A Contribution of Open Source Technologies to Support Distributed Digital Library’s Repository and Index Services

HELDER T. ZAGALO, JOAQUIM A. MARTINS, PEDRO M. ALMEIDA, JOAQUIM S. PINTO
Instituto de Engenharia Electrónica e Telemática de Aveiro
Departamento de Electrónica e Telecomunicações
University of Aveiro
3810-193 Aveiro
PORTUGAL

Abstract: – This article describes the contribution of some open source technologies to develop and evaluate the implementation of some digital library’s services. These services were developed as web services, forming a middleware framework that has as its main goal the scalability, reliability and interoperability improvement of all the system. Specifically, the repository and index services were implemented on top of Native XML Databases. To measure the impact on the framework of using web services to access these databases, it was conducted a study that establishes comparisons between access times using and not using these services.


1 Introduction

In the context of a major political imperative in Europe, the Portuguese Parliament started, three years ago, a project to build a Digital Library of the Assembly Records [1, 2]. This project was planned to go through several phases: earliest phases would deal mainly with text and digitized documents, latest ones would deal with image, sound and video documents too.

Through all these phases, the architecture for the digital library has been subjected to many refinements, and it can even be stated that its conceptualization has much evolved. So, the present conception for this architecture incorporates the majority of earliest ideas, reformulates some ones, and involve all in a much more embracing context, allowing the system to grow and to contemplate new scenarios.

In what sense is open source [3] related with all this work? Until some months ago: none. The idea to use open source technologies in this work appeared when some new conceptual ideas were introduced into the digital library’s architecture. To understand that it’s a good idea to recall some of the main advantages of these technologies:
• easy availability;
• free redistribution;
• guarantied evolution, due to their open nature.

In consequence, even if they aren’t the final choice for the production phase, they are mostly good enough for testing and evaluation of new ideas.

Presently this was really what was needed: some good and fully functional technologies, which could be used to materialize and evaluate some concepts.

2 The Overall Digital Library’s Architecture

The overall architecture of the digital library, presented in this article, follows some of the concepts about open-architectures for digital libraries, exposed already by other researchers [4, 5]. This means that the global functionality of the system is partitioned into several well-defined services, each one having a well-defined public interface, which defines the allowed requests and the possible responses and exceptions. Services are, in the real world, instantiated by server components. These components are seen like black boxes, at least from the architecture point of view, since they implement correctly the publicized interfaces.

In this way, the functionality of the digital library is in fact the result of the union of all functionalities offered by the aggregation of the individual services, which can be characterized as a whole with much more value than the sum of its parts.

In Fig.1, it is presented the overall architecture of the digital library. Even if it isn’t very explicitly exposed, the system’s architecture can be segmented in three main layers:
• the repository layer;
the index and query layer;
- the front-end layer.

The repository layer is constituted by multiple and distributed repositories of information, which can, in the real world, be as complex as the more sophisticated databases present in the market, or as simple as file systems. This layer provides functionalities to store and access information documents, which are multimedia documents (text, images, sound, video, etc.), as well as index information.

The index and query layer is composed by several specific engines (for workflow, audiovisual, cataloging information, etc.) that support discovery and indexing operations over all the information stored in the repository layer. A query submitted to these engines, return result sets, containing meta-data information about information documents corresponding to the profile requested. The extension of the meta-data returned depends from the specification of the initial query. It can be all meta-data or only the document id that allows the document retrieve.

Last, but not least, the front-end layer is the “one” in this system architecture. On one hand, it is responsible by all interactions with the clients of the system, on the other hand it implements the DAISS – Distributed Archive Index and Search System, which is the component responsible to distribute queries or index requests and retrieve, merge and serve the collected answers. The clients of the system can be either end users, “sitting” on their browsers, or computational agents, communicating with the system as a simple service. This implies two different interfaces, one based on a web server to serve the user’s browsers, and another based on a web service, which is a computational interface.

Talking about web services [6], as it is presented in Fig. 1, the elected interface between all components, inside the system, is in fact the web service. Web Services are a technology based on open standards, like XML, WSDL and SOAP, which allow, in this particular case, the components to present them selves as services and locate those selves anywhere in the network. One of the main advantages of this technology is precisely to be immune to firewalls.

Observing, one more time, Fig. 1, it’s present a service, which isn’t located in any layer, but has a
high level of access from all the other services. This service is the Services Directory. It is also based on an open standard, called UDDI [7], and allows the registration of all services, as well as their functionality. All services communicating with each other, inside the framework of the digital library, must access the Services Directory firstly, in order to know which services are available, what their functionality, and finally where they are located.

These open technologies were chosen, due the requirements imposed on the implementation of the framework. The services should be distributed, in an intranet or even in the internet, and the framework should be flexible enough to plug-in more and new services, as the system grows and accommodates more functionality. So, the services should be implemented in loosely coupled components. In fact, the services aren’t aware at all of the existence of one another, exception made by the information requested to the Services Directory, and to be participating in a more large system. Another issue, as the architecture lets assume, is the fact that the individual components don’t have anywhere a so called centralized administrator. They are the most autonomous possible, and cooperate with each other in a peer to peer paradigm (P2P) [8].

3 Implementation with Open Source Technologies

As seen in the architecture section, one clue technology used in this framework is Web Services. This technology is now in wide spread use over two major platforms: .NET [9] and J2EE [10]. In terms of functionality, the choice for one platform is a simple matter of taste. Both platforms offer very good tools do create, deploy, describe, publish, invoke and execute web services.

3.1 Implementing Web Services

In spite of all offerings from the two industry giants, open source is still a good choice to consider, if considerations like those made in the introduction section are bear in mind. So, open source technologies were chosen to implement web services, as well as some repository components.

The tools chosen to implement these services belong, in their majority, to the Apache Software Foundation [11], one of the most important organizations in the open source’s world. For web services, it was taken AXIS v1.0, which is a SOAP engine, fully compliant with version 1.1 of the protocol specification, and has also some support for version 1.2. To deploy web services created with this engine, it was used Tomcat v4.1.18, which is an Apache tool to host Java Servlets and Java Server Pages. This tool is necessary because AXIS doesn’t have real support to connect directly to the network, exception made for an included simple HTTP server that can only be used for testing. So, AXIS runs inside Tomcat, as a Servlet, and profits from some of its functionalities like authentication and security.

3.2 Implementing Repositories

For the repositories implementation, it was required support for XML storage and indexing operations. Although, many current relational databases already provide support to XML, it was decided to adopt Native XML Databases, due their higher functionality, dealing with this kind of documents. One more time, by the reasons already enumerated, the choice was to pick an open source product. The elected product was Xindice v1.0, also from Apache.

Xindice is a XML database that allows storing and retrieving XML documents in their native format. Its query language to search documents organized in collections is XPath. It gives also the possibility to create indexes, based on specific elements from the XML documents, to increase search performance.

To access Xindice, it was chosen the XML:DB API [12], seeing that it is also possible to access it from other APIs and from the command line. XML:DB is a non-profit industry initiative to promote the development of standard specifications for XML databases and data manipulation technologies. In fact, XML:DB wants to create a sort of standard technology to provide access to XML Databases, in an independent way, almost as ODBC or JDBC for traditional databases. Currently, several commercial and open source XML databases have already incorporated XML:DB interfaces. That’s why, the Xindice database used in this moment to implement repositories in the digital library’s framework, can be replaced at any moment by other product. eXist is another open source XML Database, using XML:DB, which is in consideration for repositories implementation.

3.3 Development of Repository Services

To provide repositories as services in the framework of the digital library, it was developed a web service, that uses XML:DB to access database systems and provides a minimal interface to
manipulate documents and collections. Each repository instance has its own web service’s instance, so N repositories are seen in the framework as N services.

In Fig. 2 it is presented the public interface offered by the web service, called “xmldbws”. AXIS has facilities that allow a simple browser to list all deployed web services, and their methods, in a given address. It is also possible to request immediately the WSDL file, which describes entirely the web service and can be used to generate appropriate stubs to access the service.

Analyzing “xmldbws” methods, it is reported methods to create, remove, list and count collections, as well as documents. About documents, it has one more method to retrieve a document from the repository, given its id. For querying purposes, it has two methods:
- queryCollection – to query a given collection, using the XPath language, and specifying the maximum number of results to return;
- getLastQueryMoreResults – to get more results from the result set generated by the last query occurred.

The second method is of great importance, mainly if queries must be made over the internet. Depending on the repository’s size and the query, a single search operation can generate tens, hundreds, thousands or many more results. So it’s obvious the existence of these two methods to alleviate the network load and to speedup search operations.

4 The Web Services Overhead

One very important issue about web services, and their utilization, is their overhead in load and time. Time is perhaps one of the greatest constraints, developing web applications today. That’s why it was conducted a study trying to estimate the time impact of using web services between the repositories and their clients.

4.1 Methodology

The methodology used in this study implied firstly the creation of 50,000 XML documents, more or less equal in size and structure’s complexity. Then they were used to make three simple operations in the repository: inserting, getting and removing. It was measured the average and maximum time, taken by these operations, firstly through XML:DB API alone and then through the “xmldbws” web service.

All operations were made in the same machine, to maintain the same conditions to both scenarios, discarding network delays, which would significantly influence the results.
4.2 Results

The numeric results can be seen in Fig. 3. About the difference between using and not using web services, the average of this difference is about 113.3 ms (milliseconds) for the average times of all three operations. The difference average for the maximum times is about 6750.7 ms.

What conclusions can be taken from these results? Making merely quantitative comparisons, web services take much more time to accomplish the operations. However, seeing this surplus in qualitative terms, it can maybe to be despised. This overhead rounds the hundreds of milliseconds. The impact over the delay an end user can support is very little or even none. From the end user point of view, all delays under one second can be underestimated. The problem rises when the delay ascends to several seconds, which is the case of the maximum values. But, as the averages values are very low, comparatively with maximum ones, it can be concluded that the number of times the maximum values are attained is very small.

So, at least, at the present moment, the overhead of using web services can be discarded. Maybe in the future, with all the system working, this reality can become altered.

5 Conclusions

The architecture presented in this paper allows the implementation and instantiation of multiple digital libraries, in a very dynamic way, allowing the creation and offering of more and new services. Theoretically, this kind of architecture makes possible the unlimited extensibility of the implemented frameworks. One curious and tricky feature about this architecture is its recursive nature. From the front-end layer to the repository layer, services request the execution of multiple and parallel operations on services belonging to the underneath layer. This concept can be very attractive to develop systems with a very high degree of components distribution.

The implementation and development of some of the digital library’s services, using open source technologies, demonstrated the feasibility of the solution. If, in the future, the degree of performance and/or confidence on the system gets down, these open source components can be easily replaced by commercial ones.

6 Bibliography


(http://www.opensource.org)


(http://www.w3.org/2002/ws)

[7] UDDI – Universal Description, Discovery and Integration of Web Services
(http://www.uddi.org)


(http://www.apache.org/)

[12] XML:DB Initiative for XML Databases
(http://www.xmldb.org/)