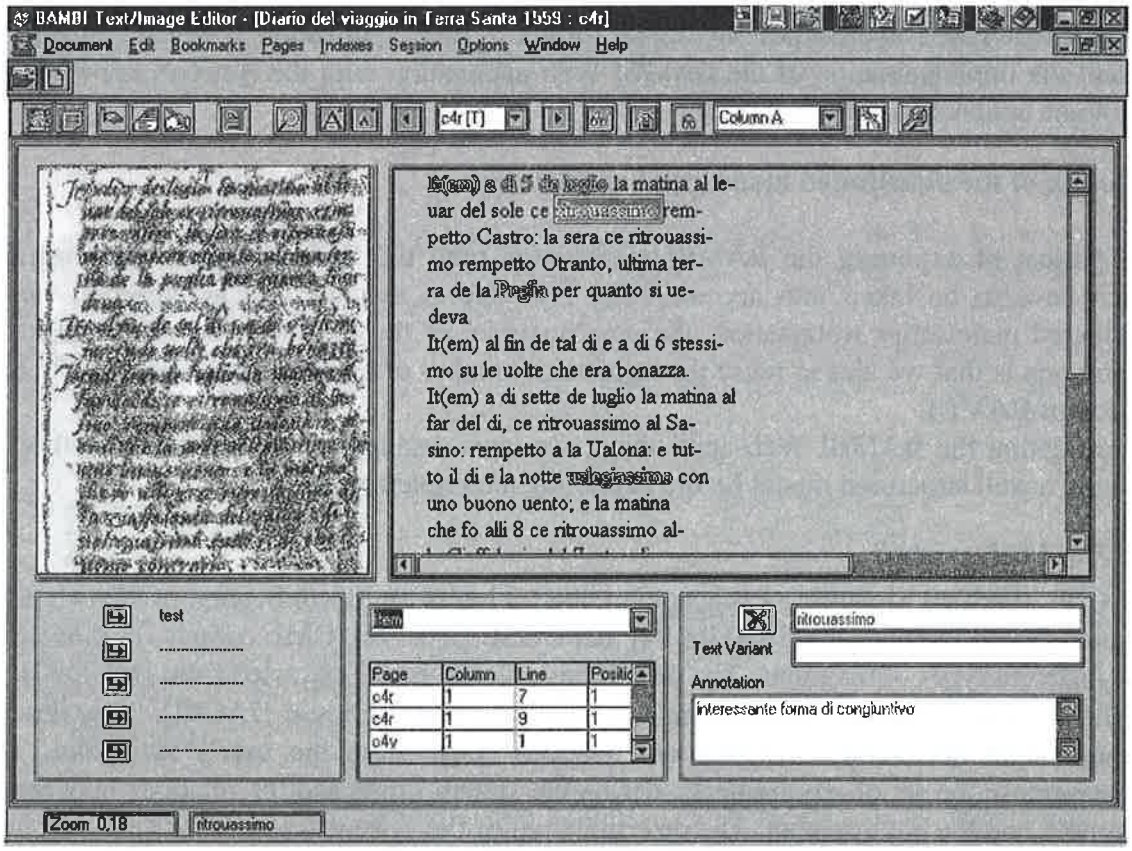

Figure 1: The BAMBI Workstation

It is obvious that a higher feature of this software should enable philologists and librarians to use the new philological workstation in a distributed way.
This distributed multimedia application proposes the same possibilities as the previous version, such as:

- send requests on manuscript contents;
- access and visualize manuscript transcriptions and images;
- perform works on manuscripts :
    - transcriptions ;
    - annotations ;
    - hypermedia navigation ;
    - links between part of image and part of text.

But also new functions tied to the web specificities, such as :

- undertake collaborative work :

    - asynchronous co-operative work (annotations, private news, manuscript exchange,...) ;
    - synchronous co-operative work (information exchange, etc),
- OCR (optical character recognition),
- Electronic restoration of documents,
- Text indexing and transcription functions.

After this introduction on the BAMBI workstation and the aim of the BAMBI web application, we present the design of the distributed BAMBI Workstation with the UML object oriented method. Next, a review of the conventional CGI-based approach, Java-based and ActiveX-based approaches for implementing a Web application is given. Finally, we present the implementation of the BAMBI Web application with the ActiveX approach and give some conclusions on the project.

2. Design of the distributed manuscript workstation

The project of exporting the BAMBI workstation onto the Web includes two constraints which have to be taken into account. The first one is that we aim to preserve, for the distributed manuscript workstation, the user interface of the local version of BAMBI. The second one is that we aim to reuse the maximum number of elements developed in the local version of BAMBI.
For designing the BAMBI Web application, we have decided to use the UML method for deriving a well structured model before beginning the implementation.

2.1. The UML method

The UML (Unified Modelling Language) method is based on the unification of the two object oriented design methods OMT (RUM 94] and OOSE [JAC 92]. UML defines six models and nine different types of diagrams. The definitions given here are extracted from [MUL 97]:

- The use cases (Figure 2) have been formalized by Ivar Jacobson [JAC92]. They describe the system behaviour, in action and reaction form, from the user's viewpoint. They decompose the set of requirements, define the system limits and the relations between the system and the environment. The determination of the system actors is the first step in the use cases. The actor, in the pictogram form, interacts with a family of interactions symbolized by an ellipse. The scenarios are attached to the use cases by defining each task of the actor. In textual and schematic form, they allow us to define the chronology and the origin of the information exchanged between the actors and the system.
- The collaboration diagrams (Figure 3) show the interactions between the objects by pointing out the structures which allow the collaboration of a group of objects.
- The sequence diagrams (Figure 4) introduce a temporal notion into the interactions between objects. The object context is not represented in an explicit manner because we concentrate on the representation of interactions.

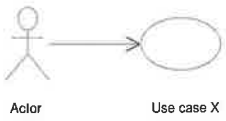

figure 2: Formalism of use Cases

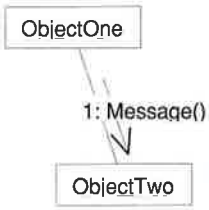

Figure 3: Formalism of collaboration diagram

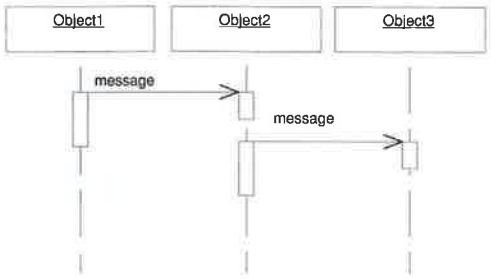

figure 4: Formalism of sequence diagram

3.2. General description of the application

The philological software allows philologists to consult the manuscript images and their transcriptions alongside the Internet network. The user wishes also to access services like the image exchange, the OCR service, etc. The figure 5 gives the architecture of the distributed manuscript workstation.

We have identified four kinds of user:

- The solo user is a user who subscribes to one or several databases of manuscripts. He can consult the manuscripts and, if he wants, he can transfer the manuscript image and the transcription on his own computer for exploiting them locally: he can create or modify transcriptions but he cannot use specialisation tools like segmentation, bookmark or verborum. A solo user is identified by a name, an e-mail address, a login, the date of subscription, etc.
- The research group allows several users to work together on a same manuscript. Each group possesses a responsible person who he is the only one authorized to ask for a publication of created or modified transcriptions. A group is identified by a name, a creation date, commentaries and a responsible person. A user can decide if the manuscript page he works on should be accessible to another user. The group and the responsible person will decide the version to publish. The work in group needs a concerted policy, we have to foresee an exchange area between the users in this group. The communication between two members of the group will be performed by classical mailing. A group user is identified by the same attributes as the solo user except the date of subscription: it is replaced by the date of accession to the group.
- The university group allows a professor to open a login for one or several databases of manuscripts. The users in this group are entirely managed by the professor.
- The public user can access the server which presents the philological application functionalities available on the Web. We consider that this person can only visualize a sample of manuscript images and transcriptions.

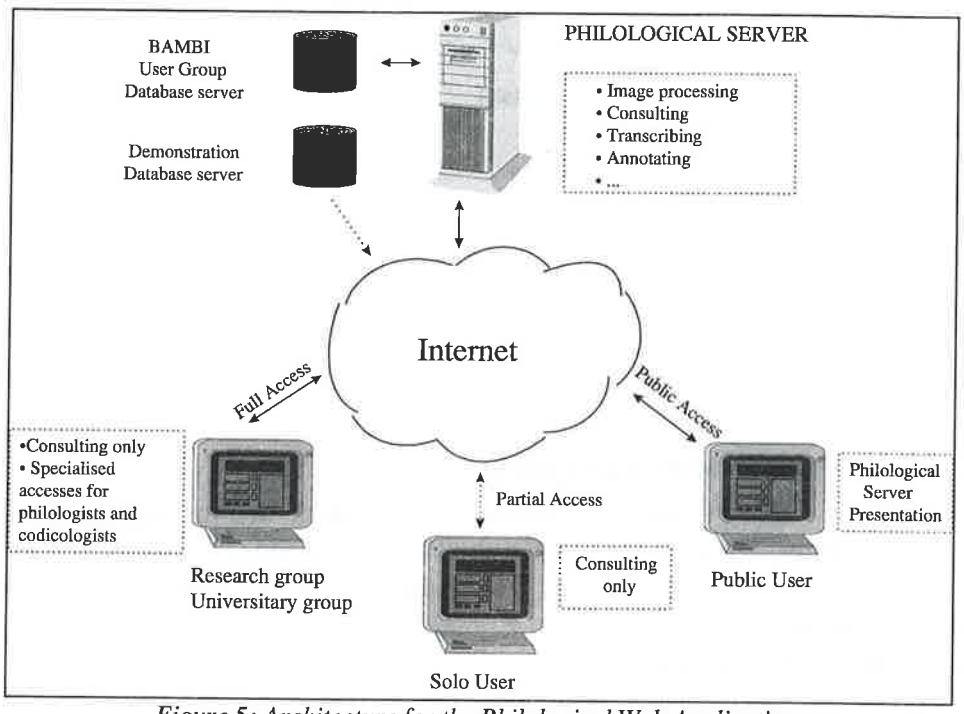

Figure 5: Architecture for the Philological Web Application

3.3. The use cases

The different actors who interact with the system are the following:

- The manager: charged with the creation of the login, the administrator user or the solo user.
- The administrators: they are the responsible persons in the research groups. They assume the management of the users in these groups.
- The philologist: user of the system, his goal is to exploit the manuscripts.
- The public user: other user but who does not work in a perfectly defined domain.

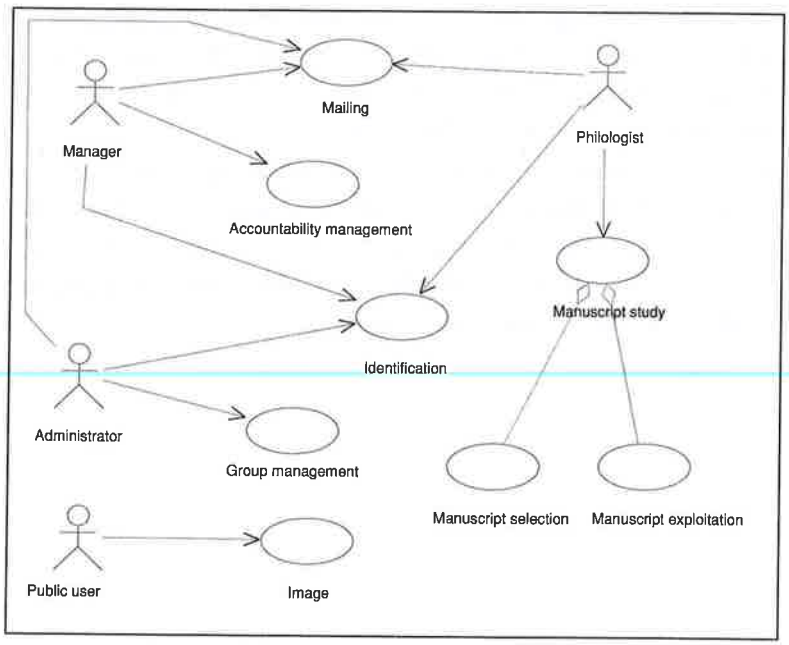

Figure 6: Use cases of the studied domain

These actors allow to define six use cases: identification, group management, accountability management, mailing, manuscript study, image. These use cases will allow to specify what the actor expects from the system. We can note that the case Manuscript study is divided in two use cases: manuscript selection and manuscript exploitation.

3.4. Study of the use case 'Identification'

Scenarios: Identification is realized by all the actors except the public user. The identification consists in verifying that the user has permission to connect but also to determine his profile (manager, administrator,etc.). When the profile is determined, the identification ends (*Figure 7*).

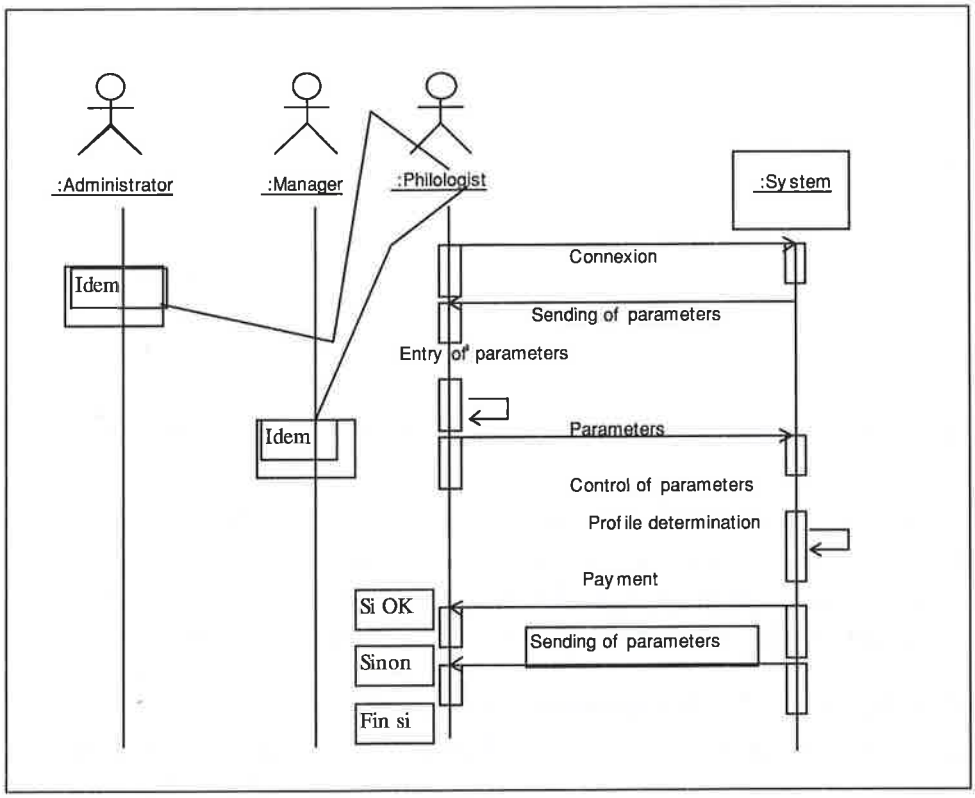

Figure 7 : Scenario Connexion/Identification

Description of collaborations: We use the actor to represent the user interface in a compact manner. The actor can also be replaced by a symbolic class (LimitOfSystem) (*Figure 8*). The user interface is described with classes representing the different windows of the application. The goal is not to define the user interface in an exact manner but to capture the general forms of interactions (*Figure 9*).

Figure 8: Different types of collaboration diagram

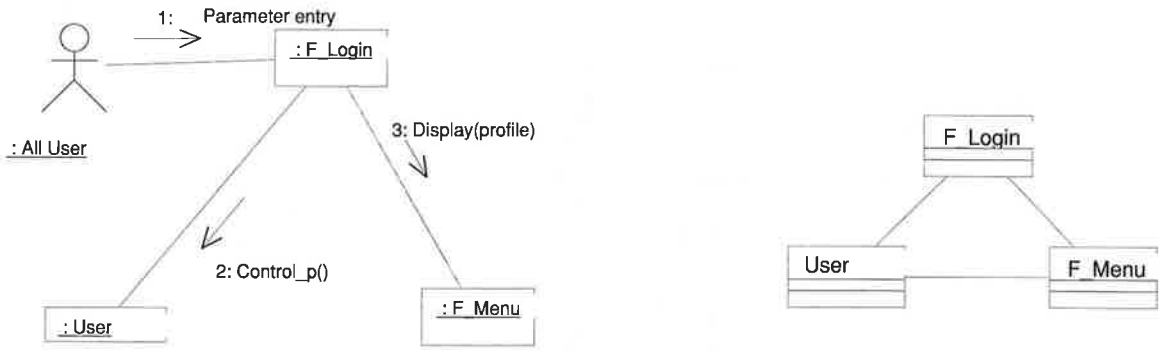

Figure 9: Collaboration diagrams and sketch diagram of classes: Connexion/Identification

A preliminary class diagram can be extracted from the previous collaboration diagrams. The sketches of class diagrams allow us to give a first representation (*Figure 9*). We adopt the conventions defined by [MUL 97]. Each object of the domain is visualized by an interface object with the same name prefixed by F_ : Login is visualized by 'F_Login'.

4. Approaches for BAMBI web implementation

The migration of the BAMBI application to a distributed form on the web should not modify the native functionalities, nor the user interface. So, to implement this new version, we decided to reuse most part of the initial BAMBI software. These constraints led us to evaluate the actual software approaches used to implement Internet applications using database [BER 96] [HAD& 96].

Today, three competitive approaches are proposed:

- the CGI (Common Gateway Interface) based approach
- the Java based approach
- the ActiveX based approach

In the next part, we will describe the main characteristics of each of them and present our choice of the ActiveX solution.

4.1. The CGI based approach

The architecture of a distributed application on the web, using the CGI approach is built on three main parts:

- the web client
- the database server
- the HTTP server with a CGI program.

The CGI (Common Gateway Interface) provides a way to write programs that will run on the server when they are invoked by the client Web browser through the HTML code. The CGI programs are often associated with HTML forms. A database query is initiated by sending a user request (using HTML forms) from the Web client to the HTTP server. Upon receiving the user request, the HTTP server invokes the CGI program to assemble user input data into database-specific SQL statements, and send them to the database server for processing. The query results will be returned by the database server to the CGI program, and then passed to the Web client through the HTTP server. Since CGI is the *de facto* standard for interfacing HTTP server with external applications, this database access scheme is the most commonly used method in today's Web world (*Figure 10*).

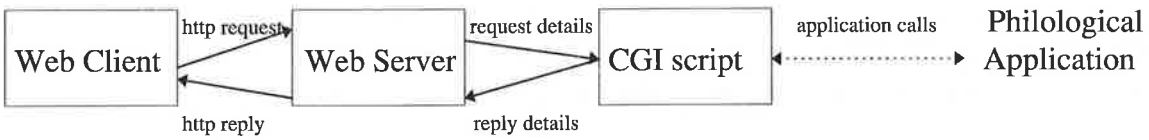

Figure 10 : Application integration via the CGI

Despite of its simplicity and wide acceptance, there are some common problems associated with the CGI-based approach for our philological application:

- the communication between a client and the database application must always go through the HTTP server in the middle, which likely becomes a bottleneck if there is a large number of users accessing the HTTP simultaneously,
- the lack of efficiency and transaction support in a CGI-based access scheme,
- the lack of user access control (the philological application access has to be reduced to specialised groups of users),
- the lack of presentation graphics, due to the limitations of HTML; presentation graphics is very important for manuscript images.

4.2. The java based approach

The Java concept permits usto execute dynamic forms on the client side, using Java programs called applets. The applets are transferred from the server station after reception of the request, and executed on the client station. The Java code is an interpreted code, so it is independent of the client' operating system. This is an advantage, but the prohibitive performance of the applets is poor.

In general, there are two methods of constructing a Java applet [DUA& 96] as a client for accessing a remote database server. One is to use a two-tier architecture (*Figure 11*) in which all client functions are implemented entirely in Java [RIT 95]. The other one is to use a three-tier architecture, in which a standalone Java server is used as a gateway, passing the request and response messages between the applet and the remote database server. The former requires extensive programming effort because a Java client has to be implemented at the protocol level for vendor-specific databases. The latter is easier to implement, because the

gateway server can be created as a standalone Java application with native client library functions wrapped in Java classes. It communicates as a server with the Java applet at one end in user-defined protocols, while accessing the database server at the other end in native client/server protocols.

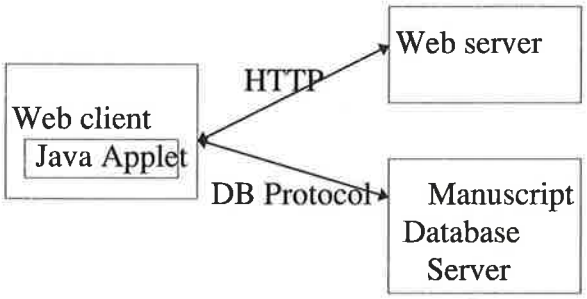

Figure 11 : Two-tier Database architecture using Java

Despite its safety and portability, Java is currently slow and this is a disaster for manuscript image manipulation. Moreover, there are no pre-defined applets and therefore the use of Java requires professional programmers. In some cases, this approach may also restrict the freedom of developers because no other APIs are given for database access.

4.3. The ActiveX based approach

ActiveX is a Microsoft solution, based on the COM model (Component Object Model). The ActiveX concept is the natural evolution of the OLE technology (Object Linking Embedding), adapted to the web needs. [CHA97] [COM96] [FAR96].

The ActiveX technology is built on a concept close to the Java 's one. An ActiveX control, as a Java applet; is a program executed on the client station, which can generate dynamic HTML forms. One major difference is that the ActiveX control is only transferred from the server station to the client application if the client application cannot find it locally. When the ActiveX control is used for the first time, it is uploaded from the server and registred in the client station. So, except for the first execution of the ActiveX control, the performance of client requests are excellent.

As there are many ActiveX controls [Coo& 96] now available for Windows 95/NT, and Visual Basic 5.0 offers a very easily support for creating ActiveX objects, we have chosen this approach for developing the remote philological application.
In addition, the local philological application has been developed in Visual Basic 4.0 and the ActiveX approach associated with Visual Basic 5.0 allows us to reuse the previous code developed. Component software, building an application from reusable parts, is an attractive idea. The fundamental notion is to create an application by plugging software components into some kinds of container. Those components may be specifically written for this application or, better yet, reused from another project. The goal is to create more reliable applications more quickly, and to spend less money doing it. Today, Visual Basic is perhaps the most common example of a container, and the components it uses are commonly loaded when needed from a machine's local disk or a file server.
Moreover, in the context of the BAMBI application, which is a specialized application, the various kinds of ActiveX controls used by the client station are well defined and secured by

the programmers. So, in such a context, the ActiveX approach seems to be a good choice, including performance an security.

5. Implementation

Figure 12 presents the mechanisms used to access the BAMBI database from a HTML page. The elements of the page are not uploaded at the same time, so we can guarantee good performance. Every part will be uploaded when a specific request is sent via the server.

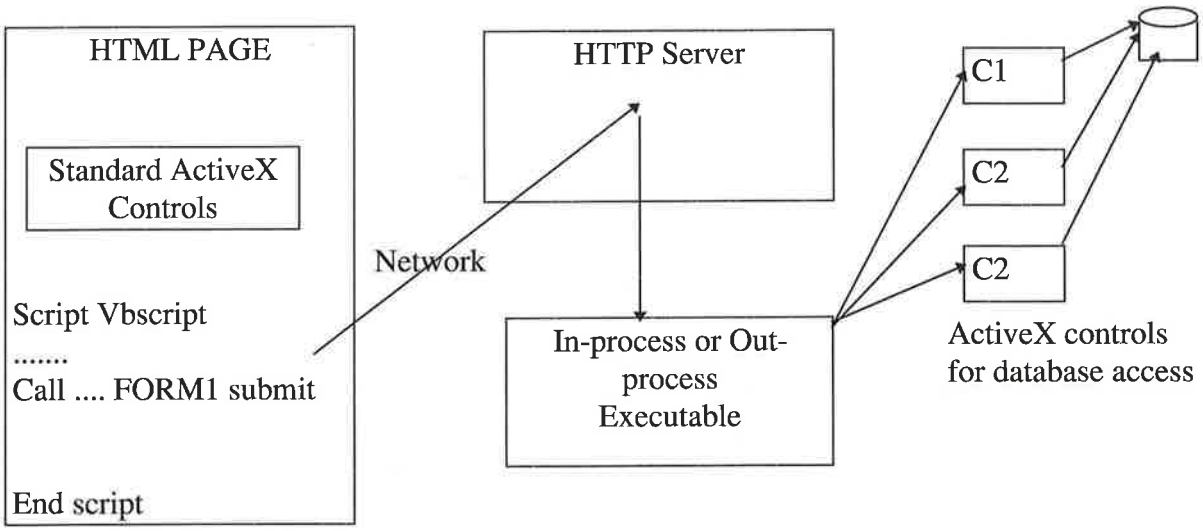

Figure 12 : Mechanisms for the BAMBI database access

In the next section, we will present an example of the implementation of ActiveX controls in the distributed form of the BAMBI application, programmed with Visual Basic. We have chosen the Window entitled "Exploiter un manuscrit", with the controls "Bookmark", "Barre titre", "Image", "Transcription", "Verborum", "Annotation" and an invisible control for the TCP/IP socket. We have extracted a HTML page with the scripts. (Only two functions are presented in this example) [FRA97]

5.1. An example of a HTML page with ActiveX controls

```
<html>
<head><script language="VBScript">
<!--
''''''''''''''''''''''''''global data declaration
Dim Image
Dim Requete
.
.etc.
''''''''''''functions for the management of the events of the controls
Sub itransctl1_EvtMotAnnot()
```

```
Call iannotctl1.search_annot(itransctl1.Glob_Select_Col,itransctl1.Glob_Select_Line,
itransctl1.Glob_Select_Nummot, itransctl1.Glob_Select_Nbmots)
end sub

Sub ibookctl1_EvtBookSelectionne()
l = len(ibookctl1.Glob_id_manu)
  l1 = Pade(l)
 s = "00005evtok00004base" & l1 & ibookctl1.Glob_id_manu
l = len( ibookctl1.Glob_id_page)
 l1 = Pade(l)
 s = s & l1 & ibookctl1.Glob_id_page
CtlWinBambi1.Glob_Buffer_in = s
        call CtlWinBambi1.SVP_Connect()
end sub
.
etc.
   </script>
<meta name="GENERATOR" content="Microsoft FrontPage 3.0">
<title>Exploiter un manuscrit</title>
<meta name="Microsoft Border" content="tl, default"></head>
<body background="Image1.jpg"><!--msnavigation--><table border="0" cellpadding="0"
cellspacing="0" width="100%"><tr><td>
</td></tr><!--msnavigation--></table><!--msnavigation--><table border="0"
cellpadding="0" cellspacing="0" width="100%"><tr><td valign="top" width="1%">
</td><td valign="top" width="24"></td><!--msnavigation--><td valign="top">
<a href="Index.htm" target><img src="TN00054A.gif" width="34" height="42" alt="Home"
border="0"></a><a href="pagetest.htm" target><img src="R0001c1.gif" width="30"
height="30" alt="Up choice" border="0"></a>
<h1 align="center">EXPLOITATION D'UN MANUSCRIT </h1>
<hr width="80%" size="6" noshade>
<p>
''''''''''''''''''''''''''''''''''''''declaration of the ActiveX controls
<OBJECT ID="itransctl" WIDTH=737 HEIGHT=209
CLASSID="CLSID:142C5600-5FFF-11D1-8263-000000000000"
CODEBASE="Bambipj.CAB#version=1,0,0,0">
</OBJECT>
<OBJECT ID="ibookctl" WIDTH=200 HEIGHT=33
CLASSID="CLSID:142C55F7-5FFF-11D1-8263-000000000000"
CODEBASE="Bambipj.CAB#version=1,0,0,0">
</OBJECT>

etc.
''''''''''''''''''''''''''''''''''''''''''''''''''''''end of the declaration of the ActiveX controls
</p>
<p><a href="Index.htm" target><img src="TN00054A.gif" width="34" height="42"
alt="Home" border="0"></a><a href="pagetest.htm" target><img src="R0001c1.gif"
width="30" height="30" alt="Up choice" border="0"></a> </p>
<p><em>Contacter l'administrateur: </em><a href="mailto:webdba@if.insa-
lyon.fr">mailto:webdba@if.insa-lyon.fr</a>
```

<!--msnavigation--></td></tr><!--msnavigation--></table></body>
</html>

The tag **<SCRIPT>** permits to execute the CSS (Client-Side Script) from the browser, more precisely, a CSS can be sent from a HTML page to the browser. The ActiveX objects can generate events managed by the scripts.

5.2. Description of some tags and parameters used in the HTML page

The tag **<OBJECT>** is used for the ActiveX controls. It permits to specify different parameters to initialise the ActiveX control on the WEB page.

The parameter **ID** provides an identification to refer to the <OBJECT> from all parts of the document (**ID**="itransctl" WIDTH=737 HEIGHT=209).
The parameter **CLASSID** permits to identify the objet on the station. The identification of the class is memorised in the base of register of the station (**CLASSID** ="CLSID:142C5603-5FFF-11D1-8263-000000000000").
The parameter **CODEBASE** permits to retrieve the object associated to the control. An ActiveX control uses the services supplied by various files. These files are gathered in an other file named Cabinet file (**CODEBASE** = Bambijp.CAB).

Many other parameters (*NAME*, *DECLARE*, *HEIGHT*, *HSPACE*, etc.) can be defined to specify the ActiveX controls.

We can notice that the Visual Basic 5.0 software permits to develop easily WEB applications. The migration of the initial BAMBI application to the WEB has been simplified by the using of the new functionalities of Visual Basic 5.0 such as the ActiveX controls.

6. Conclusion

The migration of the BAMBI workstation to the web has required us to analyse the functioning of the web and to examine precisely the actual tools able to solve the constraints involved in this evolution.
First, we decided to use the UML method, to formalize the new architecture of the BAMBI application. This step has permitted us to perform a well structured analysis before beginning the implementation.
An evaluation of the existing tools such as CGI, Java and ActiveX, to develop the web BAMBI application, led us to choose the ActiveX approach. Moreover, this solution permits us to reuse the main part of the initial BAMBI program and to achieve easily the evolution of the philological station on the web, without substantial modifications of the existing screens. So, as this new version of BAMBI permits philologists to work alone or concurrently from various sites, it is a good support to work.

References

[BON 97] O. Bonnaterre, A. Bozzi. S. Calabretto. and al. *Better Access to Manuscripts and Browsing of Images : Aims and results of an European Research project in the field of digital Libraries BAMBI Lib-3114*, CLUEB, 1997, 176 pages, ISBN N° 88-8091-569-X.

[BOZ 97] A. Bozzi. S. Calabretto. *Digital Library and Computational Philology : the BAMBI (LIB - 3114) project*, In: Proceedings of the First European Conference on Research and Advanced Technology for Digital Libraries. Pisa, Italie. September 1-3, 1997, Lecture Notes in Computer Science n°1324, Springer Verlag, 1997. p. 269-285

[CAL 97] S. Calabretto, J.M. Pinon. *Modelling of a medieval manuscript database with HyTime*. In: Proceedings of ICCC/IFIP Conference on Electronic Publishing : EP'97. New Models and Opportunities. The University of Kent at Canterbury, Great Britain. April 14-16, 1997.

[CHA 97] D. Chappel. *Au cœur de ActiveX et OLE*. Microsoft press. 315 pages

[COM 96] T. Combs, J. Combs, D. Brewer. *Programmer avec ActiveX*. Editions Sybex

[FAR 96] B. Farrar. *ActiveX* . Simon & Schuster Eds.405 pages

[FRA 97] C. Franklin. *Programmation Internet en Visual Basic*.Thomson Publishing, 355 pages

[JAC 92] I. Jacobson. *Object-Oriented-Software Engineering, A Use Case Driven Approach*, Addison-Wesley. 1992

[MUL 97] P.A Muller. *Modélisation objet avec UML*. Editions Eyrolles. 420 pages

[BER 96] H. Berghel. *The client's side of World Wide Web*. Communications of the ACM. Vol.39, N°1, january 1996, pp.30-40

[COO& 96] T. Coombs, J. Coombs and D. Brewer. *ActiveX source book. Build an ActiveX-Based Web Site*. Ed. John Wiley & Sons, INC. 1996

[DUA& 96] N. N. Duan and B. Atlantic. *Distributed Database Access in a Corporate Environnement Using Java*. Fifth International World Wide Web Conference, May 6-10, 1996, Paris, France

[HAD& 96] S. Hadjiefthymiades and D. Martakos. *A generic Framework for the Deployment of Structured Databases on the WWW*. Fifth International World Wide Web Conference, May 6-10, 1996, Paris, France

[RIT 95] T. Ritchey. *Programming in Java-Beta 2*. Indianapolis, New Riders Publishing, 1995

[RUM 94] J. Rumbaugh. *Modélisation et conception orientées objet*. Edition MASSON. 1994. 515 p.