Ali S. Elbekai
Dr. Nick Rossiter
School of Informatics
Northumbria University
Pandon Building, NE1 8ST
Newcastle upon Tyne
44 191 243 7613
ali.elbekai@unn.ac.uk
nick.rossiter@unn.ac.uk

Vassil T. Vassilev
Dept of Computing, Comm. Technology & Mathematics
London Metropolitan University
v.vassilev@londonmet.ac.uk

Virtual Exhibitions Framework: Utilisation of XML Data Processing for Sharing Museum Content over the Web

Summary
In this paper we will describe a prototype implementation of a framework for organising virtual exhibitions, which uses information provided by the collaborating museums in the form of Web services. The museum content published by the collaborating museums is organised in a homogeneous virtual exhibition space by an exhibition curator and is accessible from a single point of entry – the Virtual Exhibition site. The prototype assumes the content published is an extract of CIDOC-compliant museum database, which allows easy standardisation and further dissemination. The prototype system presented is built entirely using public domain stack of technologies for processing XML data in Java (J2SE, J2EE and additional XML and Web Services packages). It functions as an entirely server-side Web application executed by a Tomcat server connected to a backend database (one for each participating museum plus one for the exhibition itself). Furthermore, the paper will describe one further step in the direction of accomplishment of truly pan-European collaboration for organisation of virtual exhibitions. Using a single XML schema for specification of the common exhibition information and utilising contemporary information technologies for processing XML data over the Web, the approach adopted demonstrates how to add a new European dimension in the inter-museum collaboration and to achieve wider access to the rich European cultural heritage. The system will be demonstrated during the conference.

1 INTRODUCTION

As the Web becomes increasingly the world's official media, many museums, achieves, libraries, and cultural heritage centres throughout the world invest in documenting their collections and in publishing their material via the web, making

their information accessible on the Internet. Generally, museums use many different styles in presenting their collections on their Web pages. In addition, some museums provide sophisticated searching facilities so that the user can retrieve information about required items by subject, date, and place. Others classify their material into groups to satisfy the user's requirement. It is fairly certain that with such sophisticated new technologies, the need for standards to manage the information that these collections contain becomes more and more urgent. As a result, several projects have been conducted in the last few years in order to develop a unified standard for organizing and managing data and information in such institutions. In the same way, many museum organisations are working together to develop standards and techniques and to make their resources more widely and easily available. On this basis, (ICOM-CIDOC) and CIDOC documentation standard working group (DSWG) have engaged in the creation of a general data model for museums. CIDOC have particularly focused on information interchange. As a result, this effort resulted in 1999 in the first complete package of the "CIDOC Conceptual Reference Model" (CRM). Such an achievement was a result of intensive hard word form many experts in the field of IT and museum information many specialists contributed to this work, in order to fully exploit the potential of the CRM as a means to enable information interchange and integration in the museum community and beyond.

Later, the CIDOC group held meeting in London 1999, and they decided to submit CRM proposal to ISO for standardization. Consequently, CIDOC has been authorized to use the ISO facilities that resulted in the acceptance of CIDOC CRM as well as a defined global standard under ISO reference (number ISO/AWI21127) by ISO/TC46/SC4 [ICOM-CIDOC 1996-2003]. In other words, CIDOC will make use of the services of ISO and collaborate with the respective ISO committees to bring the CRM to a suitable final shape and to ensure the widest global agreement on it in the whole international community interested in this field. With regard to the CRM standard, the primary role of the CRM is to serve as a basis for mediation of cultural heritage information and thereby provide the semantic 'glue' needed to transform today's disparate, localised information sources into a coherent and valuable global resource [*ICOM-CIDOC* 1996-2003].

As mentioned before, such massive work has been developed and optimized by CIDOC. The "CIDOC object-oriented Conceptual Reference Model" (CRM), was developed by the ICOM/CIDOC Documentation Standards Group [*ICOM-CIDOC* 1996-2003, *ICOM-CIDOC* 2001, *IGMOI* 1995]. Since September 2000, the CRM has being developed into an ISO standard in a joint effort of the CIDOC CRM SIG and ISO/TC46/SC4 (ISO/AWI 21127). It represents an 'ontology' for cultural heritage information i.e. it describes in a formal language the explicit and implicit concepts and relations relevant to the documentation of cultural heritage.

This paper describes one further step in the direction of the accomplishment of truly pan-European collaboration for organisation of virtual exhibitions. Using a single XML schema for specification of the common exhibition information and utilising contemporary information technologies for processing XML data over the Web, the approach adopted demonstrates how to add a new European dimension to the inter-museum collaboration and to achieve wider access to the rich European cultural heritage. Furthermore the paper describes a prototype

implementation of a framework for organising virtual exhibitions, which uses information provided by the collaborating museums in the form of Web services. The museum content published by the collaborating museums is organised into a homogeneous virtual exhibition space by an exhibition curator and is accessible from a single point of entry – the Virtual Exhibition site. The prototype assumes the content published is an extract of CIDOC-compliant museum database, which allows easy standardisation and further dissemination. The prototype system presented is built entirely using a public domain stack of technologies for processing XML data in Java (J2SE, J2EE) and additional XML and Web Service packages.
The rest of the paper is structured as follows: Section 2 describes the object-oriented conceptual reference model. This leads up to a description of a prototype implementation of a framework for organising virtual exhibitions, which uses information provided by the collaborating museums in the form of Web services in section 3. Section 3.1 presents our use case diagram for the proposed system. Section 3.2 presents the museum association diagram (class diagram). Section 3.3 presents the packages of the system. Section 3.4 presents the ER diagram for the system. Section 3.5 presents the component diagram for the system. Section 3.6 introduces the deployment diagram for the proposed system. Section 4 reviews the related work. Section 5 presents conclusion and further of work.

2 OBJECT-ORIENTED CONCEPTUAL REFERENCE MODEL (OOCRM)

The object-oriented CIDOC Conceptual Reference Model (referred to as “CRM”) is the result of work done by the CIDOC Documentation Standards Group, from 1994-2000, and the CIDOC CRM Special Interest Group from 2000-2002. CIDOC came from an initiative to define the underlying semantics of database schemata and document structures needed in museum documentation for the support of good practice in conceptual modelling, data transformation, data exchange, information integration and mediation of heterogeneous sources.
Basically, the scope of the CRM is the formal knowledge of museums, that is information required solely for the administration and management of cultural heritage institutions. That implies that any other information relating to museum personnel, accounting and visitor statistics, is not largely covered by CRM scope. Besides, CRM is specifically intended to cover contextual information: the historical, geographical and theoretical background in which individual items are placed and which gives them much of their significance and value. Besides, the term cultural heritage collections is intended to cover all types of material collected and displayed by museums and related institutions, as defined by ICOM [C*IDOC CRM* 2004, *Roberts D.A.* 1990]. This includes collections, sites and monuments relating to natural history, ethnography, archaeology, historic monuments, as well as collections of fine and applied arts. The exchange of relevant information with libraries and archives, and the harmonisation of the CRM with their models, falls within the CRM's intended scope.

3 RELATED WORK

[*Vassil Vassilev et al* 99] present both a general description and a technical specification of the information system for museum information processing. [*ICOM-CIDOC* 1996-2003, *CIDOC CRM* 2004, *CIDOC* 1995a] that is being developed into an ISO standard as a joint effort of the CIDOC CRM SIG and ISO/TC46/SC4 (ISO/AWI 21127). CIDOC is the result of work done by the CIDOC Documentation Standards Group, from 1994-2000, and the CIDOC CRM Special Interest Group from 2000-2002, as the result of an initiative to define the underlying semantics of database schemata and document structures needed in museum documentation for the support of good practice in conceptual modelling, data transformation, data exchange, information integration and mediation of heterogeneous. [Vassil Vassilev et al 2000] worked on previous projects for publishing museum content over the Web (in archaeology, in watermark images and in industrial heritage). [Nicholas Crofts 2003] described a practical application of the CIDOC CRM in integrating a large and diverse set of data sources. These data sources, accumulated over a number of years, all contain information relating to Geneva's architectural and cultural heritage.
As a result the previous works do not describe a framework for organising virtual exhibitions over the Web. This paper describes one further step in the direction of an accomplishment of truly pan-European collaboration for organisation of virtual exhibitions. Using a single XML schema for specification of the common exhibition information and utilising contemporary information technologies for processing XML data over the Web, the approach adopted demonstrates how to add a new European dimension in the inter-museum collaboration and to achieve wider access to the rich European cultural heritage.

4 PROTOTYPE DEVELOPMENT OF THE SYSTEM

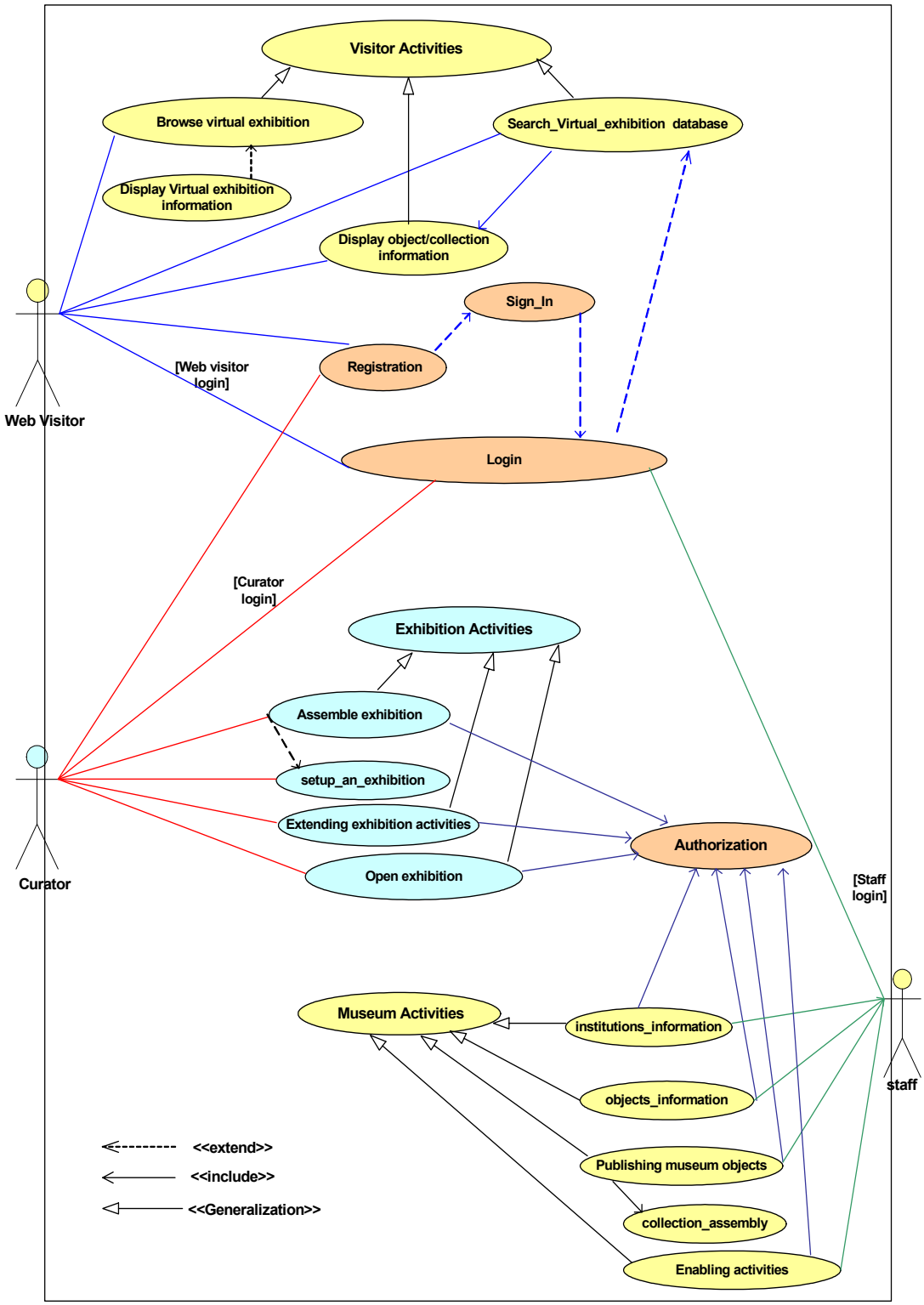
Basically, object orientation is an approach to developing software systems based on data items and the attributes and operations that define them. The concept of the object-oriented approach in system development has been evolved by time during last decades [*Britton Carol* 2000]. At the beginning, interest in the object oriented approach focused on programming language issues; later this concept has grown to cover the whole of the system development process, capturing initial system requirements right through to the final software product. The advantages of an object-oriented approach include many well accepted design goals of quality program development such as modularity, modifiability and maintainability [*ICOM-CIDOC* 2001]. As a result, the benefits to be gained in developing software systems with the object oriented approach is undoubtable, especially with large, complex systems. Throughout the analysis and design stages in this project, the Unified Modelling Language (UML) will be used to presents the concepts and techniques necessary to effectively use system requirements captured using use cases to drive the development of a robust design model. The main aim from using UML [*Bergner Klaus et al* 97] is to be able to produce detailed object models and designs from system requirements and in addition to identify the system use cases and expand them into full behavioural designs. Finally, we expand the analysis into a design ready for implementation.

4.1 Use case diagram

A use case is a sequence of actions that an actor performs within a system to achieve a particular goal. Also the use cases describe the interaction diagrams [*Carlson David* 2001]. These interactions represent the main events that occur in the application domain. Later, during the design phase, these events are translated into the messages that trigger operations. A use case documents the interactions between the roles of the system users called actors and subsets of system functionality. Analysis starts with the search for the actors (categories of users) of this access museum system. An actor represents a generic role played by someone or something interacting with the system. A use case diagram captures a model of several use cases, which depend on one another and how one or more actors interact with those use cases. Figure 1 shows a use case diagram for the museum system and the diagram show the relationships between the actors and the use cases. It is not always easy to determine the boundaries of the system, but by definition, the actors are always outside it. The actors are 'recruited' from the system's users and the people responsible for its configuration and maintenance. They are divided up into the following categories: Web visitor, Curator and Staff.

Furthermore each use case, rendered as an oval in the diagram, accompanied by structured document information such as a goal statement, priority, assumptions and list of activities, describes how the actors fulfill the identified goal. As shown in Figure 1, dependency between use cases may be labelled as <<extend>> or <<include>> or <<generalization>>. *Assemble exhibition* includes Authorization, which means that the first use case (referred to as the base use case) depends on the results or outcome of the included use case. The use case is extends *Assemble exhibition* in this model. The extensions specify behaviour that is optional or exceptional in the description. This use case may be the extension used by another use case.

Figure 1: use case diagram for the museum system



4.2 Object identification and class deriving

Basically, the first step in constructing a class diagram is to identify objects in the problem domain. These may be physical objects, such as people or documents

of organizational entities. All the objects identified at this stage relate to the problem domain. In the next paragraph all the nouns underlined are identified as possible objects.

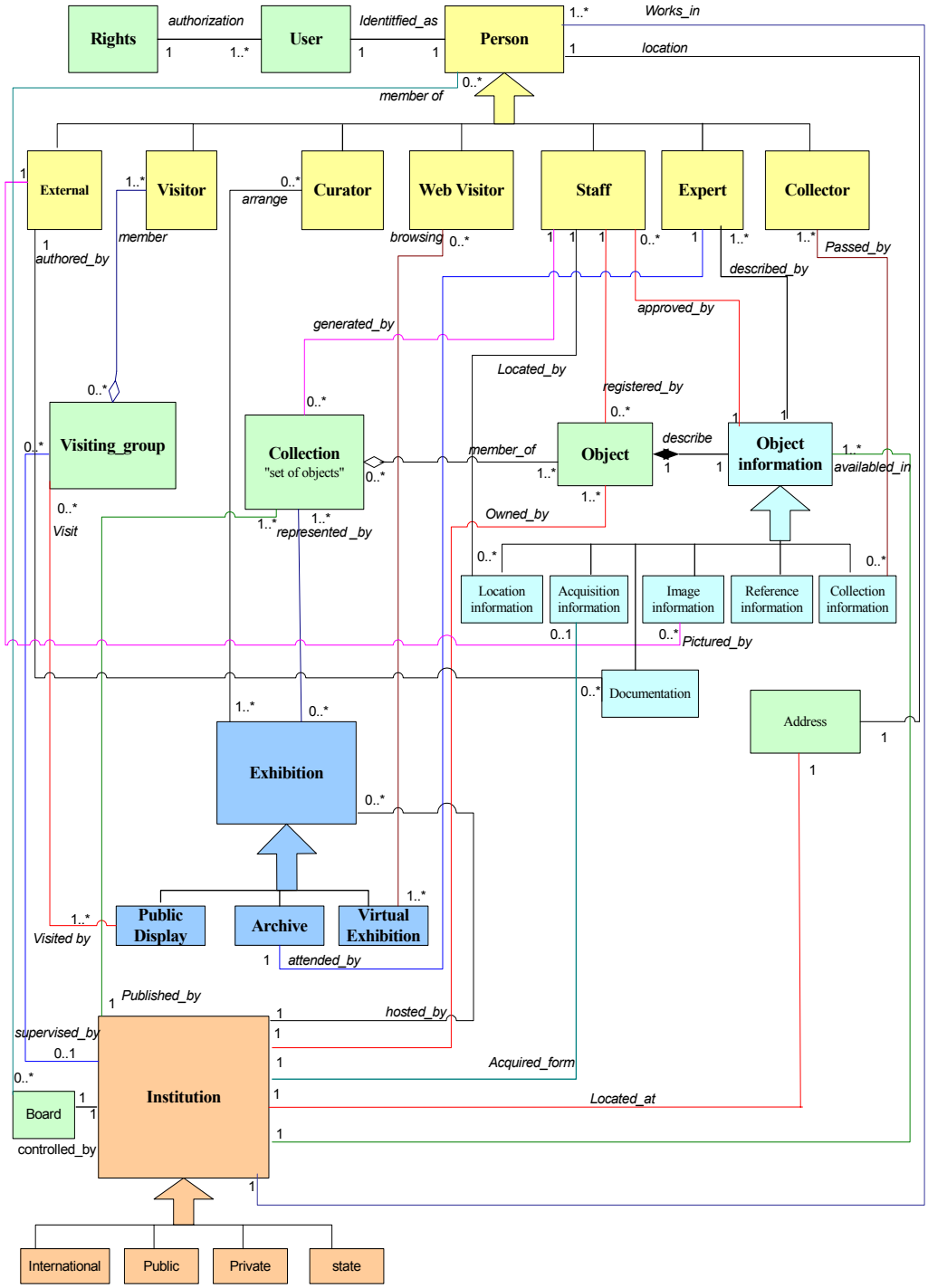
The proposed museum system aims to provide different kind of services to different kinds of clients. These clients are different kinds of users; these users are recognized as persons. In addition every user has limited access to the system (rights). They are categorized as follows:

Internally, staffs are responsible for publishing the museum objects and collections (set of museum objects), in addition to their main task in handling different kinds of information, such as data related to objects, institutions and staff. Next, there is the curator is the person who manages the exhibition. Then, there is the collector the person who collects information related to museum objects (collection information- CIDOC). The information related to museum objects is classified into six categories: image information, documentation, acquisition information, location information, reference information and collection information. Externally, the proposed system has three different kinds of persons. Firstly, the Visiting groups, those who visit the Public display (kind of exhibition). Also, the public display can be visited by individuals (visitors). Secondly, the External person (who is a member of the Board), this board belonging to an institution; the institution could be public, international, private or state institution. In addition, the museum archive will be considered as a type of exhibition in the proposed system. Finally, the proposed system will be visited via the web by different kinds of visitors (web visitors). These visitors will visit the virtual exhibition, which is a kind of exhibition. In the proposed system, the address might be needed for the institutions and persons. The purpose of identifying objects in the problem domain is to derive useful classes. The object classes that can be picked out from the museum system problem brief are shown below. Table 1 gives a summary description of the nouns is identified objects in our proposed system. More details about the class diagram are presented in Figure 2.

Table 1: Object classes

Person	Collection	Exhibition	Institution
Staff	objects	virtual exhibition	public
Curator	object information	public display	state
Collector	image information	archive	private
Web visitor	acquisition information		
international			
Visitor	location information		Address
Visiting group	reference information		
External	collection information		
Board	documentation		
User			
Rights			

Figure 2: Class Diagram

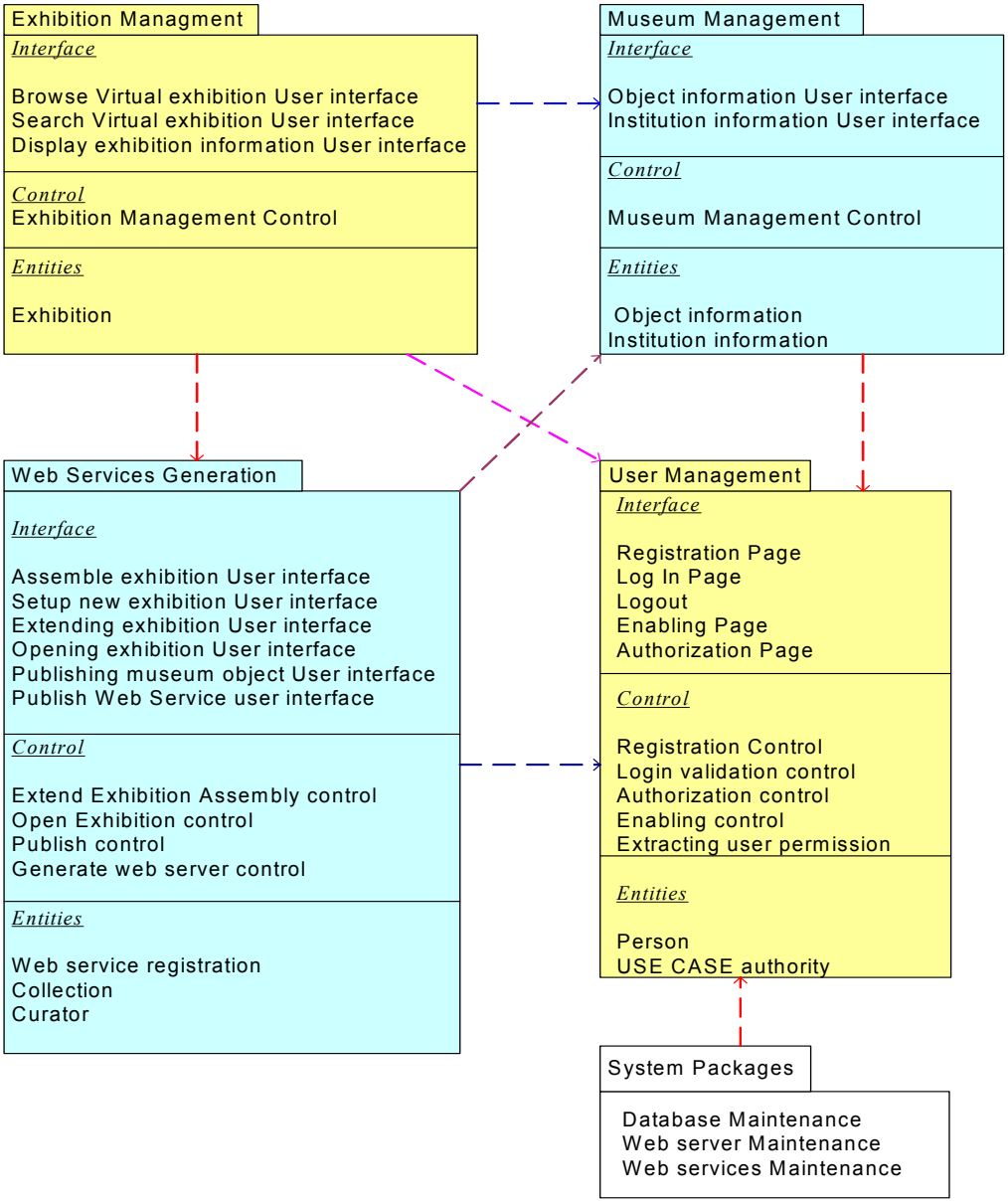


4.3 Packages

Basically, a package is a grouping of pieces of a model. Functionally, the packages are very useful in managing models. The packages we have identified represent different views of the proposed system. These views are organized and described in packages as follows: exhibition management, museum management, web services generation and user management. For each package in Figure 3 a subset of the classes that it contains is listed. The

dependency arrows in the diagram indicate that the Exhibition Management subsystem depends on three packages such as the Museum Management, Web Services Generation and User Management. The Museum Management depends on the User Management package and also the Web Services Generation subsystem depend on the Museum Management and User Management subsystems.
Moreover, the System Package depends on the User Management subsystem. However the User Management subsystem is specified independently of the others. Furthermore as shown in the diagram each package contains interface to user pages, control class and one or more entities.

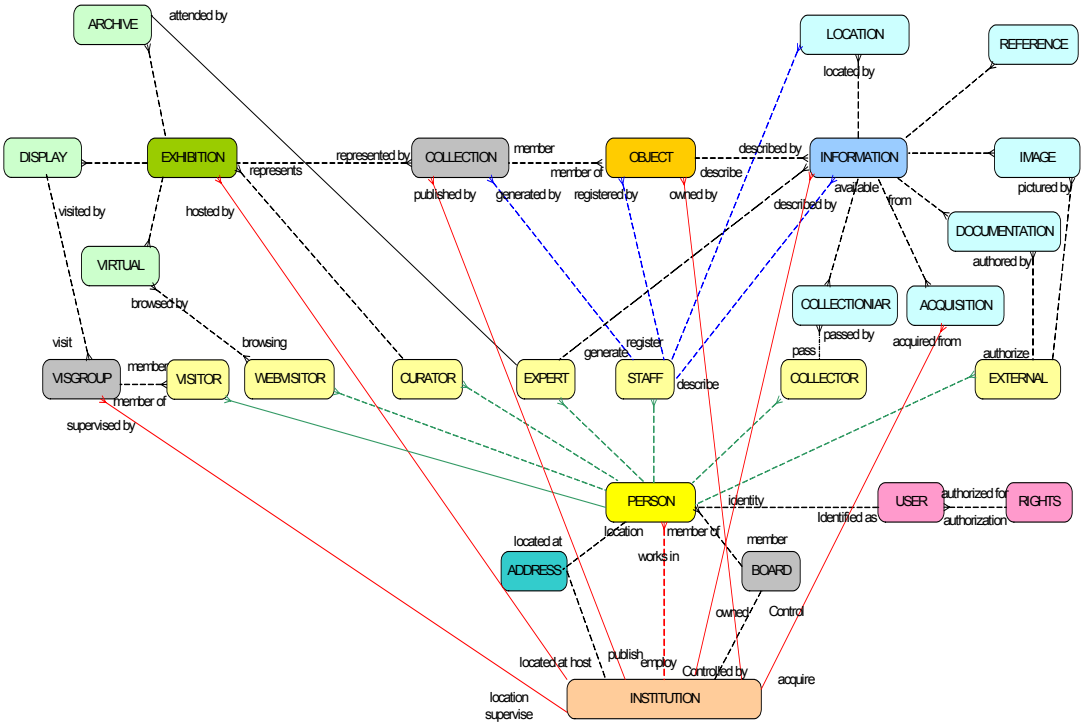
Figure 3: Packages Diagram



4.4 Design Database for the museum system

Basically, the Entity Relationship Diagrams (ERDs) [*Chen P. P.* 1976, *Kossmann F. et al* 99] illustrate the logical structure of databases for our proposed system. Figure 4 shows the cardinality constraints between the objects classes. As can be seen from the diagram each Collection may have many Objects. Also the Object may have much Information. Information has Location, Reference, Image, Documentation, Acquisition, and Collection information. The Exhibition has many Collections, and the Exhibition has kinds such as Archive, Display and Virtual exhibition with one to many cardinality. An Institution has many Visitor groups, Exhibition, Collection, Object, Information and one Address. Note that the Person could be Staff, Expert, Curator, Web visitor, Visitor groups, Collector and External this also one to many cardinality constraints. More details on the entity relationship between the objects classes for our proposed system are shown in Figure 4.

Figure 4: Entity Relationship Diagrams (ERD)



4.5 Design XML schema specification

Basically, the XML schema [*Galloway Trace et al* 2001, W3C 2004, *Holstege Mary et al* 2004] is used to specify the structure and constraints on the XML documents. Moreover the XML schema language can be used to define, describe and catalogue XML vocabularies for classes of XML documents. In our proposed system we design XML Schema that defines the elements, attributes, child elements, whether an element is empty or include text, data type for

elements and attributes and fixed values for elements and attributes in our system. Figure 5 shows a fragment of our XML Schema. Our XMLSchema has exactly one root element as <XMLSchema>. There are many <complexType> elements in our schema, each of which defines an element type in the schema. Any element that contain attributes or child elements is defined by using a complexType. Furthermore the XML Schema specification makes a clear distinction between *definitions* and element declarations. <element> is a declaration of an element that may appear in a valid document instance, but it does not define that element type. It declares that an element named “collection” has type “collection” that is elsewhere defined in the schema as <element name=”collection” type=”collection”>. In fact, it declares the root element for vaild document instances. Each of the <complexType> definitions includes child elements that define the content model and/or attributes for this element type. Both the child element definitions and the attribute definitions specify the type of their content. Each complexType must define the content model for its child elements. The content model for collection, object, objectInformation, exhibition, institution and person is a sequence which has identical properties to sequence in an XMLSchema. The XMLSchema collection definitions may use element type inheritance. As the documention definition is a subtype of objectInformation, and is derived by *extension.* this is specified as follows:

```
<complexType name="documentation">
        <complexContent>
          <extension base="objectInformation">
          </extension>
        </complexContent>
 </complexType>
```

As shown in the above example, <documentation> extends the content model of objectInformation by combining all element and attribute definitions.
Because both documention and its parent type objectInformation are defined with a sequence content model, the new elements defined for documention will be appended to the end of the sequence defined for objectInformation. This capability of XML Schema enables schemas to be written in a much more object-oriented style. In addition, each element maybe optional. This property is implemented in XML by adding a minOccurs=”0” attribute to each element declaration.

Figure 5: Fragment of our XMLSchema for the proposed system

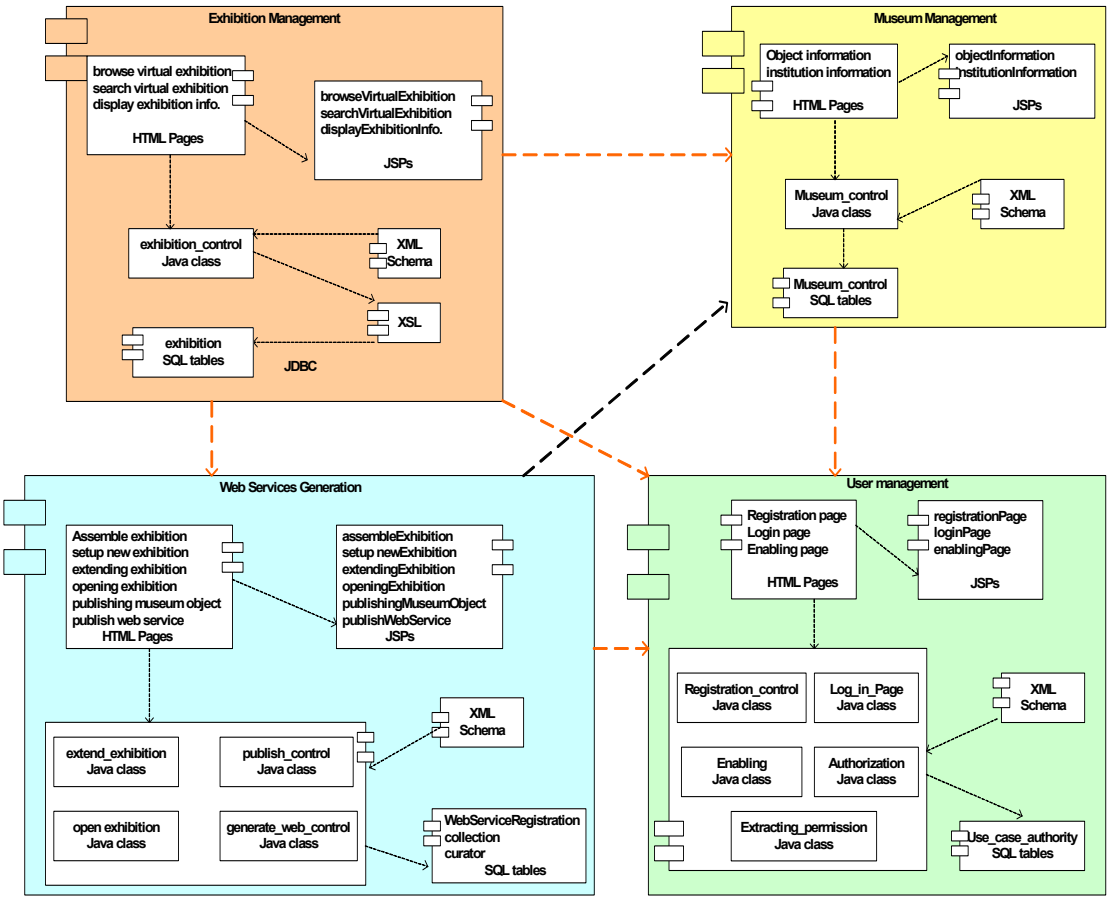
```
<XMLSchema>
 <element name="Collection" type="Collection"/>
 <complexType name="Collection">
   <element name="id" type="String"/>
   <element name="num" type="Integer"/>
   <element name="name" type="String"/>
   <element name="title" type="String"/>
   <element name="publishing_Date" type="Date"/>
   <element name="remote" type="String"/>
   <element name="features" type="String"/>
   <element name="purpose" type="String"/>
   <element name="description" type="String"/>
   <element name="URL" type="String"/>
   <element name="exh_exh_id" type="Integer"/>
   <element name="inst_inst_id" type="Integer"/>
   <element name="staf_staf_id" type="String"/>
   <all>
   <element name="member_of" minOccurs="0">
        <complexType>
          <element ref="Object"/>
        </complexType>
   </element>
   <element name="represented_by" minOccurs="0">
        <complexType>
          <element ref="Exhibition"/>
        </complexType>
   </element>
   <element name="generated_by" minOccurs="0">
        <complexType>
          <element ref="Staff"/>
        </complexType>
   </element>
  </all>
</complexType>
<element name="Object" type="Object"/>
  <complexType name="Object">
      <element name="number" type="String"/>
      <element name="name" type="String"/>
      <element name="title" type="String"/>
      <element name="registration_Date" type="Date"/>
      <element name="description" type="String"/>
      <element name="staf_staf_id" type="Integer"/>
      <element name="cllctr_cllctr_id" type="Integer"/>
      <element name="inst_inst_id" type="Integer"/>
  </complexType>
        <all>
          <complexType>
            <choice minOccurs ="1" maxOccurs="unbounded">
              <element ref="objectInformation"/>
            </choice>
          </complexType>
          <element name="owned_by" minOccurs="1">
            <complexType>
              <element ref="Institution"/>
            </complexType>
          </element>
        </all>
    </complexType>
    <complexType name= "objectInformation">
        <element name="Info_id" type="integer"/>
        <element name="description_date" type="Date"/>
        <element name="description" type="String"/>
        <element name="obj_obj_id" type="integer"/>
        <element name="Inst_Inst_id" type="integer"/>
        <element name="cllctr_cllctr_id" type="integer"/>
        <element name="staf_staf_id" type="integer"/>
        <element name="ext_ext_id" type="integer"/>
        <element name="exp_exp_id" type="integer"/>
        <all>
          <complexType>
            <choice minOccurs ="1" maxOccurs="unbounded">
              <element ref="object"/>
            </choice>
          </complexType>
          <element name="owned_by" minOccurs="1">
            <complexType>
              <element ref="institution"/>
            </complexType>
          </element>
          <element name="approved_by" minOccurs="1">
            <complexType>
              <element ref="staff"/>
            </complexType>
          </element>
        </all>
    </complexType>
</XMLSchema>
```

4.6 Design the component diagram

There are four main packages in the component diagram such as exhibition management, museum management, web services generation and user management. First, there is the Exhibition Management package, which contains packages like HTML pages for the interface, JSPs for processing the HTML pages, exhibition control (Java class), XML schema, XSL stylesheet and exhibition SQL tables [William W. Provost 2002]. Second, there is the Museum Management package, which contains components like HTML pages, JSPs, XMLSchema, Museum control (Java class, SQL tables) as sub packages. Third, there is the Web Services Generation, which contains components like HTML pages, JSPs, XMLSchema, user management control (Java classes, SQL

tables) as sub packages. Finally, there is the User Management package, which contains components like HTML pages, JSPs, XMLSchema, web services generation control (Java classes, SQL tables). Furthermore the diagram shown in Figure 6 shows the usage dependencies as a relation between packages and which package requires another. The dependency is shown as a dashed arrow and the arrowhead points from the component to the one on which it is dependent. Figure 6 shows the component diagram for our proposed work in more details.

Figure 6: Component diagram



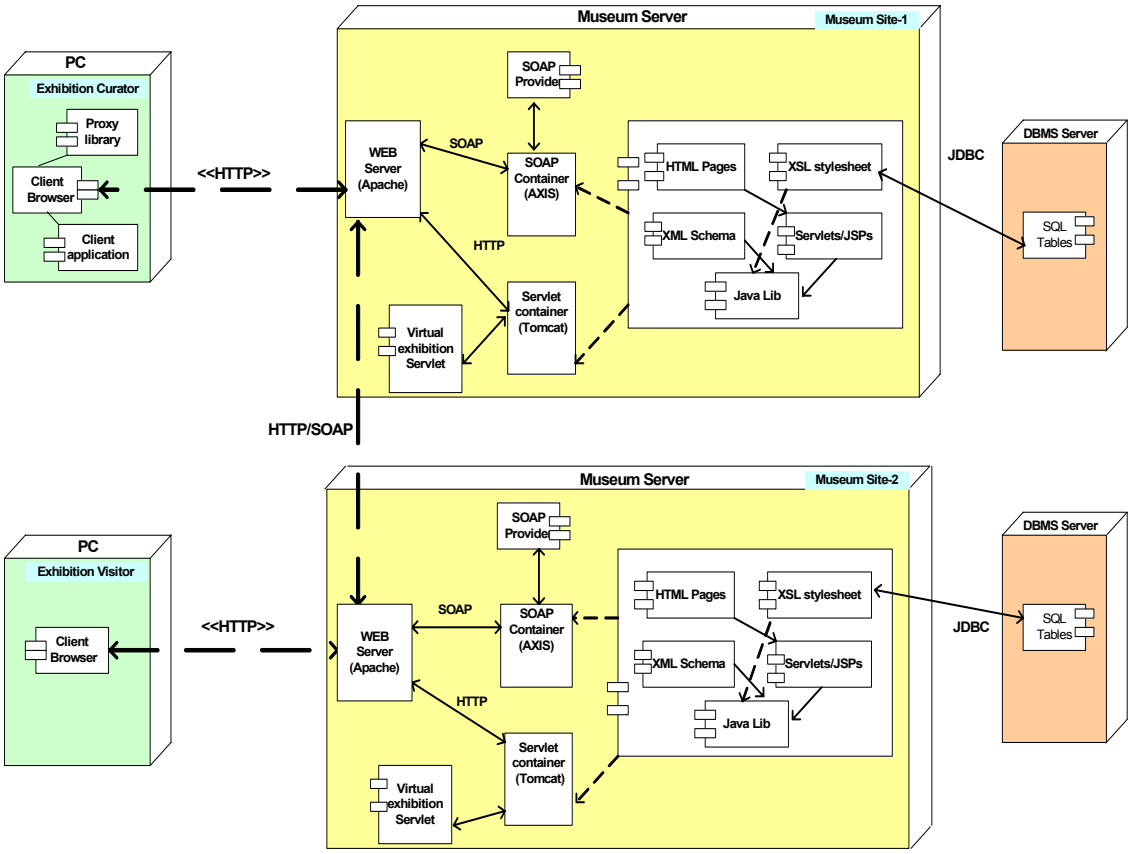
4.7 Design the deployment diagram

The deployment diagram is used to show the configuration of run-time processing elements and the software components and processes that are located in them. Figure 6 shows the deployment diagram for our proposed system. The deployment diagram shows the three nodes such as PC (Exhibition Curator, Exhibition Visitor), Museum Server and DBMS Server that represent the nodes for the client, application server and database server respectively. Furthermore the Museum Server (Application server) represents the node that will process user requests from the Web server and send application responses back to the Web server. The application server node will host the different kinds of the system components such as Servlet container (tomcat), SOAP container,

XSL stylesheets, Java Lib and Web Server. The Web Server node will receive the user requests and send responses from the application to the Client over HTTP protocol. Each of these components deploys some task, for example, the XSL stylesheet is used to translate from the XQuery as a user request to an SQL string, from SQL to XML and from XML into an HTML document that can be read by the web browser. The museum server was implemented using the Tomcat Java application server [Jakarta Project]. Tomcat is the reference implementation for the Java servlet technology [Sun Microsystems]. Moreover, the Database Server node that will host the database server is used by the components in the application server node to store and retrieve the data used by the System. To be more specific the deployment diagram for the museum system shown in Figure 6 is composed of three-tiers:

- Web browser (Client) that can connect to the museum server, i.e., to access the Java servlets. The client can use PCs to run a simple web browser. The client communication protocol XML uses HTTP.
- Museum server, a set of servers and internal network connecting them. This provides a web server capable of accessing data from DBMS and making it available to the client. Technology choices for the middle-tier include a Web server, Web sever with servlets (Tomcat), SOAP container (AXIS), a Virtual exhibition servlet, Java Server Pages, HTML Pages and XSL stylesheets [W3C XSL 99]. The communication protocol between the database and the middle tier could be JDBC.
- DBMS server with SQL Tables provides database storage [Bourret 2004].

Figure 7: Deployment diagram



5 CONCLUSION

We have presented a prototype of a framework for organising virtual exhibitions, which uses information provided by the collaborating museums in the form of Web services. The museum content published by the collaborating museums is organised in a homogeneous virtual exhibition space by an exhibition curator and is accessible from a single point of entry - the Virtual Exhibition site. The prototype assumes the content published is an extract of a CIDOC-compliant museum database, which allows easy standardisation and further dissemination. The prototype system presented is built entirely using public domain stack of technologies for processing XML data in Java (J2SE, J2EE and additional XML and Web Services packages). It functions as an entirely server-side Web application executed by Tomcat server connected to a backend database (one for each participating museum plus one for the exhibition itself). Furthermore, this paper describes one further step in the direction of accomplishment of truly pan-European collaboration for organisation of virtual exhibitions using a single XML Schema for specification of the common exhibition information and utilising contemporary information technologies for processing XML data over the Web. The approach adopted demonstrates how to add a new European dimension to

the inter-museum collaboration and to achieve wider access to the rich European cultural heritage.

6 REFERENCES

Bergner Klaus, Andreas Rausch, Marc Sihling. Using UML for Modelling a Distributed Java Application, Institute for Informatics. (http://www4.informatik.tu-muenchende). July 1997.

Bourret Ronald. XML and Database. Published online at URL http://www.rpbourret.com/xml/XMLAndDatabases.htm, last updated July 2004.

Britton Carol. Object-Oriented Systems Development: a gentle introduction. University of Hertfordshire and *Jill Doake* Anglia polytechnic University. 2000.

Carlson David. Modelling XML Applications With UML Practical e-Business Applications. Foreword *by Jeffrey Hammond,* Rational. Software Corporation. 2001.

Chen P. P. The entity-relationship model: Toward a unified view of data. ACM Transactions on Database Systems. **1**(1):9-39. 1976.

CIDOC CRM. Mappings to the CIDOC CRM can be accessed from a list of Technical Papers on the CIDOC CRM SIG Web site, at http://cidoc.ics.forth.gr/technical_papers.html, accessed April 2004.

Crofts, Nicholas, Combining data sources – prototype applications developed for Geneva's department of historical sites and monuments based on the CIDOC CRM. 2003.

Galloway Trace, Altova, Inc. Principles of XML Schema Design. USA, May 2001. Inc Beverly, XML conference & Exposition, December 8-13, 2002. Baltimore convention centre. Baltimore, MD. USA. World Wird Web Consortium. "XMLSchema". At http://www.w3.org/xml/schema, as accessed 24 June 2004.

Holstege Mary and Asir. Vedamuthu. XMLSchema: Component Designators. World Wide Web Consortium, Working Draft WD-xmlschema-ref-20040716, July 2004.

ICOM-CIDOC. Introduction to the International Committee for Documentation of the International Council of Museums (ICOM-CIDOC). 1996-2003. http://www.willpowerinfo.myby.co.uk/cidoc/cidoc0.htm#English

ICOM-CIDOC. CRM Special Interest Group, Working Group of CIDOC, 2001. http://cidoc.ics.forth.gr/who_we_are.html.

IGMOI: International Guidelines for Museum Object Information: The CIDOC Information Categories, produced by the (CIDOC) of the (ICOM). Oct 1995. http://www.cidoc.icom.org/guide/guide.html.

Jakarta Project. Tomcat version 5.5. Reference Implementation for the Java servlet 2.2 and Java Server Pages 1.1. http://jakarta.apache.org/tomcat/.

Kossmann F. and D. Kossmann. A Performance Evaluation of Alternative Mapping Schemes for Storing XML Data in a Relational Database. In Rapport de Recherché No. 3684, INRIA, Rocquencourt, France, March 1999.

Provost W., An XML Validation Architecture Using XML Schema, XPath, and XSLT. February 2002.

Roberts D.A.. Terminology for museums. Proceedings of an International Conference, held in Cambridge, England, 1988 (1990). Edited by D.A. Roberts. Cambridge: Museum Documentation Association.

Sun Microsystems. *The Java Servlet specification*. Documentation. 2004. http://java.sun.com/products/servlet

Vassilev V., B. Gaydarska. Reducing the complexity of CIDOC object-oriented model implementation through ontological minimisation. CAA2000, University of Durham, UK (2000).

Vassilev V., I. Rangelova, G. Simeonova et al. Drill-down Navigation inside Archaeological Museum Database. Dynamic Classification and Controlled Terminology Implementation. Proc. EAA2000, Portugal Institute of Archaeology, Lisboa, Portugal (2000).

Vassilev V., I. Stoev, I. Rangelova et al. Museum Information Systems: CIDOC data model implementation in the ArchTerra project. Bol. CILEA No. 69 (1999).

W3C. Extensible Markup Language (XML), at http://www.w3c.org/XML/, accessed 13 April 2004.

W3C XSL Working Group, W3C Recommendation on XSL Transformations (XSLT) version 1.0 (http://www.w3.org/TR/xslt). November 16, 1999.