DIGITAL LIBRARY MIDDLEWARE WEB SERVICES:
MODULE TO ACCESS A Z39.50 BASED VIRTUAL UNION CATALOGUE

Joaquim A. Martins, Hélder T. Zagalo, Joaquim S. Pinto
Instituto de Engenharia Electrónica e Telemática de Aveiro
Departamento de Electrónica e Telecomunicações
University of Aveiro
3810-193 Aveiro, Portugal

Abstract: - We are building a set of middleware Web Services to support a digital library, offering improved scalability, reliability and interoperability. This paper describes one module of this overall middleware related with the conception and implementation of a virtual union catalogue, which aims to allow interoperability with Z39.50 capable systems and to be an added value tool to do bibliographic search in several distributed bibliographic systems. It is also described a middleware component, based on Web Services, which will allow the access to the virtual union catalogue through a programmable interface and to evaluate and test the possibility to use XML protocols to increase the interoperability among bibliographic systems.

Key-Words: - Web Services, Z39.50, bibliographic systems, digital library, middleware

1 Introduction
Ubiquitous access to resources is nowadays a reality due to the Internet. Such things like to consult your bank account, to pay your electric and telephone bills, to buy any kind of products in your country or abroad, in fact in any place in the world, without leaving the comfort of your home were, not so long ago, science-fiction things. In fact, this and much more is possible just by having a PC connected to world wide computer network, the Internet.

Ubiquitous access to the resources is also a concern of modern governments, which are increasingly committed to implement information systems to allow a more transparent, simple and efficient way for the citizens to access the services of the huge, bureaucratic and labyrinthine public administrations. In this context the Portuguese Republic Assembly of Deputies launched, two year ago, a project to build a Digital Library of the Assembly Records [1] so that the citizens can have access to what happens in the Parliament and, the deputies, historians and investigators can have a fast and flexible working tool. The work described in this paper is part of this framework.

Traditional libraries can’t escape the Internet and today lots of them around the world use the Web to offer on-line access to their bibliographic catalogues in order to make consults, reservations, etc, offering precious services for students, professors, investigators and citizens. In spite of these new services offered by libraries the bibliographic search is still an ungrateful task. To find the relevant resources, adaptation to different information systems, that use different search languages and semantic, communication protocols and user interfaces, are some of the difficulties that users often face dealing with these systems [2].

This paper describes the conception and implementation of a virtual union catalogue, which aims to respond to some of the enumerated difficulties and to be an added value tool to do bibliographic search in several distributed bibliographic systems. It is also described a middleware component, based on Web Services, which will allow the access to the virtual union catalogue through a programmable interface and to evaluate and test the possibility to use XML protocols to increase the interoperability among bibliographic systems.

This is a subsystem of a larger middleware module, the distributed archive, index and search system (figure 1) that will support the above mentioned digital library of the Portuguese Assembly of Republic. This will allow increased scalability, improved performance and reliability and the integration of different sources and types of information in an overall integrated system able to cross-reference all the archives. For the moment is built the server for the parliament records and are under construction the servers for the historic, photographic and video archives. In this paper will be described in more detail the module to deal with...
bibliographic system through the Z39.50 protocol and that forms a virtual union catalogue.

2 Virtual Union Catalogues
Since the 70ths many institutions, with large distributed bibliographic repositories, have a constant concern in unifying its several origins of resources or, in other words, creating collective catalogues. Initially, the collective catalogues, with emphasis for the MELVYL public system (University of California) and the commercial systems OCLC, RLG and WLN, were implemented as centralized systems with a single data base [3]. In this way, the collective catalogue is built by first gathering, processing and storing of the records coming from the local systems. As a consequence of this implementation there is the need for a periodic actualization (weekly, monthly, etc.).

Later, after the Z39.50 [4] protocol has been implemented in the bibliographic systems to allow remote access to them, several institutions began studies and work in order to change their centralized collective catalogues by virtual ones. The virtual union catalogue isn’t stored in only one place but created in real time through the simultaneous and distributed searches to the various local catalogues, using exactly the Z39.50 protocol. Some examples of institutions which have been making efforts in this way are the California Digital Library (CDL) [5] and the Canada National Library [6]. This type of implementation avoids the duplications of bibliographic references and the periodic actualizations.

There are several research initiatives in the field of virtual union catalogues, like the vCuc – virtual Canadian union catalogue Project [7], the VEL – Virtual Electronic Library Project [8] or the CDL – California Digital Library initiative [5], that have identified a set of issues related with the implementation of this type of catalogue which raises several challenges:

- **Concurrency of sessions** – the Z39.50 protocol, in spite of being session oriented and allowing, in the 1995 version, concurrent search, only defines interactions between two machines (called “Source” and “Destination” in the standard). This leads to necessity of developing client applications able to support multiple and concurrent sessions with several servers;
- **Duplicate records** – when a search is done simultaneously in several systems the user may receive many duplicate records. Several libraries can have the same title and thus the user receives more than one record about the same item;
- **Semantic consistency of searches** –

![Figure 1 - General Architecture](image-url)
servers and destination data bases support a large variety of Z39.50 attributes and their combinations by which it is possible to do the search. Often, the several implementations associate different semantics to the same attributes. To make searches to multiple servers, they must conform with the least common denominator in order to have results with coherence;

- **Interconnection with other applications** – usually the search in a distributed collective catalogue is done in the context of other operations: referencing, cataloguing, researching, etc. In this way it is needed that the Z39.50 client be able to transfer the gathered information, in an efficient way, to other applications.

Actually there are some working virtual collective catalogues accessible by the public, like the Deutshe Bibliothek [9], the Virtual Karlsruhe Catalogue [10] and the Virtual Canadian Union Catalogue [11]. However their main functionality is centered in the distributed simultaneous search, lacking other features of a virtual union catalogue.

In our work we tried to deal with all the above enumerated questions and to show possible solutions for them.

### 3 The developed Virtual Union Catalogue

Based on recent work in the field of virtual union catalogues, we designed and developed a virtual union catalogue, which is able to create and to handle several parallel Z39.50 sessions and presents possible solutions for the above raised questions. One of our system objectives is to improve the quality of the distributed searches, taking into account the semantic consistency of searches and the received data usability, in order to identify and remove duplicate records.

Our system follows the original idea of a virtual union catalogue, but presents a significant add-on compared with the existing virtual union catalogues. These, at least the ones that can be accessed by the public, show their results in separate catalogues, one for each searched library. This forces their users to have some additional hard work to select the appropriate data. In opposition to this, our system shows the search results in a unique catalogue, letting the user “think” that he is working with only one library. All the records received by our system, coming from several sources, are first processed and merged before being presented to the user. In addition, to each record is appended information about its sources, being possible to know the libraries where a specific work (book, journal, etc.) is located.

#### 3.1 Architecture

In the design of the virtual union catalogue architecture several requirements where weighted, which where considered mandatory to implement an efficient and flexible system. Firstly, the system should be able to support several simultaneous users and to maintain multiple sessions with Z39.50 servers. Secondly, the system should allow its reorganization in order to be used in different contexts and situations, thus being component oriented.

According with this, as shown in figure 1, the system has basically two main modules: a Web server and a module for distributed searches (DSS – Distributed Search System). The web server does the interface with users, through their web browsers. This module receives the search requests and presents the final data to the user. In the actual context of computing, where the web browsers are more and more being adopted as the preferred user interface the choice if the web server module was obvious. In the other hand, at least for now, to use a web server to allow the massive access to information systems is the best choice.

The DSS module is the responsible to consult and to receive information from Z39.50 servers, as well as to process and to merge the received records. Its architecture is shown in figure 2 and is component oriented. This means that this module is composed of several independent components, which can be used individually to do specific tasks or may be reorganized to do, as a new module, another mission.

The main components of the DSS module are:

- The Z39.50 client;
- The Record normalization;
- The Record and duplicate processor;
- The front-end;

#### 3.1.1 Z39.50 Client

The Z39.50 client is the main component of this module. It is the responsible to do the searches in the Z39.50 servers and to receive the results coming from them.
Figure 3 shows the component architecture, where one can see the objects that make it and the flux of data and signalization among them. The objects are:

- **ZSession** – responsible for the implementation of Z39.50 protocol and thus for the creation and management of the session with a server;
- **ZClient** – responsible for all the component management, namely the reception of search requests, the control of active sessions, signalization with external entities, and the presentation of the record sets;
- **ZStore** – this allows the storage and regrouping of the several sessions records.

To have a clear idea of the relation among these objects and of its functioning let's follow what happens when a search request is made to the component.

The search request arrives to the component through the ZClient object. For each server to which it is necessary to make a search, the ZClient object verifies if there is an active session, which corresponds to an active ZSession object. If it exists, the search request is immediately forwarded to it, if not, the ZSession object is created and then the request is passed to it.

Then, each ZSession object sends the request to the server to which it is connected and waits for the response. The response starts by the number of records found for that search and then the ZSession object requests the exact quantity of records it wants. As the records arrive to the ZSession object, it sends them to the ZStore object.
which regroups them with the other records received by other ZSession objects.

Meanwhile, the ZClient object checks the ZStore object to verify if the number of received records until that moment fulfils the request done. When that happens or when all ZSession objects signalize that they haven’t more records for that request, the ZClient object signalizes the external entity that made the request that the response is ready. Now that entity must collect the set of records which are the response of the several servers to its search request.

As one can see, this component executes its operations asynchronously. Some entity may ask the component to do a search request and go on doing other tasks. When the component ends the search execution it signalizes asynchronously the entity, which can then get the collected records. This is possible because all the Z39.50 client component objects have their own execution thread, allowing all the operations inside the component to be done in parallel.

This component may be used in an independent way or integrated into other systems serving, for example, as a search engine for the Z39.50 protocol. It can be integrated in bibliographic cataloguing systems or proprietary applications for bibliographic citations.

3.1.2 Record Normalizing
The responses from Z39.50 servers to the requested searches come usually in a proprietary format. Those responses arrive as records, like a data base record, but with a complex format peculiar to the bibliographic systems. Most of the bibliographic catalogues uses the standard format MARC, which is the conjunction of the standard ISO 2709 with a set of metadata elements, defined in the several MARC nuances, like the USMARC, now known as MARC21 [12], the UNIMARC [13], the UKMARC [14], etc. These nuances, in spite of having the same base format (the ISO 2709), appear as different sets of metadata and even with numbering plans different from their tags. This diversity of nuances results from the particular needs of each country to have its own terms to describe the bibliography. However, when one must join and process in the same system records coming from several different places with incompatible formats, that puts obvious problems in the analyses, extraction and presentation of information carried by them. From this results an obvious need to normalize the records to a common denominator.

There are yet other reasons to do the conversion. It is our aim with this system that the records which arrive to the user are simple enough to be user friendly and, at the same time that they have enough information to be used also by professionals. To achieve this, the MARC format is very complex.

Additionally the records must be reused in the client side, to pass them to other applications or simply to be stored locally which puts the problems of coding and decoding the ISO 2709 standard and the copyright of records purchased to cataloguing services. Due to this problem, precisely, there are actually many OPACs - Online Public Access Catalogs that present their search results in other formats than MARC. To overcome this it was decided to convert all received records to Dublin Core (DC) [15] and XML [16]. This decision is conformant with the Bath Profile [17], one of the more actual Z39.50 profiles. This profile is one ISO IRP – Internationally Recognized Profile and groups the experience of several former profiles like the ATS-1, CENL or DanZIG. The transformation is based in a conversion document between MARC21 and Dublin Core [18] which is in the LoC – Library of Congress.

3.1.3 Processing of records and duplicates
This component has the mission to identify and to remove duplicate records, keeping only one record with the identification of all the sources, storing and managing the received records.

The experience of several research groups in the field tell us that, at a simplified level, the duplicate detection may be done by the ISSN or ISBN identifiers [12], but more advanced detection levels need the execution of more sophisticated algorithms like the ones found in OCLC and AMICUS systems [6]. The problem with these ones is its higher costs and response times. In our component it is used the Dublin Core Identifier field to identify the duplicates. If this field is empty in the record, the identification is tried through the remaining fields.

3.1.4 The Front-end
The Front-end component is the module responsible for the coordination activities and for the execution of search requests that cross all other components. It has also the responsibility to do the interface with the exterior, making the integration of all other components and presenting the module to the external systems also as a component itself.
3.1.5 Implementation
The DSS module was implemented in JAVA [19]. The native capacities of this programming language in order to develop concurrent and network applications, as well as its portability and suitability to develop components were determinant to choose it. All the module components were implemented as JavaBeans [20].

The Z39.50 protocol implementation was done based on a commercial product called ZedJAVA [21]. It consists of a Java classes library which implements the protocol data structures and the message coding and decoding. This implements the protocol static part. The dynamic part was developed and implemented by our components.

To handle the XML structures was chosen the Java API for XML Processing version 1.1 (JAXP 1.1) [22] from Sun Microsystems. This API supports the two interfaces for XML manipulation: the Document Object Model level 2 (DOM 2) [23] and the Simple API for XML version 2.0 (SAX 2.0) [24].

As Web server was chosen the IIS – Internet Information Services from Microsoft because, as the development platform is the Windows operating system, is the one with the best integration capabilities. Besides that, as the search module was developed in a multi-platform environment (Java), it was also a test to the capacity of integration of the two systems. The DSS module is also a JavaBean component and, through the JavaBean – ActiveX bridge from Sun, it is seen as an ActiveX component by the web server. From this point on its manipulation can easily be done with ASP scripts in the server [25].

4 Access Middleware Based on Web Services
The Z39.50 protocol (ANSI/NISO Z39.50-1995 / ISO 23950) is at the application level and defines interactions between two machines: one client and one server. The protocol allows a client application to send search requests to the server, which searches its data bases looking for records that math a certain criteria and return all or some of them. This protocol has some versatility, allowing it to offer other services besides bibliographic applications.

However, in spite of all that the Z39.50 protocol has to offer, it’s often heard among the bibliographic communities the following comments: “It’s to complex to implement …”, “It doesn’t work …”, “It isn’t needed now, since we have the Web …”, etc.

These critics are due, largely, to the several protocol implementations than to the protocol itself. In fact, the protocol is complex, flexible and powerful, which leads to different functionalities interpretation by the different implementations. Beyond that, most implementations don’t implement the full protocol, using only some subsets accordingly with their interests. After this, it’s easy to understand all the critics to the protocol and the low level of interoperability between the different systems. Because of this showed up some profiles, like the Bath Profile, and others, trying to give new opportunities to the protocol.

One can conclude that simplicity and interoperability are key concepts that can promote the use of a protocol (as can be seen with the HTTP example). Based on this assumption and in the developed work of the virtual union catalogue, we think that the key to the bibliographic information systems resides in the adoption of simpler access protocols and record formats. This was the motivation to do the conception and development of a middleware component, using Web Services [26], which allows a simpler access to the virtual union catalogue by other applications and systems.

4.1 The Web Services
The main objective of any middleware platform is to reduce the complexity of system integration and thus be able to create some artificial homogeneity among a set of heterogeneous systems. Some examples of such platforms are: CORBA - Common Object Request Broker [27], J2EE – Java 2 Enterprise Edition [28] or COM – Component Object Model from Microsoft [29].

On the other way, the Web Services can be classified as middleware for middleware [30], allowing also to accommodate the heterogeneity and diversity of other middleware systems. They are seen by several authors as a great promise for the next middleware generation.

The Web Services platform uses the Internet infrastructure ubiquity to apply middleware solutions with given maturity, with the advantage of using existing Web protocols and open XML based standards, in the main three areas:

Communication Protocols – from which stand up the SOAP – Simple Object Access Protocol [31], an W3C specification;
Service Description – where we find another standard from W3C, the WSDL – Web Services Description Language [32];

Service Discovery – for which there is the protocol UDDI – Universal Description, Discovery and Integration directory [33].

By this way, regarding the Web Services as a means to approach different systems and applications, we believe that it’s adoption by the bibliographic community would bring a significant increase in the interoperability among their systems, allowing to enhance the existing services, like the access to bibliographic records, information about the existing works and the work sharing among libraries.

4.2 Architecture and Implementation

Any process change always implies a difficult initial phase with some inertia, mainly when it has as consequence the radical cut with the strong established practices. This is also the case in the library field; where since long years have been made huge investments in cataloguing and searching systems.

For this reason we present here an intermediate solution which doesn’t imply any cut with legacy systems or any change in the thinking process by the people that run and manages the bibliographic systems. The solution, which is outside the libraries, depends on them to offer the service, but on the other side, the libraries don’t need to know this service.

This is a service that when aggregated to the virtual union catalogue offers the possibility to consult several bibliographic systems, simultaneously or not, through a programmable interface in a form of Web Service.

In figure 4 it is shown in a brief way the service architecture, which we called Bibliographic Access Web Service.

This service appears in this architecture as an embedded component in the Web server. Really the service access is done through the web server, which detects that one wants to access the specific bibliographic search service via Web service and directs the request to the specific component that implements it. In order to proceed with the searches, the component uses the DSS module from the virtual union catalogue.

The tools used to implement the Web Service where the Visual Studio .NET and the component was done as an ASP.NET [34] using the C# language. The access from the Web Service to the DSS module was pacific, since the DSS appears to the system as an ActiveX.

5 Conclusions

Virtual union catalogues continue to be a research field of high concern in the library milieu. There are still many open questions, to which the researchers continue to look for solutions. On the other hand the people with less knowledge in the technology field start to understand that the potential around the concept of the virtual
collective catalogue can bring great benefits which can lead to a new vision of the traditional library space.

The use of new middleware platforms, like the described one based on Web Services, can contribute to a high degree of approximation of the library computing systems infrastructures. This improved interoperability will result in a greater universe of facilities, beyond the ones we are used today. In this way it is possible the integration of several services, at several levels, as being a virtual library which represents several physical libraries.

Our current and planned future work is to enhance this Web Services middleware platform in order to have a distributed Digital Library and Archive and not only a Virtual union catalogue, maintaining the identity of each individual system but being able to support cross-referencing among them all and to support several types of media (text, image, video, etc.). A prototype is being made in the aim of a contract with the Portuguese Republic Assembly.

References:
[9] Die Deutsche Bibliothek (http://z3950gw.dbf.ddb.de)