DigitArq: Creating a Historical Digital Archive

Miguel Ferreira[1], José Carlos Ramalho[2]

[1] Information Systems Department, University of Minho,
Campus Azurém 4800 Guimarães – Portugal
mferreira@dsi.uminho.pt

2 Informatics Department, University of Minho,
Campus Gualtar 4710 Braga – Portugal
jcr@di.uminho.pt

Abstract

In this paper we present the steps followed in a project called DigitArq that aimed at building a centralised repository for archival finding aids. At the Arquivo Distrital do Porto, finding aids existed in several different formats and media. Migration was used to convert all the finding aids into a single normalised format based on an international standard – the EAD/XML. After migration, archival management software was developed to maintain the collected information and assist archivists in the creation of new finding aids. Archival finding aids are described by hierarchical structures which can easily be described in XML but present interesting issues while using Relational Databases. The relational data model employed is described in detail together with the reasons that made us choose this kind of implementation. Some statistics of the migration process will also be discussed in order to give the audience some insight about the legacy problem and the necessary investment to deal with it.

Keywords: Hierarchic models, Relational DB, EAD/XML, Metadata, Migration, Digitalisation, Optical Character Recognition.

1 INTRODUCTION

One of the most important tasks performed at an Archive is the description of items in custody. Archival description differs from bibliographic description in a number of ways. While the latter is centred on the single publication, archival description centres on aggregations of documents that share the same provenance. These descriptions are also called finding aids and are basically metadata about those items. Finding aids are available to consumers allowing them to perceive what items are kept by the Archive and used by archivists to locate specific items within thousands of others.

Over time, novel and richer finding aids are adopted by Archives, while at the same time, indexes, lists, inventories, catalogues and transference guides are produced to better serve the needs of its customers. Although these finding aids help users and archivists attain the artefacts they seek, they comprise a heterogeneous universe extremely hard to manage. The lack of coherence between most finding aids makes updating information a nightmare. To put an end to this scenario, it was carried out at the Arquivo Distrital do Porto[1] (ADP) a project called DigitArq whose major goal was to homogenise the entire collection of finding aids and serve as a first attempt at the edification of a Digital Archive.

This project comprised several stages. The first one consisted in the conversion of a series of paper-based documents and digital databases containing different types of finding aids to a

[1] OPorto's district archive and record centre.

single digital format based on an international standard – the General International Standard Archival Description [1] – *Digitalisation and Migration.*

The second stage of the project aimed at constructing a centralised database to store all the collected finding aids and developing special purpose software to manage all that information – *Archival Management Software*.

A third stage consisted in the development of a search engine that allowed Internet users to find and browse the collections – *Search Engine*.

Preservation is another great responsibility of an Archive. Different techniques can be used to address this issue such as microfilming and digitalisation. The ADP already provided a digitalisation service to their customers, which allowed them to buy digital copies of books and documents. This service was provided in an *ad-hoc* basis without a well-defined plan or workflow. Document images were kept on CD-ROMs, which where difficult to maintain and preserve. This setting was becoming unbearable given that a lot of images where not saved for future use due to the inherent difficulties in their retrieval.

Since finding aids were now going to be published online it made sense to develop software to allow archivists to publish document images as well. This software should be able to simplify the digitalisation process (using digitalisation profiles) and guarantee the long-term preservation of the digital objects and the media involved. Such a system opens new opportunities for business, as images can be sold online to customers all over the world – *Digital Objects Management Software*.

Section 2 of this paper describes the type of information that is produced daily at an Archive and provides a swift overview of some project decisions concerning mainly the use of the Encoded Archival Description (EAD) standard as a mean to describe archival finding aids. Section 3 contains a detailed description of the digitalisation process that was applied to paper finding aids that existed at ADP. Section 4 reports some of the techniques used to migrate several types of digital objects to the EAD standard. Section 5 outlines the most important features of an Archival Management Software developed during the course of this project. Section 6 briefly describes a Search Engine that allows local and internet users to find specific items by searching through the catalogue (finding aids database). We continue on section 7 by introducing a Digital Objects Management Software developed for with the purpose of managing and publishing digital content. The following section (section 8) describes a middleware-based strategy for storing hierarchically-structured data using relational databases. We conclude on section 9 by discussing some proposals for future work.

2 FINDING AIDS AND STANDARDS

The information produced at an Archive (i.e. finding aids, a.k.a. archival descriptions) is organised hierarchically, providing a top-down, general to specific description of a collection of items. A book, for instance, is not described alone. A description of the organisation, person or family that produced the book is included (represented at the root level by *fonds* in Figure 1), as well as the description of all the physical and logical sections that characterize that producer.

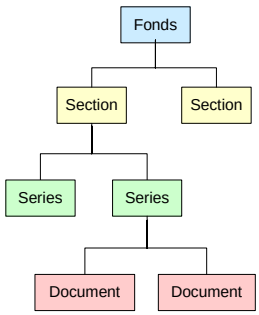

Figure 1: Hierarchical organisation of finding aids.

One might think of archival finding aids as trees of description nodes whose root element describes the producer and the leaf nodes provide descriptions of documents that cannot be further divided. Between the root and leaf nodes, information that describes the physical structure of the producer and the logical groupings of documents within the organisation is provided.

At ADP, finding aids were scattered throughout different folders and databases, usually in different formats and sometimes in distinct physical locations – the centralisation of information was desperately needed.

Before centralisation, all finding aids should be converted to the same standard format. The *General International Standard Archival Description* [1] and its most notable XML representative – the *Encoded Archival Description* [2] – were chosen to accommodate all of the different types of finding aids.

2.1 Encoded Archival Description (EAD)

The Encoded Archival Description (EAD) was developed in the 1990s as way of encoding traditional paper finding aids in machine-readable form. Because there was no universally followed standard for creating finding aids, the originators of EAD gathered sample finding aids from a number of repositories and tried to accommodate the range of practice among them [3]. This resulted in a very flexible design where most elements are optional and where concepts can be described in a range of different ways.

The following sections quickly summarise some of the project decisions related to the use of EAD.

2.1.1 Date ranges

One of the first problems detected on the EAD standard was its weak capacity to describe date ranges. Typically, a document's description comprises two dates – the initial and the final dates of creation. The EAD standard provides a single element to describe dates and it doesn't define any format for those dates. A possible workaround for this would be defining our own syntax to clearly distinguish one date from the other, e.g. `<unitdate> 1436/1441 <unitdate>`.

In our project we created an alternative solution that uses two instances of the `<unitdate>` element (perfectly contemplated by the EAD Schema) and used the ISO standard for the date format (YYYY-MM-DD), e.g., `<unidate datechar='initial'> 1436-00-00 </unitdate> <unidate datechar='final'> 1441-00-00 </unitdate>`.

The length of our date format was set to be fixed (YYYY-MM-DD) for easy parsing and sorting. The description of incomplete information was resolved using zeros to represent unknown date components.

2.1.2 Numbered versus Unnumbered Components

In the EAD specification each level of description is represented by an element called Component – <c>. This element serves as a wrapper for all the description fields that comprise the archival description.

Components can be represented with two distinct, yet equivalent, syntaxes:

a) making use of `<c1>, <c2>, …, <c12>` elements whose numeric value represents the depth of the component within the description tree;

b) using a more general representation that doesn't specify the component's depth, i.e., a simple <c> element.

We chose to use the <c> element as the wrapper; since its position within the XML document already provides its depth and because it's easier to process by XML Style-sheet Transformations (XSLT).

2.1.3 Level and Otherlevel attribute

Each <c> element is accompanied by an attribute *level* that specifies the correspondent level of description. The EAD specification allows the use of the values class, collection, fonds, subfonds, series, subseries, file, item, recordgrp , subgrp and otherlevel.

These values did not satisfy the needs of ADP; which were: Fonds, Subfonds, Section, Subsection, Subsubsection, Series, Subseries, Instalation Unit, Composed Document and Simple Document.

Using the attribute value *otherlevel* one can use an optional attribute called *otherlevel* and specify a custom value, e.g. `<c level='otherlevel' otherlevel='Instalation Unit'>…</c>`

2.1.4 Mixed content elements

EAD provides a set of special elements that can be used anywhere in the description to annotate significant parts of the text. These are called floating elements and give the archivist the capacity to enrich a description with additional semantics. One can enhance the contents of descriptive field by marking special parts of the text with additional tags, e.g., `<scopecontent> The founder of this institution, **<person>***John Marshal***</person>**, was born in **<date>***August 1904***</date>** and soon...</scopecontent>`.

All elements that can have a floating element as a child are said to be elements with mixed content and records with elements like these are called half-structured records.

The importance of these elements in this context is well recognised although, its use poses considerable problems from the graphical user interface point-of-view. How shall one allow users to mark-up special parts of the description without displaying any tags? Using some sort of colour scheme to highlight the annotated text? That could, certainly, be a way to do it; even so, storing this kind information in a relational database poses considerable problems. Mapping the intricate structure of this type of information to the relational model produces a massive number tables resulting in a tremendous effort while retrieving information. Half-structured data can not be efficiently stored and managed in a relational model. All floating elements were, therefore, left out. This means we do not have half-structured records and that we do not need to face the problem of half-structure data storage.

2.1.5 Control Access

Each description record could include keywords extracted from a custom thesaurus to identify any relevant topics using controlled language. By using a controlled set of keywords, searching for a specific item is easier and results are more accurate.

The keywords are defined in a two-level basis: category and term. By doing so, we have reduced the ambiguity of some particular terms or expressions, e.g., the term “John Frederic Kennedy” can be interpreted as a past USA president or a contemporary airport. If one describes the term as “Airports/John Frederic Kennedy” the ambiguity is lost. The controlled language keywords are included in the description using the elements: controlaccess/list/defitem/label and controlaccess/list/defitem/item.

The system administrator is the only person allowed to edit the custom thesaurus.

3 DIGITALISATION OF PAPER-BASED FINDING AIDS

A large quantity of finding aids inside ADP existed solely on paper. They consisted mainly on typewritten documents that were produced at the Archive in its early years. We wanted to test the viability of using character recognition (OCR) to digitalise a large number of poor quality documents. It was our belief that it would be more efficient (i.e., faster, cheaper, lesser human resources) than a manual approach.

The digitalisation process comprised the following steps:

a) **Digitalisation/Scanning;** this step consists in processing the paper-based documents by means of a scanner. The output of this procedure is a set of image files, one per each page of the original document. This operation was executed long before the beginning of the

project; thus no statistic data was recorded on how long it took to conclude. The documents were digitalised in grey-scale 300 dpi.

b) **Model Identification;** in order to define the appropriate annotation strategy for each type of document, one must identify the models of documents according to their layout. We have identified three distinct models.

Model	Structure	Annotation technique
A	Very unstructured. The information on each record was extremely variable.	XML-based, according to a custom built schema
B	Very structured. Information organised in a table.	State-based annotation strategy where the opening of a tag defined a new state of interpretation by the parser (e.g. section tags in LaTex)
C	Slightly structured. Record information was reasonably variable.	State-less annotation strategy, using record separators.

c) **Optical character recognition;** after digitalisation, image files were processed by an optical character recognition (OCR) package[2] that converted those images into text files containing the wordings of the original documents.

d) **Error correction;** even though OCR technology has greatly improved over the past years, character recognition packages are still far from perfect. If the original document lacks quality, the results of recognition carry several grammatical errors. The resulting text must, therefore, be revised and errors corrected by a human operator. It was used the OCR software correction feature that underlined words that don't belong to a certain dictionary.

e) **Annotation;** in order to properly process the recognised text, this must be annotated. The annotation process consists of introducing special marks (i.e. tags) in the original text in order to enrich it with semantics that can be processed automatically by a computer program.

f) **Migration;** this step consists in developing and applying transformation scripts to the annotated text with the main objective of converting it to its definitive format.

Two full-time (7 hours/day) operators were recruited to perform the recognition, correction and tagging of the digitalised documents.

The time necessary to convert paper-based finding aids can be calculated by adding the time needed to perform each of the previously described steps. In this project, the digitalisation time was not recorded since it was done before the beginning of the project; recognition and correction phases were performed simultaneously using the Omnipage package; recognition and migration times are considered insignificant when compared to the other time components involved.

[2] In this project, it was used the package Omnipage Pro 12 to perform character recognition.

The total time necessary to convert a document can therefore be calculated using the following formula:

$$t_{conversion} = t_{recognition+correction} + t_{anotation}$$

Equation 1: Simplified formula for calculating the conversion time.

It is important to emphasise the fact that the original finding aids were of very poor quality causing recognition to be extremely imprecise; a long time was spent correcting the errors introduced by the OCR package and the density of information that could be found on each record was not, by any means, constant (i.e. the quantity of useful information that could be extracted from distinct records could be extremely variable, thus affecting the time needed to annotate those records).

The following tables summarise the time consumed by each of the described steps.

Model A	**Statistics per Page**			**Statistics per Record**		
	$t_{rec+corr}$ **(minutes)**	$t_{annotation}$ **(minutes)**	$t_{conversion}$ **(minutes)**	$t_{rec+corr}$ **(minutes)**	$t_{annotation}$ **(minutes)**	$t_{conversion}$ **(minutes)**
1	7,1	14,6	21,6	0,9	1,9	2,8
2	14,8	9,5	24,3	3,4	2,2	5,5
3	7,1	15,9	22,9	0,5	1,2	1,7
4	6,8	12,3	19,8	1,2	2,1	3,4
5	8,3	30,0	38,3	2,1	7,5	9,6
6	11,7	10,1	21,8	2,6	2,2	4,9
7	4,8	13,7	18,6	0,3	0,7	1,0
Average	**8,6**	**15,2**	**23,9**	**1,6**	**2,5**	**4,1**

Table 1: Conversion statistics for finding aids of model A.

Model B	**Statistics per Page**			**Statistics per Record**		
	$t_{rec+corr}$ **(minutes)**	$t_{annotation}$ **(minutes)**	$t_{conversion}$ **(minutes)**	$t_{rec+corr}$ **(minutes)**	$t_{annotation}$ **(minutes)**	$t_{conversion}$ **(minutes)**
1	4,7	11,9	16,5	0,9	2,2	3,1
2	9,7	11,4	21,0	1,0	1,2	2,3
3	8,2	5,3	13,5	0,8	0,5	1,3
4	13,1	4,8	10,4	1,5	0,6	1,2
5	7,0	11,3	18,3	0,3	0,5	0,8
6	9,8	23,0	31,3	0,2	0,5	0,7
7	6,9	11,5	18,5	0,3	0,5	0,7
8	7,8	5,6	11,1	0,6	0,4	0,8
9	7,2	10,0	17,2	0,6	0,9	1,5
10	5,2	5,2	10,4	0,7	0,7	1,3
Average	**8,0**	**10,0**	**16,8**	**0,7**	**0,8**	**1,4**

Table 2: Conversion statistics for finding aids of model B.

Model C	Statistics per Page			Statistics per Record		
	$t_{rec+corr}$ (minutes)	$t_{annotation}$ (minutes)	$t_{conversion}$ (minutes)	$t_{rec+corr}$ (minutes)	$t_{annotation}$ (minutes)	$t_{conversion}$ (minutes)
1	13,5	19,5	33,0	5,8	8,4	14,3
2	17,7	14,5	32,1	7,5	6,1	13,6
3	14,9	10,9	25,8	6,5	4,8	11,3
4	11,3	14,5	25,8	4,4	5,7	10,0
5	11,4	14,6	26,0	4,0	5,2	9,2
6	7,2	7,8	15,0	2,8	3,1	5,9
Average	**12,7**	**13,6**	**26,3**	**5,2**	**5,5**	**10,7**

Table 3: Conversion statistics for finding aids of model C.

The conclusions one can extract from this experience is that, for extremely heterogeneous documents such as these, one might expect an average conversion time (i.e., digitalisation + correction + annotation) of 22,3 minutes per page [4].

No measurements have been made for the conversion of the same documents using a manual approach, so no comparison of the two methods can be performed. Nevertheless, it is important to point out that, even though the archivists had to acquire new knowledge on how to perform the necessary procedures, they felt that converting the documents manually would be much more fatiguing than the semi-automatic approach; something that would lead to numerous human errors due to operator unsatisfaction and weariness.

4 MIGRATION OF DIGITAL FINDING AIDS

Besides paper-based documents, many digital files and databases existed at the Archive. To name some examples, digital material was found in Word, Excel, XML, Access and Filemaker databases.

The biggest challenge though, was the conversion of two CDS/ISIS hierarchical databases. The field structure in those databases was not fully compatible with the EAD standard, so some field multiplexing had to be performed (i.e. information stored in a single field had to be separated and assigned to several, more specific, ones). The separation of data was based on a set of natural language rules (subsequently translated to regular-expressions) that were deduced by archivists while analysing the original descriptive text. Different cues were identified and used to create these rules. A simple example of a rule could be the identification of a particular sequence of words (or its derivatives) at the beginning of a sentence. When such a pattern was detected, the whole sentence could be copied with a high degree of certainty to a specific destination field. When unknown text patterns were found by the system, they were presented to the operator and a new set of rules were appended to the system.

Given the heterogeneity of the file formats and the structural differences between each document, the migration of digital finding aids involved creating special purpose scripts or transformers for each document. The migration workflow usually involved exporting the original database or document file to a comma separated format, XML or, in extreme cases, to unstructured text and then process it using a combination of different techniques: Perl scripts and regular-expressions, XML Style-sheet Transformations, XML-DT (for XML transformations that needed regular-expressions), etc.

5 ARCHIVAL MANAGEMENT SOFTWARE

The migration stages resulted in several hundred XML files marked up according to the EAD schema. The development of software capable of storing, managing and retrieving all this information was urgently needed.

A database was created and the EAD/XML files where imported using middleware-based application specially designed for this purpose (described in section 8). This middleware application converted the hierarchical structure of the EAD files to the relational model. The same middleware was used by the Archival Management Software to interact with the database.

Compared to other off-the-shelf solutions, the Archival Management Software offered a set of innovative features:

a) **Relative referencing;** It is common in archival management software to identify items with an absolute path reference that uniquely identifies a particular item and also positions it on the description tree. It was very common for archivists to provide incorrect references to items, especially on large trees. Because the reference was used to position the item in the tree, changing a single letter could move it to an erroneous branch of the tree or even to a completely different collection. To prevent this kind of human mistakes, references to records were handled in a relative manner.

b) **Constrained hierarchy;** Hierarchy could be corrupted if some constraints were not imposed by the software. There is an order by which levels of description can be organised, i.e., a subseries must be child of a series and never the other way around since that would violate the rules of description.

c) **Detection of errors in descriptions;** While archivists are describing fonds, errors and omissions are frequent. The software was able to generate a list of errors and warnings upon user request. The software was able to point out missing dates, swapped initial/final dates, empty mandatory fields, etc.

d) **Inference from lower levels;** In archival finding aids, special information can be automatically attained from lower levels of description. For example, each level (or node) of description comprises two dates corresponding to the earliest and latest date (range dates). For non-leaf nodes, this information can be automatically calculated by traversing all its child sub-trees and collecting the correspondent earliest and latest dates.

e) **Drag and Drop;** Drag and drop was used to move records within the description tree. This eliminates the chances of reference inconsistencies, since references were automatically updated by the system without human intervention. Copy/Cut & Paste of a selection of nodes was also possible, greatly improving the time needed to describe regular collections.

f) **Cooperative work;** The software allowed different users to work in a collection at the same time.

g) **Reports**; Thirteen different reports were developed. This allowed archivists to create printed versions of finding aids. These reports could be exported to PDF for easy exchange of information.

h) **Accounts and Logging;** The system provided a simple username/password authentication mechanism to recognise its users. A special account was created for the System Administrator that offered some administrative options such as displaying work statistics (e.g. how many records per day are being described at the archive), user account management, statistics about archival descriptions (total number of records per fonds,

description completeness, etc), server load monitoring, etc... Logging of user actions such as login/logout times, record creation/removal/updating and others was also performed.

i) **Controlled language**; A custom thesaurus was developed that allowed administrators to define the list of keywords that could be used by archivists to describe items.

j) **EAC;** The software supported the EAC [5] (Encoded Archival Context). The EAC standard is used to exchange authority controlled information about the fonds in custody.

k) **Import/Export;** The system was able to import EAD/XML files and comma delimited data (CSV). It was also able to export EAD/XML and DScribe CALM Natural Format files.

A simpler standalone version of the archival management software was developed to be distributed to external organisations that periodically send documentation to the Archive. The producers of this documentation can now provide their own archival descriptions reducing the time necessary for incorporations. The integration of these external finding-aids with the ones already hosted by the Archive was straightforward since EAD/XML files were used to exchange information between the applications. Fonds merging is also contemplated [6].

6 WEB-BASED SEARCH ENGINE

A web-based search engine was developed allowing local and remote users to find and browse the Archive's collections.

A large percentage of users that access the Archive's web-site live abroad, especially in countries such as Brazil, France, Switzerland and United Kingdom. Allowing users to browse the Archive's collection remotely, significantly improves the quality of the service provided.

The web interface provides a tree-based view of a collection, allowing the user to browse consistently through the contents of entire fonds.

The web interface also allows users to browse digital objects, i.e., hierarchically organised digital copies of documents in custody [7].

7 DIGITAL OBJECTS MANAGEMENT SOFTWARE

Archives are becoming aware of the extreme importance of digital preservation. In the near future, most documentation will be delivered in digital formats and archives must be prepared, technically and logistically, to cope with this new form of documentation.

The DigitArq project gave a step forward in this direction, by developing an infrastructure capable of embodying and managing digital documents. Although, in this first stage, digital objects are limited to image files, the infrastructure it self, its functionalities and metadata were created with a serious consideration for future expansions [8].

Digital representations of items in custody can be associated to the same finding aids used to retrieve original documentation. Customers can now consult online copies of documents, reducing the number of book requests, thus lessening staff workload. This also constitutes a way to preserve original materials since customers no longer need physical contact with the documentation. Besides that, the online publication of documents allows customers from foreign countries to read and buy high resolution copies of documents, opening new ways for business – electronic commerce.

Some of the functionalities offered by the Digital Objects Management Software are summarised next:

a) Definition of digitalisation profiles, i.e., a set of configuration parameters, previously defined by an operator, that lead to optimal digitalisation results for a specific type of document.

b) Safe storing of images to Compact Discs (CDs).

c) Automatic assignment of references to CDs and respective physical location management.

d) Creation, Importing and updating of Digital Objects (please note that, in this context a digital object might comprise a set of image files that only make sense together).

e) Association of metadata to individual images as well as to whole digital objects.

f) Automatic creation of image derivates according to the digitalisation profiles. Image derivates are used for different publication purposes: online, printing, selling, etc.

g) Construction of hierarchical digital objects, i.e., images can be structured hierarchically depending on how they are logically organised within the original document.

h) Media monitoring, e.g. shows an alert when CDs expire.

8 HIERARCHICAL VS RELATIONAL INTEGRATION

In order to ingest EAD/XML files into a relational database, a middleware solution based on an abstract class called LazyNode was used. LazyNode class defined a comprehensive set of methods that allowed the programmer to fully manipulate the hierarchical structure of finding aids.

Method signature	*Description*
Sub Upload	Updates database with the information stored in memory.
Sub Download	Brings information from the database to the application level. This accounts for the refreshing of information as well.
Function Children() As LazyNodeCollection	Returns a collection of child nodes.
Sub RemoveChild(ByVal child As LazyNode)	Removes a child node.
Sub AppendChild(ByVal child As LazyNode)	Appends a new child node.
Function HasChildren() As Boolean	This method returns True if the selected node contains child nodes, and False otherwise.
Function CreateNode() As LazyNode	Creates an instance of a node. This instance should be appended to a node.
Function Clone() As LazyNode	Clones the selected node.
Function Parent() As LazyNode	Returns the parent node or *null* if the selected node is a root node.

Table 4: LazyNode abstract class method definition.

Apart from the described methods, *sets* and *gets* were implemented for each property that comprised a node description (e.g., Unit Title, Unit Reference, etc.).

The abstract class was called LazyNode because the information was loaded from the database to the application only when required. By doing it in this way, the application was not overload with unnecessary information and network traffic was severely reduced.

Two implementations of the LazyNode abstraction were developed: EADLazyNode and SQLLazyNode. The first was responsible for manipulating EAD/XML files while the later dealt with the intricacies of the communication between the application and the relational database.

This middleware application is transparent in what concerns accessing different storage types/formats. From the application point of view, the description trees can the stored in any format provided that the nine abstract methods are correctly implemented over the underlying storage layer.

Transparency allows the development of *catamorphisms (algorithmic patterns)* over different storage types. This is a strongly welcome property since most operations performed automatically by the description management software consist of entire fonds traversals, e.g. revision of fonds, inference, construction of description summaries and reports, etc. Another important advantage of this model is its capacity to convert data between the two different storage formats. The conversion is attained by traversing the source description tree and recreating it in an empty destination tree.

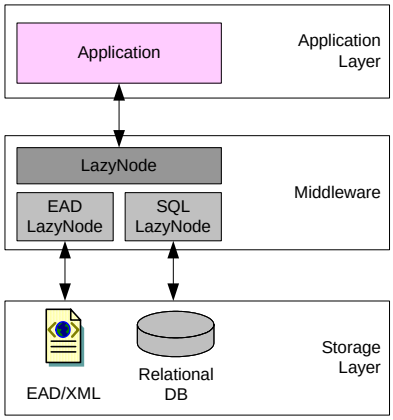

Figure 2: Layer-based architecture.

Although the model is simple to use and appropriate to the requirements of our project, it carries some disadvantages. The information is hierarchical, not relational, so in order to perform some basic tasks, like finding a specific node, a full tree traversal must be performed by the application (this issue was tackled differently by the web-based search engine, as the tree was constructed from bottom to top).

Full traversal operations (e.g. fonds revision) can take up to 1,4 minutes for the average-sized tree (around 1000 nodes). Although performance was not astounding, consulting with users showed us that it was not a major problem since this kind of operations were performed only once or twice a day.

9 CONCLUSIONS AND FUTURE WORK

In this paper we present all the stages that comprised a project called DigitArq that was carried out at the Arquivo Distrital do Porto. Some lessons can be learned from this experience. For instance, the statistics presented concerning the digitalisation, correction and annotation of irregular documents can help managers to schedule equivalent tasks in their own projects.

At the present time, the introduced Digital Objects Management Software only supports image files. Future developments on this software should make it capable of working with other types of digital objects.

Along this paper, special attention was given to a middleware solution capable of storing hierarchical information in relational databases. The proposed solution grants transparency in accessing different storage types. The conversion between data models can be accomplished by a simple traversal of the source tree. Catamorphisms can be easily implemented over any storage format since a common unique interface is exposed to the application layer. Implementations for EAD/XML files and Relational Databases were developed.

Because the model is general-purpose it doesn't take advantage of any special features provided by relational databases. In this context, future work could be developed to optimise its implementation over this type of storage. The use of stored procedures can be a step in the right direction.

Faster loading and network traffic reduction could be accomplished by adding cache and pre-loading mechanisms to the middleware layer. The implementation of these methods would require additional control data to certify that cached information is coherent at all time.

10 REFERENCES

1. ICA, *ISAD(G): General International Standard Archival Description, Second edition*, in *ICA Standards*. 1999, International Council on Archives.
2. LC, *EAD - Encoded Archival Description*, Library of Congress.
3. Caplan, P., *Metadata fundamentals for all librarians*. 2003, Chicago: American Library Association. ix, 192 p.
4. ADP, *Projecto Digitarq: OCR Statistics - Relatório Final*. 2004, Arquivo Distrital do Porto: Porto.
5. Group, E.W., *Encoded Archival Context (EAC)*.
6. ADP, *Projecto Digitarq: Módulo Conversão/Descrição - Relatório Final*. 2004, Arquivo Distrital do Porto: Porto.
7. ADP, *Projecto Digitarq: Módulo Pesquisa Web - Relatório Final*. 2004, Arquivo Distrital do Porto: Porto.
8. ADP, *Projecto Digitarq: Módulo Arquivo Digital (GOD) - Relatório Final*. 2004, Arquivo Distrital do Porto: Porto.