UNIVERSITY OF AMSTERDAM

UvA-DARE (Digital Academic Repository)

System evaluation of archival description and access

Zhang, J.

Publication date
2011

Link to publication

Citation for published version (APA):
Zhang, J. (2011). *System evaluation of archival description and access*. [Thesis, fully internal, Universiteit van Amsterdam]. IR Publications.

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, P.O. Box 19185, 1000 GD Amsterdam, The Netherlands. You will be contacted as soon as possible.

UvA-DARE is a service provided by the library of the University of Amsterdam (https://dare.uva.nl)

Download date:18 Nov 2026

Chapter 2

Usage of XML Retrieval for Archival Access

This chapter investigates how EAD can be used for search systems driven by EAD finding aids, and how XML retrieval techniques can support such a system. As proof of concept and research baseline, we build a search system driven by EAD finding aids and supported by XML retrieval techniques.

2.1 Introduction

Archival access is increasingly occurring with electronic resources and shifting to the World Wide Web. The technical standard EAD contributes to this shift. We see a rendezvous between traditional information representation by archivists and information access technologies on the World Wide Web. Archival finding aids in EAD attempt to guide people to archival records. We see more usages of such technologies for archival access.[1] With the widespread use of XML in archives, there are promises, but also pitfalls. The application of XML retrieval on EAD finding aids seems natural, but is it? This leads to the main research question from a system-centered point of view, which is:

- *Q1: How do XML retrieval techniques support a search system driven by archival finding aids in EAD?*

[1] Increasingly, Web 2.0 (O'Reilly, 2007) is becoming a denominator for interactive archival access. The Polar Bear Expedition project of Yakel et al. (2007) is a realization of these ideas, in which an interactive personalized archival access system—driven by finding aids that users can comment on—has been designed. Samouelian (2009) investigates the extend of to which Web 2.0 features have been integrated into archival information systems, and suggests that archival professionals are embracing Web 2.0 to promote their digital content and redefine relationships with users. On the other side of the coin, archival access may be improved by attracting users through social media as Twitter or Facebook, but Crymble (2010) shows that participation in social media does not necessarily lead to more users. We note that in this dissertation, we only focus on IR aspects of archival finding aids in EAD.

To answer the first sub-question, we develop a system to move towards a tangible construct. Evans and Rouche (2004, p.315) point to a methodological issue by discussing the use of systems development research methods, and already suggest that adopting a user-centric prototyping approach in a research context allows for exploration of the interplay between theory and practice, advancing the practice, while also offering new insights into theoretical concepts. Therefore, we add a component in archival research methods (Gilliland-Swetland and Mckemmish, 2004). Duranti (2001) refers to archival science as a system itself, where the properties of the system can be investigated, supporting the development of new knowledge and as a demonstration of the stability of archival theory.

1.1 How can archival description principles that underlie archival finding aids be translated to an archival information system design?

Archival finding aids in EAD have multi-level descriptions in accordance with *General International Standard Archival Description* (ISAD(G), International Council on Archives (1999)).[2] This means that these finding aids can be approached from different levels of details. The second sub-question explores meaningful usages of XML retrieval techniques on EAD finding aids.

1.2 What levels of detail are possible when providing focused intellectual access to archives?

Section 2.2 explains the technical background of XML, EAD, and XML information retrieval. Section 2.3 discusses archival access with EAD finding aids. We describe our system called *README* in Section 2.4. The conclusion in Section 2.5 offers an interpretation of archival access in this digital age, after looking at the theory, real-world instances of archival access applications, and the development of such an application—which is our research baseline.

2.2 Related Work

This section introduces and explains XML, EAD, and XML retrieval with their key concepts relating to the chapter's problem statement.

[2] EAD is an encoding standard, and it enables sharing archival records among archives. This is reflected in the close link between EAD and ISAD(G), which has been approved by the International Council of Archives (ICA) as a standard that defines the elements that should be included in an archival finding aid. It is a hierarchical model. The advent of archival finding aids in EAD has made ISAD(G) particularly useful, because it enables the sharing of records between archives by upholding four principles that all archives in theory should share in their multi-level description of each record, namely (i) the description of the general to the specific; (ii) information should be relevant to the level of description; (iii) descriptions should be linked between levels by making clear the level of the unit of description; (iv) non-repetition of information by using the hierarchy of description.

2.2.1 Extensible Markup Language

Goldberg (2008) presents an illustrative guide on Extensible Markup Language (XML, Bray et al. (2008)). It was designed to manage information. Moreover, it is also a specification for describing the structure of that information. XML evolved from Standard Generalized Markup Language (SGML, Goldfarb (1984)). SGML is a metalanguage—a language that describes languages—that has proved useful in many large publishing applications. However, a problem to adoption is that SGML is too general and is more complex than Web browsers can cope with (Bosak and Bray, 1999). Therefore, Bosak and Bray (1999) created XML consisting of rules that anyone can follow to create a markup language from scratch. EAD is such a markup language.

The basic XML concepts relevant to our research on content-orientated information retrieval—and which we discuss here—are the notion of an *element*, the distinction between *well-formed* and *valid* XML with for example a Document Type Definition (DTD). We show how XML documents, such as an EAD finding aid, can be viewed as *trees*.

Element Angle-bracketed labels, such as `<titleproper>` or `<c01>` in Figure 2.2, are called tags. In XML, there is always a begin tag as `<titleproper>` and its corresponding end tag `</titleproper>`. Anything between this pair of tags is called an XML element. This can be another element or more, which is then called a sub-element, or text (content) only, or instances of both which is called mixed content.

Well-formed The Web's main language is Hypertext Markup Language (HTML), which is a electronic-publishing language, mainly for easily creating the layout—or appearance—of Web pages using a W3C pre-defined set of tags (Berners-Lee et al., 1994; Bosak and Bray, 1999). Unlike HTML, XML is a markup language that is extensible, because it allows for the definition of custom tags that are meaningful to people. However, any XML document must satisfy certain rules in order to be XML. When an XML document satisfies these rules, it is well-formed, and only then it can be computationally processed. Goldberg (2008) lists these rules, such as the case-sensitivity of tag names, and the non-overlap of XML elements, where there is only one root element and each element has only one parent, so as to enforce a strict hierarchy of elements.

Valid An XML document is only XML, when it is well-formed, i.e. conforms to a syntax. However, additional rules can be enforced in terms of the vocabulary (which elements and attributes can be used) and grammar (frequency of elements and nesting), therefore effectively allowing the definition of a custom markup language. A document type definition (DTD) is often used to make a set of XML documents also valid. DTDs are essentially extended context-free grammars expressed in a notation that is similar to the extended Backus–Naur form (EBNF) notation (Knuth, 1964;

Wirth, 1977).

Tree In computer science, a tree is a widely-used data structure that equals a hierarchical tree structure with a set of linked nodes. Given the hierarchy of elements, an XML document also consists of a hierarchical set of linked nodes. Let us take Figure 2.2 as an example of an XML document. We see that <ead> is the root node, which is a member of the node set that has no parent (or 'superior'). The lines connecting the nodes (which equals elements) are named 'branches.' Element <unitid> is a node without children, and is called a 'leaf' node. The nodes in the tree have several depths, where the root node has a depth of one, element <archdesc> has a depth of three, or <c01> has a depth of four. Any tree can be mapped to a DTD, and a DTD can be used to construct a tree.

Bosak and Bray (1999) sketch what can be expected from XML. An expectation is that XML will solve some of the Web's biggest problems, which is the difficulty to find the one piece on the Web that you need. Common uses of the tree data structure, and thus XML, is to manipulate hierarchical data and make information easy to search. XML is primarily used to exchange information between either (i) people and systems, (ii) among people, or (iii) among systems. In the case of the former, the logical structure in an XML document is called document-centric, and in the case of the later, it is called data-centric. In XML there is the fundamental concept of separating the logical structure of a document from its layout, where the latter is usually created in Cascading Style Sheets (CSS) or with HTML, possibly in conjunction with XSLT (Goldberg, 2008). However, it is the exploitation of the logical structure for information access that is of primary interest in XML retrieval. What is the logical structure of EAD?

2.2.2 Encoded Archival Description

Forde (2005) discusses changes that have occurred in access and preservation, and mentions surrogacy programmes. The initiative that started in 1997 to materialize by creating 'electronic' finding aids in a technical standard called *Encoded Archival Description* (EAD, (Pitti, 1997)) is also a surrogacy program. EAD is jointly maintained by the Library of Congres (LoC) and the Society of American Archivists (SAA). Surrogacy programmes enable long-term use and remote continuous access to archives. Microform technology was a step towards this access (Forde, 2005, p.194), but with the emergence of the World Wide Web (WWW, Berners-Lee et al. (1994)), remote and continuous access is possible because of online access.

The sharing of geographically 'scattered' records is a motivation for the development of EAD (Pitti, 1997, p.269). An important promise of automation is the desire of archivists to have the ability to cooperate with each other, and to exchange records with other systems (Bearman, 1979, p.183). Haworth (2001,

p.21) notes that the emergence of EAD as standard could not be avoided given the plethora of 'electronic' finding aids independently created and contained in a variety of software-independent systems which limited their exploitation and sharing. Kiesling (1997) remarks that strict and unambiguous adherence to the standard is needed to achieve this goal, and EAD may be the catalyst.

An increasing number of archives and manuscript libraries, and also museums (e.g. Chandler (2002)), use EAD to encode archival descriptions that describe unique primary resources in the form of archival materials, such as corporate records and personal (hand-written) papers, and for access (Pitti, 1999). For example, Hill et al. (2005) discuss three different online services providing access to finding aids relating to three different archives, but the similarity is the key role of EAD and its application for creation, storage, indexing, searching and presentation of finding aids. These archival collections may have millions of unique items, which can be in any form or medium. For example, plans, drawings, charts, maps, photographs, audio, and video (Pitti, 1999).

The archives are organized and described hierarchically. EAD consists technically of a set of descriptive elements to describe the archives. Figure 2.1 depicts a part of the official EAD Document Type Definition (DTD). The version of EAD researched in this dissertation is the official 2002 version.[3] The three highest level elements are `<eadheader>`, the optional `<frontmatter>`, and the heart of an archival finding aid consisting of archival descriptions in `<archdesc>`—the body of archival materials (Ruth, 1997, p.320). These descriptions usually consist of a Descriptive Identification Element `<did>` and a Description of Subordinate Components `<dsc>`. The components `<C`$_n$`>` of the whole are recursively nested in `<dsc>` within `<archdesc>`, where $n \in \{01, ..., 12\}$, see Figure 2.1 and 2.4. In other words, `<c02>` is the sub-component of `<c01>`, and so on. A component can also be unnumbered. The EAD files can be deeply nested and lengthy in content with thousands of pages (or more) (Pitti, 1999). Figure 2.2 shows the XML source code of the *Nationaal Archief* EAD finding aid '2.09.70' on the topic 'Bijzondere Rechtspleging' (in English: *Special Justice*). This topic deals with justice in the Netherlands after the Second World War for people who committed crimes during the War. Figure 2.3 sets out the HTML presentation of the *Nationaal Archief* system (of 2009) of this finding aid.[4]

2.2.3 XML Information Retrieval

Traditionally, IR has dealt with the retrieval of complete documents or full-texts. Usually, a user is only interested in a part of a document. As Hjørland (2000, p.33) argues, there is a distinction between document retrieval and fact retrieval, and puts forward the idea to store and only retrieve the facts or 'in-

[3] Retrieved 2011/01/11 from `http://www.loc.gov/ead/ead2002a.html`

[4] The *Nationaal Archief* revised its system in the middle of 2011. The examples used in this dissertation, and the studies conducted, are before this change.

```
<!ENTITY % m.desc.full
  '%m.desc.base; | dsc | dao | daogrp | note'>
<!ENTITY % m.did
  'abstract | container | dao | daogrp | langmaterial | materialspec |
   note | origination | physdesc | physloc | repository | unitdate |
   unitid | unittitle'>
<!ELEMENT ead
   (eadheader, frontmatter?, archdesc)>
<!ELEMENT eadheader
   (eadid, filedesc, profiledesc?, revisiondesc?)>
<!ELEMENT archdesc
   (runner*, did, (%m.desc.full;)*)>
<!ELEMENT did
   (head?, (%m.did;)+)>
<!ELEMENT dsc
   ((head?, tspec?, (%m.blocks;)*),
    (((thead?, ((c, thead?)+ | (c01, thead?)+)) | dsc*)))>
<!ELEMENT c01
   ((head?, did, (%m.desc.full;)*, (thead?, c02+)*) |
    (drow+, c02*))>
```

Figure 2.1: A part of the EAD Document Type Definition.

formation' contained in the documents.

The indexing and retrieving of elements, which may contain facts or 'information,' in XML documents is done using XML information retrieval (XML IR, Lalmas (2010)), which is a branch of information retrieval that deals with the retrieval of arbitrary parts of XML files given the document-centric XML structure, and attempts to use the XML markup of documents to the fullest for 'focused' information access by not only providing direct access to a whole document, but also to a part of it. This also conforms to the principle of least effort (Zipf, 1949), as it deals with the burdon of reading more information (i.e. full-text) than is actually necessary. We then also speak of *focused retrieval* (Joty and Sadid-Al-Hasan, 2007; Lalmas, 2010; Trotman et al., 2007). This markup represents the different granularities and complexities of these files. The structure is exploited to expose information. For example, in response to a user query on a collection of digital textbooks marked-up in XML, an XML retrieval system may return the content of a paragraph, section or chapter estimated to best answer that query.

As Lalmas (2010) notes, the concept of XML IR existed before it became known as such. She points out that with the introduction of XML in 1997, there were proposals to gain access to logically structured documents—with DeRose (1997) making the case for EAD—before the existence of the term XML IR. For example, Wilkinson (1994) shows in 1994 that knowledge of the structure of

```
<?xml version="1.0" encoding="utf-8"?>
<ead>
 <eadheader countryencoding="iso3166-1" dateencoding="iso8601"
langencoding="iso639-2b" scriptencoding="iso15924"
repositoryencoding="iso15511" findaidstatus="unverified-full-draft">
   <eadid countrycode="NL" mainagencycode="NL-HaNA"
publicid="-//Nationaal Archief//TEXT (NL-HaNA::2.09.70::Ministerie van
Justitie: Directoraat-Generaal Bijzondere Rechtspleging, Personeelsdossiers)//NL"
urn="2.09.70.ead.xml">2.09.70</eadid>
   <filedesc>
     <titlestmt>
       <titleproper>Inventaris van de archieven van het
       Directoraat-Generaal voor de Bijzondere Rechtspleging
       (1945-1958), met taakopvolgers en uitvoerende instanties
       (1945-1986); Deel III Personeelsdossiers</titleproper>
       <author>Centrale Archiefselectiedienst, Winschoten</author>
     </titlestmt>
     <publicationstmt>
       <publisher>Nationaal Archief, Den Haag</publisher>
       <date normal="2008" era="ce" calendar="gregorian">(c)
       2008</date>
     </publicationstmt>
   </filedesc>
   <profiledesc>
     <creation audience="internal">Deze digitale toegang is in
     <DATE normal="2009" era="ce" calendar="gregorian">
     2009</date>vervaardigd door het Nationaal Archief op basis
     van de richtlijn
     <title linktype="simple">Het gebruik van Encoded Archival
     Description op het Nationaal Archief versie 1.7.2</title>.
     Eindredactie: Henny van Schie,
     <date normal="20090429" era="ce" calendar="gregorian">6 mei
     2009</date>.</creation>
     <langusage>This finding aid is written in
     <language langcode="dut" scriptcode="Latn">
     Dutch</language>.</langusage>
     <descrules audience="internal"/>
   </profiledesc>
   <revisiondesc audience="internal">
     <change>
       <date normal="" era="ce" calendar="gregorian"/>
       <item/>
     </change>
   </revisiondesc>
 </eadheader>
 <archdesc level="fonds" type="inventory">
   <did>
     <head>Beschrijving van het archief</head>
     <unittitle label="Naam archiefblok:">Ministerie van Justitie:
     Directoraat-Generaal Bijzondere Rechtspleging,
     Personeelsdossiers</unittitle>
     <unittitle type="short">DGBR, Personeelsdossiers</unittitle>
     <unitdate label="Periode:" type="inclusive" normal="1945/1986"
      era="ce" calendar="gregorian">
     1945-1986</unitdate>
```

Figure 2.2: A part of the *Nationaal Archief*'s XML encoding of its largest EAD finding aid '2.09.70' (*'Inventaris van de archieven van het Directoraat-Generaal voor de Bijzondere Rechtspleging (1945-1958), met taakopvolgers en uitvoerende instanties (1945-1986); Deel III Personeelsdossiers'*) in terms of file size.

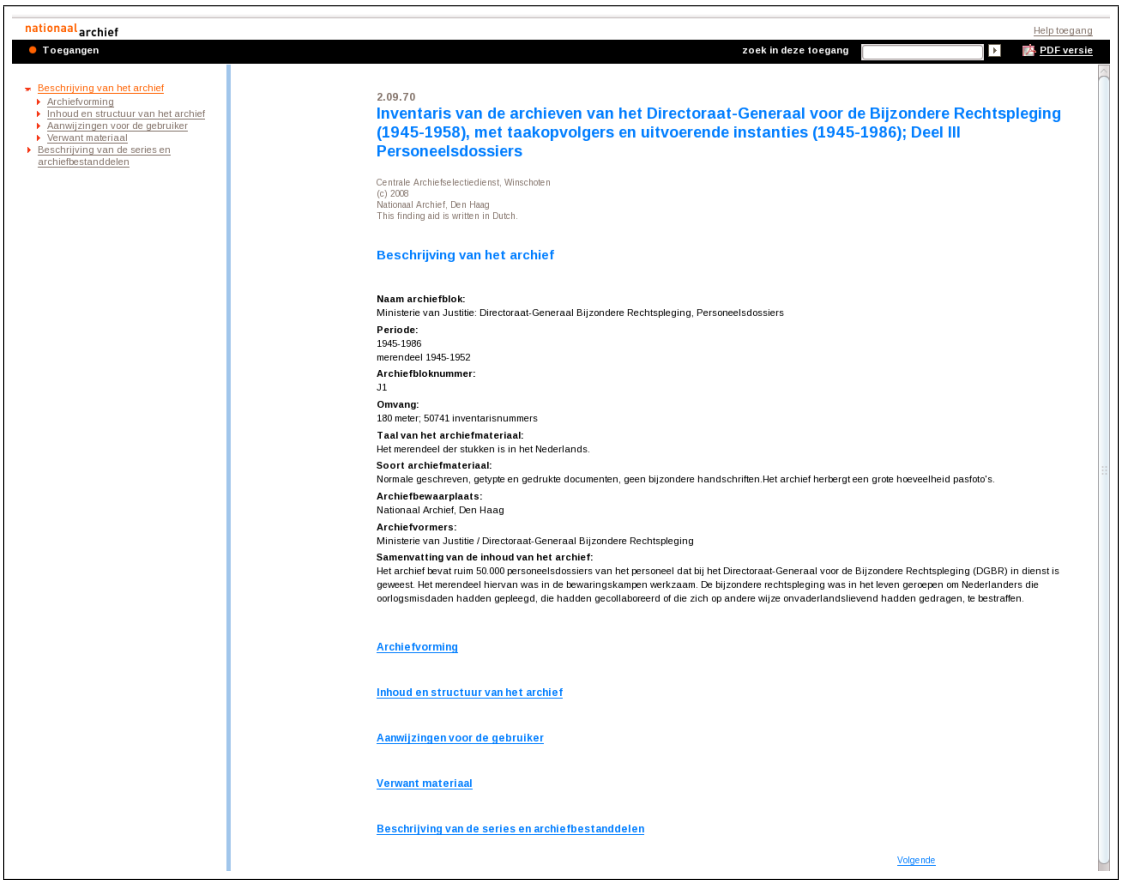

Figure 2.3: *Nationaal Archief* 's HTML presentation of its largest EAD finding aid '2.09.70' in terms of file size. Image courtesy of the *Nationaal Archief*.

documents can lead to improved retrieval performance. Numerous user studies with different experimental setups (e.g. Hammer-Aebi et al. (2006); Larsen et al. (2006); Pharo (2008)) also show that users prefer to interact with smaller nuggets of information. Furthermore, in the archival realm, Bearman (1979, p.188) implicitly referred in 1979 to the pivotal concepts in XML IR of granularity and direct access while the state of the art of that time was subject retrieval based on provenance and content indexing (Lytle, 1980).

The INitiative for the Evaluation of XML retrieval (INEX) started in 2002 to provide an infrastructure—common evaluation benchmark—for evaluating the effectiveness of content-orientated XML retrieval systems (Gövert and Kazai, 2002a). The predominant path to evaluate the system retrieval effectiveness is with the use of test collections, which usually consists of a set of documents, user requests referred to as topics, and relevance assessments specifying the set of 'right answers' for the user requests. The INEX infrastructure is defined by Gövert and Kazai (2002a, p.1) as a test collection of real world XML documents along with standard topics and respective relevance assessments.

We briefly recall the history and methodology of INEX. The topic develop-

ment process consisted of submission of candidate topics by the research participants themselves, then the participants had to judge a topic's top hundred retrieved results. This is manual and labor-intensive. INEX started with a test collection consisting of fifty-five topics inspired on articles from the IEEE Computer Society (Gövert and Kazai, 2002a). There were two types of topics: (i) the *content-only* (CO) topics that request only content related conditions with no knowledge of the document structure, and (ii) *content-and-structure* (CAS) topics which refer explicitly to the XML structure by specifying target elements (e.g. search only in the Summary). The research participants had so submit a ranked list of results that their system produced (i.e. 'runs') so as to compare their results. This type of experimentation continued in 2003 (Fuhr et al., 2004), 2004 (Malik et al., 2004), and 2005 (Malik et al., 2005).

In 2006, INEX used another XML document collection consisting of English articles from the Wikipedia project (Denoyer and Gallinari, 2006; Malik et al., 2006). A reason for this switch is that Wikipedia's content can also appeal to users with backgrounds other than computer science, as it was the case with the IEEE articles, making 'realistic' user studies—observation and measurement of user interaction with XML documents—possible. User studies in IR are driven by tasks, a user is asked to do something.[5] In IR, the domain-knowledge has a positive effect on search interaction (e.g. Lazonder et al. (2000); White et al. (2008)). To ask a random user to search for information related to their interests seems more suitable than a random topic.

INEX consists of different tracks, which are different research paths that investigate a part of the challenges of focused retrieval. Fuhr et al. (2007) discuss the Ad Hoc Track, which is the predominant track of INEX. Ad hoc search is described as a simulation of how a library might be used and involves searching in a static set of documents given a new set of topics. It follows the same methodology as set out in previous years, but also allows passage retrieval (Trotman et al., 2007). Simulating real use cases, this track also features tasks. Prior to 2007, there was the Thorough Task that asks systems to return any elements ranked by their relevance to the topic (Malik et al., 2006). Since 2007, INEX features three tasks: (i) the *Focused Task* that asks to return a ranked-list of non-overlapping results, (ii) the *Relevant in Context Task* that requests the return of a non-overlapping ranked-list of results, but grouped by document, and (iii)

[5] However, user studies are not necessarily task-based. Archival user studies have involved interviews to gather data unrelated to specific tasks. Understanding users is pivotal to evaluate and improve systems, and to understand the interaction—a dialog of a user elaborating and changing the needs and a system responding to this behavior—between systems and users (Lesk, 2004). However a difficulty is, as Lesk (2004, p.227) writes, that users are not always able to make clear what they actually search for. Studies in the archival domain have set the first steps towards understanding archival interaction by investigating the users' information seeking needs and behaviors. The study of Yakel (2004) researches what genealogical researchers want and do by conducting interviews with them. Ultimately, this type of studies helps us not only to understand the user needs, but to improve information systems for people.

the *Best in Context Task* that asks to return a single starting point of a document. Kazai et al. (2004) explain the issue of overlap as the retrieval of multiple overlapping (i.e. nested) result elements, e.g. a system may retrieve the same nugget of information from multiple nested elements. As the results of the user study of Hammer-Aebi et al. (2006) show, users do not appreciate overlap when they search for information, and as such, it must be removed. What does this all mean for EAD retrieval?

2.3 Archival Access: Information Retrieval with EAD

This section aims to investigate this chapter's first research question. We look at the utilization of EAD for information retrieval in terms of its particular structure and its applications by archives.

2.3.1 Retrieval with EAD

Ruth (1997) presents a structural overview that explains the use of EAD markup with illustrative code examples, and also the link with the ISAD(G) principles. DeRose (1997) also points to the difference in content and structure, and the need of structure to search in content.[6]

As illustrated in Tsikrika (2009), structured text retrieval supports the representation and retrieval of the individual document components defined by the logical structure as represented in a hierarchical document, such as an EAD file. This structure can be distinguished in two types of units Tsikrika (2009): (a) atomic units (or 'text content elements') that only contain text and no XML elements, and (b) composite units (or 'nested elements') that contain other units and can be further 'decomposed'. The same is true for EAD, see Figure 2.4, where atomic units such as `<unitid>` or `<unitdate>` are represented as leafs and composite units like `<did>` are non-leaf nodes. However, we extend this representation with *mixed content* nodes, i.e. elements that contain both text and other elements, where there are usually multiple text nodes, and each text node in an element can be retrieved separately in XML Path Language (XPath, Clark and DeRose (1999)). An instance of a mixed node could be the composite unit `<unittitle>` that may have been annotated with a semantic tag like `<persname>`.

There is no shortage of metadata in archival finding aids, but it is "just a matter of finding the right hook to make them more accessible" (Kiesling, 2001,

[6] The move from simple flat-form databases—such as catalogs and abstracts—to highly structured documents— such as EAD finding aids—makes the question of what kind of structure to represent increasingly more important (DeRose, 1997, p.304). An answer is to use structure that increases the ease of use (DeRose, 1997, p.308). The structure of finding aids is characteristically hierarchical with layers upon layers of substructure. Archival finding aids in EAD are also navigation tools with hypertext links, mimicking paper documents with indexes and table of contents (DeRose, 1997, p.307).

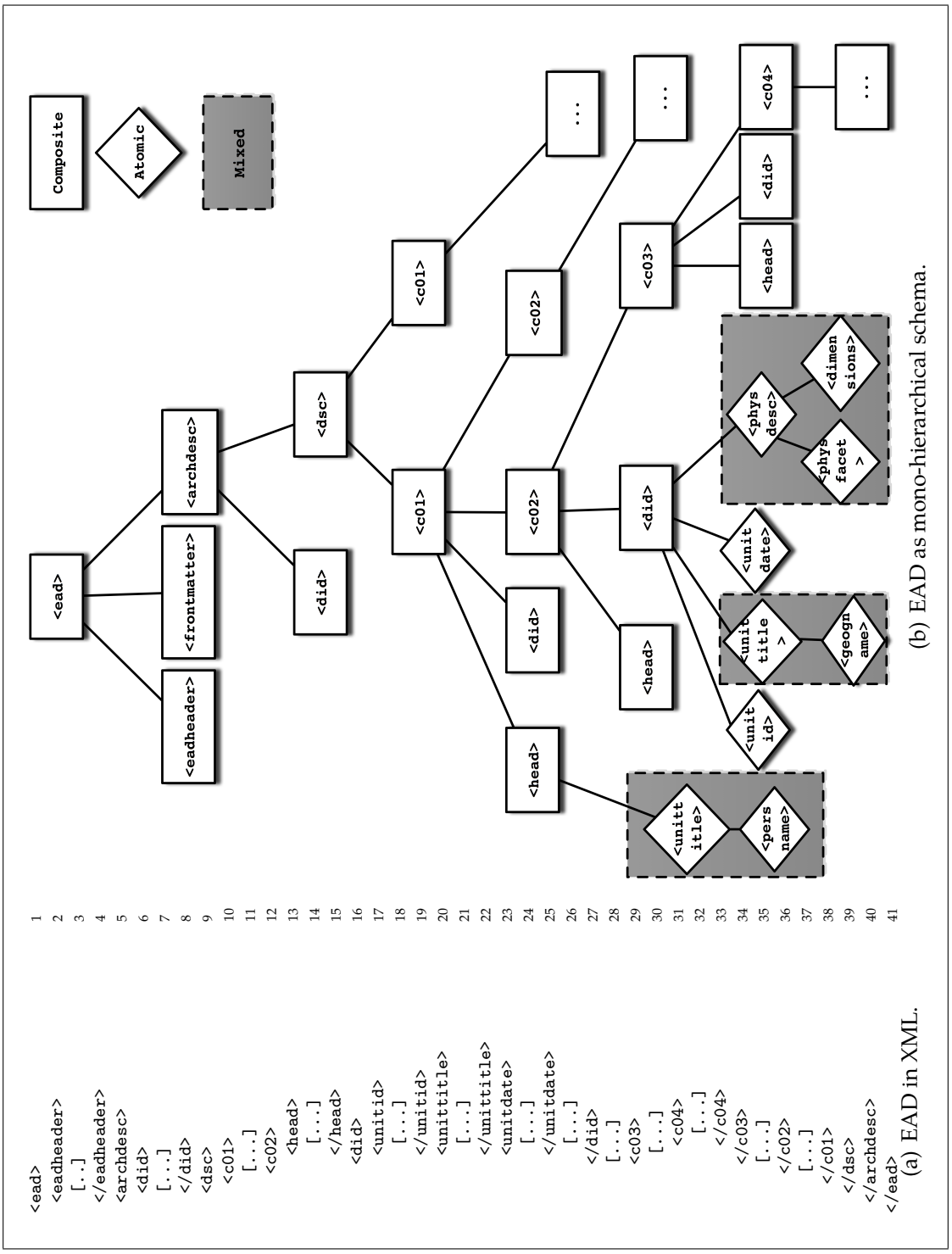

Figure 2.4: Representation of Encoded Archival Description as mono-hierarchical schema using XML elements as nodes for information retrieval (rotated clockwise).

p.87). XML retrieval techniques can be employed to deal with this problem, to maximally and most effectively exploit these 'hooks.' Using this markup, we could zoom into any of them—at the same time index and retrieve them.

2.3.2 Archival Metadata Retrieval Systems

Archival data encoded in EAD is structured data. Commonly used relational databases do not provide a perfect solution to store this type of data. XML databases are developed instead to provide a better solution to capture and preserve the richness of the structure in a data structure, though the question is still open whether archival users would seek information using powerful queries with XML structure, and whether it would be more effective for retrieval. A pointer to an answer is the study of Duff and Stoyanova (1998) which presents results of rating elements for access, and not surprisingly, the Title was perceived as the most useful element for retrieval. Another question is whether archivists would accept such technology, with Yaco (2008) pointing to the challenges of the technical implementation of EAD, and with Yakel and Kim (2005) also noting the overall slow adoption due to factors as the small staff size of many repositories, the lack of standardization in archival description, a multiplicity of existing archival access tools, insufficient infrastructure, and difficulty in maintaining expertise.

A multiplicity of existing archival access tools is available, including several open-source solutions, such as eXist (Meier, 2003). Other alternatives more commonly used for archival finding aids in EAD are PLEADE (EAD on the Web) developed in France and used by the French National Archives (Sévigny and Clavaud, 2003), Cheshire3 (Larson and Sanderson, 2006) as used by the Archives Hub in the UK, Archon developed at the University of Illinois (Prom et al., 2007; Schwartz et al., 2007), or the Digital Library eXtension Service (DLXS)[7]. Table 2.1 presents an overview of these and additional systems.

What are the features in terms of archival access do these systems offer? The analysis of search features available for Web sites driven by EAD (instances of digital archives) by Zhou (2007) shows that eighteen of the fifty-eight Web sites (or 22%) did not have any search features. Moreover, nearly half of the tested systems did not have a proper ranking, like ranking by call number order—which is for users often pointless—or the results are presented randomly. However, it should be noted that this study looked at Web sites only, which may be driven by the same systems. Similarly, another survey is presented by Huffman (2008) that shows an overview of search features of thirty-three Web sites driven by EAD that also include a search engine. A finding of that study is that less than half of the investigated archives had options to search in metadata fields, and most performed full-text searches. Although Zhou (2007) and Huffman (2008) look at XML-based EAD search systems, it is still unknown what

[7] Retrieved 2011/05/11 from `http://www.dlxs.org/docs/12a/intro/arch.html`.

systems have to offer to archives in terms of actual XML retrieval.

Therefore, Table 2.1 depicts our overview of the most widely used systems for (archival) metadata retrieval, so we do not investigate the archives, but the technical backbone that drives them. We investigate these systems by auditing their documentation and testing their implementations. The table shows the names of the systems, how the content and structure have been indexed, whether it supports a type of relevance ranking (or whether it is plain batch data retrieval), and institutions which have adopted the system. The systems are either available as open-source software, or as a commercial product. We make a distinction between field-based indexing and full XML-based indexing. The former refers to indexing based on certain metadata fields, like for example Names, Dates, Titles, Places, Subjects, Repository, or Summary, hence generally, indexing happens only with top-level components. The latter means that it is fully XML-based, where all XML structure of the original EAD documents is preserved, including their depth and breadth, and it is possible to optimally query the database with a XML query language like XQuery (Boag et al., 2002; Chamberlin, 2003).

For archives that seek to use open-source software, only Cheshire3 has full XML-based indexing with relevance ranking support, or else they are faced with field-based indexing. For commercial systems, there are the MarkLogic and TEXTML systems which provide both full XML-based indexing and information retrieval based on relevance ranking. All the (archival) metadata systems in Table 2.1 are distinctly different, though XTF partly uses Lucene (Hatcher and Gospodnetic, 2004) for information retrieval.

Besides retrieval using the EAD representation, there is the layout (presentation, user interface). The EAD Cookbook (Prom, 2002) consists of XSLT stylesheets to render a user interface from EADs. There are also content management tools. The Archivists' Toolkit is software particularly developed for this purpose (Westbrook et al., 2007). There are content management systems tailored to archival description that also include an access component, such as Archon—using a home-built module—or ICA-AtoM[8] that uses Lucene.

2.4 README System Description

This section covers the technical presentation of the README system.

2.4.1 Usage Scenarios

We can envision usage scenarios in designing a system. These scenarios are informal stories about user tasks and activities. Similarly, in the archival domain, and with EAD finding aids, we describe the following (original) usage scenarios, which have been randomly picked, that the system should be able to

[8] Retrieved 2011/01/11 from `http://ica-atom.org/`.

Table 2.1: Overview of commonly used (archival) metadata retrieval systems.

	System	Indexing	Relevance Ranking	Examples
Open-source	*Archon*	Field	no	UC Berkeley, Texas A&M University
	Cheshire3	Full XML	yes	British Library ISTC, Archives Hub, SHAMAN Digital Preservation Project
	eXist	Full XML	no	Columbia University Libraries
	Lucene/SOLR	Field	yes	Smithsonian, Rhode Island Archival and Manuscript Collections Online, University of Virginia
	PLEADE	Field	yes	French National Archives
	XTF	Field	yes	Online Archive of California, California Digital Library, Arizona Archives Online, Ohio Archives Online
Commercial	*Adlib Archive*	Field	no	Aston Martin Heritage Trust, National Archives of Ireland
	ARCHI-LOG	Full-text	no	Eastern Townships Research Center, numerous French-Canadian archives
	DigiTool	Field	yes	Boston College, Leiden University, Florida State University, Universidade do Porto, University of Melbourne
	DLXS	Field	yes	National Library of Medicine, University of Michigan Museum of Art
	MAIS-Internet	Full-text	yes	72 Dutch archives on Archieven.nl
	MarkLogic	Full XML	yes	University of Chicago, Elsevier, Greenwood Publishing, McGraw-Hill Education, Oxford University Press, Princeton Theological Seminary, University of Toronto Library, Library of Congress
	TEXTML	Full XML	yes	Northwest Digital Archives, Rocky Mountain Online Archive, A2A

cater to. These typical usage scenarios have been copied and pasted from real world e-mail correspondence between archivists and people asking for assistance from the *Nationaal Archief*. In IR, these are also known as *ad hoc* queries.

- *"Do you know whether the 'Ministerie van Verkeer en Waterstaat' has an own archive which includes PTT documents relating to wireless telegraphy?*
- *"Please can you send me the link or the pdf files that contain the Catalogue of Ancient Dutch Maps."*
- *"I am writing a book about foreigners in Beijing from the Boxer Rising in 1900 to the Communist takeover 1949. Jonkheer Frans Beelaerts van Blokland was the Dutch Minister in Peking during the World War One. I am very interested in seeing any papers that you may hold relating to his years in Peking."*
- *"As per our earlier telephone conversation, I would like to get more information about my family. I have been able to trace back some correspondence concerning my Grandfather (–name removed–) under the dossiers released over the Westerling affair. However, I am unable to trace records covering my Grandfather's activities during WWII."*
- *"I am a trying to find documents related to Mexico's independence. Professor –name removed–, from Leiden University, told me that a Dutch Consul (RHL Heidsieck) who served in Mexico in the mid XIX century, donated some of them to the archive of the foreign ministry. Could you please advise me how to proceed. I am willing to visit the archives soon."*
- *"I am researching the Netherland Cyprus Consulate in XIX. century. Later, I want to extend my research. So my research will include the Netherland Consulates in the Ottoman Empire in XIX. century. Therefore, I want to learn whether there are any berats or fermans in your Archives. If it is, how can I read them? I can read Ottoman Turkish."*
- *"I am conducting research on the Jewish refugee camps and deportation camps in Holland from 1938 to 1944. My family came from Ostfriesland, Germany to Rotterdam in December 1938. They lived in the refugee camp 'Koninginnehoofd' at Wilhelminakade No. 74 in Rotterdam until 1940. I have learned many facts about Camp Westerbork. Is there any way to learn more about Camp Koninginnehoofd or about Jewish refugees in Rotterdam at this time?"*

2.4.2 EAD Finding Aids

We have obtained in total 5,934 EAD finding aids from the *Nationaal Archief* (National Archives of the Netherlands or NA), where the statistics are shown in Table 2.2 with the distribution of the length of content in bytes (without XML markup) and of the structure in terms of XML tags.

When we assume that one type-written page is 2 kilobytes in size, then there are 46 EAD finding aids that have more than 250 pages (512,000 bytes) in terms of the *content only*. On average, the content of an EAD finding aid consists of 19 type-written pages, and the median of the content length is between 3-4 pages.

Table 2.2: Statistics of the *Nationaal Archief* 's EAD finding aids.

Feature	Value
Number of Files	5,934
Content Length (bytes)	M=38,605 (Mdn=7,662)
XML Markup (tags)	M=2,322 (Mdn=544)
Correlation Content + Markup	Pearson's $r = .922$, Spearman's $\rho = .855$, Kendall's $\tau = .685$

According to Pitti (1999), detailed archival descriptions average 15-30 pages in length. So on average, the *Nationaal Archief* EAD finding aids can be considered 'detailed.' We show that there is strong positive and significant correlation on a 1% significance level (2-tailed) between content length and XML markup (Pearson's r, Spearman's ρ and Kendall's τ), likely due to the completeness of the markup.

The distribution of the top ten most frequently occurring elements over all finding aids is shown in Table 2.3. We see that `<unittitle>` is the most frequently occurring tag in the *Nationaal Archief* 's set of EAD finding aids, followed up by element `<did>`, and `<unitid>`. These three elements alone make up for more than fifty percent of all EAD structure. These elements are used to describe the lists of files and items in the `<dsc>` nodeset.

Table 2.3: Top 10 distribution of elements ($N = 13,775,835$).

Element	Count	Percent
`<unittitle>`	2,397,086	17.40
`<did>`	2,347,000	17.04
`<unitid>`	2,318,869	16.83
`<unitdate>`	1,595,161	11.58
`<physdesc>`	647,012	4.70
`<p>`	556,576	4.04
`<c03>`	459,269	3.33
`<c04>`	401,438	2.91
`<c05>`	361,138	2.62
`<c02>`	360,561	2.62

2.4.3 Indexing and Storage

Before the indexing and storage, we preprocess the files to make them strictly well-formed XML—which was a prequisite for indexing in an XML database. We do not check for validity, and add minor modifications to the official 2002 EAD standard, of which the most noticeable one is that we convert all XML tags to uppercase. We pre-process the EAD files by adding the physical filename and the title `<titleproper>` as attributes in the root node `<ead>` using XSLT,

then process them again in XML Lint, and then cleaning them up (like making all tags uppercase) in Tidy.[9] Since we deal with mostly Dutch language data, we use the ISO/IEC UTF-8 character encoding.

The system is based on MonetDB with the XQuery front-end Pathfinder (Boncz et al., 2006) and the information retrieval module PF/Tijah (Hiemstra et al., 2006). All of the *Nationaal Archief* 's 5,934 EAD finding aids made available to us are indexed into a single main memory XML database that completely preserves the XML structure and allows for powerful XQuery querying. We index the collection without stopword removal, and use the Dutch snowball stemmer.

2.4.4 Retrieval Model

For the retrieval of individual and any arbritary elements, we employ statistical language models (LM) (Ponte and Croft, 1998), i.e. the probability distribution of all possible term sequences is estimated by applying statistical estimation techniques. The probability of each individual term is calculated using the *maximum likelihood estimate* (mle), which corresponds to the relative frequency of a term t_i in an element e, $P_{mle}(t_i|e) = \frac{\mathrm{tf}_{i,e}}{\sum_t \mathrm{tf}_{t,e}}$ where $\mathrm{tf}_{i,e}$ is the term frequency t_i normalized by the sum of all frequencies in an element e.

We estimate the probability that the element model can generate the given query q. By applying Bayes' theorem, this can be obtained by

$$P(e|q) = \frac{P(q|e) \cdot P(e)}{P(q)} \propto P(q|e) \cdot P(e) \tag{2.1}$$

where $P(q)$ can be ignored for ranking, and the prior $P(e)$ is assumed to be uniform. The query likelihood (or conditional probability) is based on a model that represents an element using a multinomial probability distribution over a vocabulary of terms. For each element, a model on an element is inferred, such that the probability of a term given that model is $p(t|e)$. The model is then used to predict the likelihood that an element could match a particular query q. We make the assumption that each query term can be assumed to be sampled identically and independently from the element model. Applying this assumption, the query likelihood is obtained by multiplying the likelihoods of the individual terms contained in the query:

$$P(q|e) = \prod_{t \in q} P(t|e)^{n(t,q)} \tag{2.2}$$

where $n(t, q)$ is the number of times term t is present in query q.

[9] Retrieved 2011/05/11 from `http://www.xmlsoft.org/xmllint.html` and `http://tidy.sourceforge.net/`.

To deal with zero probabilities because of non-existing terms in case there is sparse data, smoothing techniques are applied. The retrieval model uses Jelinek-Mercer smoothing, which is a mixture model between the element model and the collection as background model, so

$$P(t|e) = (1 - \lambda) \cdot P_{mle}(t|e) + \lambda \cdot P_{mle}(t|C) \tag{2.3}$$

where $P_{mle}(t|C) = \frac{ef_t}{\sum_t ef_t}$, ef_t is the element frequency of query term t in the collection C, and the λ is set to 0.15.

2.4.5 Querying and User Interfaces

We discuss now the three approaches deployed in the README system, which is written in Perl using XHTML, CSS, and JavaScript. The connection with the database server is made using a socket and XML RPC. After processing the query in the system, we post-process the results. An issue in post-processing of XML retrieval results is content overlap due to the hierarchy and nesting of elements. Kazai et al. (2004) explain the overlap problem in content-orientated XML retrieval evaluation. Hammer-Aebi et al. (2006) find that the problem of overlapping elements can be solved in end-user systems at the interface level with a contextual view that groups results by document, instead of using an atomic view of element retrieval with a ranked list. Conversely, for convenience, we remove overlap of content by default in our XML IR systems.

The origin of the finding aid is made clear by showing an icon in front of a result that corresponds to a source. In our case, there is one source namely the *Nationaal Archief* , but the system is also setup to allow for search across institutions. For each retrieval approach, we present user interfaces of the README system with as example the query "bijzondere rechtspleging" (in English: *special justice*) and its retrieved results.

Approach 1: Document Ranking

The XML database is queried using XQuery extended with Narrowed Extended XPath I (NEXI) (Trotman and Sigurbjörnsson, 2004). For document (collection) ranking, we provide the root element `<ead>` (the whole EAD finding aid) as target element, i.e. all descriptions in an archival finding aid in EAD are treated as one element. The corresponding *Whole Fonds* (WF) interface is depicted in Figure 2.5.[10]

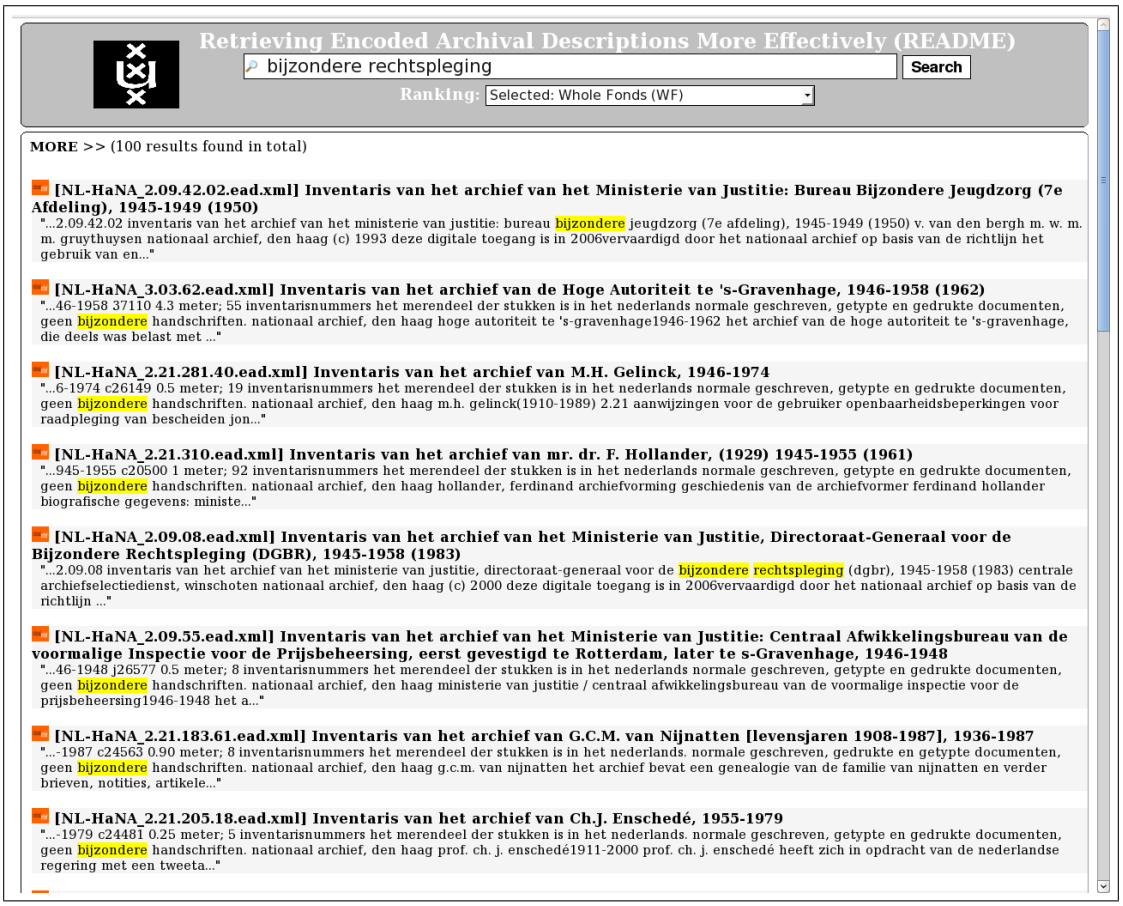

Figure 2.5: Document retrieval with Whole Fonds (WF).

Approach 2: Element Relevance Ranking

For element relevance ranking, see the *Individual Archival Material* (IAM) interface in Figure 2.6, we do not provide a structural hint in the form of a target element, hence any EAD element can be retrieved, including the absolute XPath of an element. Such a path could look like

```
/EAD[1]/ARCHDESC[1]/DSC[2]/C01[4]/C02[8]/DID[1]/HEAD[1]
```

It describes the position of an element in the XML tree hierarchy, starting from the root to the leaf node. The snippet of XQuery code in Figure 2.7 illustrates the procedure element ranking that retrieves N elements stored in `$nodes`.

[10] The term *fonds* is defined by Pearce-Moses (2005) as: "The entire body of records of an organization, family, or individual that have been created and accumulated as the result of an organic process reflecting the functions of the creator." Retrieved 2011/06/24 from `http://www.archivists.org/glossary/term_details.asp?DefinitionKey=756`.

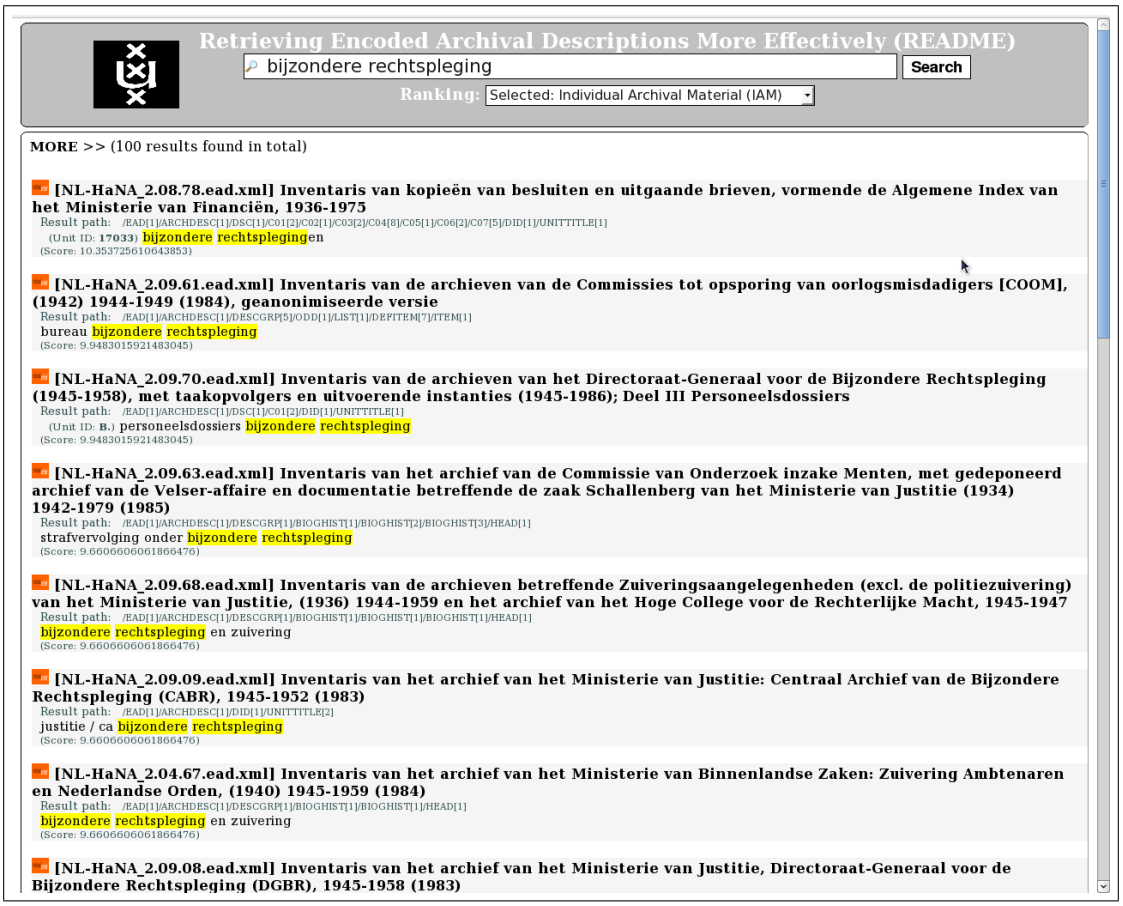

Figure 2.6: Element retrieval with Individual Archival Material (IAM).

```
declare function readme:retrieveAllAMC($query as xs:string, $model as
xs:string, $start as xs:integer, $end as xs:integer) as node()*
  {
     let $options := <TijahOptions ir-model="$model"
            collection-lambda="0.15" rmoverlap="true" prior="no_prior"
            returnNumber="N" />

     let $query_text := tijah:tokenize($query)
     let $query_nexi := concat("//*[about(., ", $query_text, ")]")

     let $qid := tijah:queryall-id($query_nexi, $options)
     let $nodes := tijah:nodes($qid)
```

Figure 2.7: A part of the function readme:retrieveAllAMC. XQuery code that illustrates the initialization of system parameters and the use of NEXI for querying. Here, we search in any nodes using the XPath expression //*, which corresponds to any elements in the document. This piece of code is used for element retrieval.

Approach 3: Aggregation-based Ranking

As Haworth (2001, p.14) notes, archival finding aids have the structure of a tree, and the multi-level description refers to a part-to-whole relationship where de-

FIGURE 2.8: Aggregation-based retrieval with Archival Materials in Context (AMC).

scriptions are related to each other. The deeper one gets in a finding aid, the shorter the descriptions become. The reason is that according to ISAD(G), information on higher levels cannot be repeated at lower levels, thus descriptions at lower levels are dependent on higher level descriptions. Therefore, descriptions must be displayed together to be comprehensible (Haworth, 2001, p.15).

However, XML retrieval systems, like README, return elements. How do these systems deal with descriptions that inherit the concept of higher level descriptions? How deep should a system go? There is a dilemma for these systems. If the lowest level possible element gets returned, the user may be clueless as there is no information contained in the descriptions of a higher component. If a description of a higher component gets returned, the promise of direct access to relevant information cannot be fulfilled as the user still has the burden to browse the finding aid. Kazai et al. (2004) experiment with the 'overlap' problem, and note that an element may be more exhaustive than any of its descendants alone given that it also contains non-relevant information, while these lower level elements may not contain enough relevant information alone (Kazai et al., 2004, p.73).

```
let $result := for $node at $relevance in $nodes
    return
    <result>
       <rel> { $relevance } </rel>
       <num> { (count($node/preceding::*) + 1) } </num>
       <file> { data($node/ancestor-or-self::EAD/@FILE) } </file>
       <title> { data($node/ancestor-or-self::EAD/@TITLE) } </title>
       <headers> {
          for $node_ancestor in $node/preceding::HEAD[1]
          return
             <heading_before>
                {
                $node_ancestor,
                <headingpath>{readme:getPath($node_ancestor)}</headingpath>
                }
             </heading_before>
       } </headers>
       <path> { readme:getPath($node) } </path>
       <id> {
          for $node_ancestor in $node/preceding-sibling::UNITID[1]
          return <id2> { data($node_ancestor) } </id2>
       } </id>
       <id> {
          for $node_ancestor in $node/preceding-sibling::CONTAINER[1]
          return <id2> { data($node_ancestor) } </id2>
       } </id>
       <text> { $node/normalize-space() } </text>
       <score> { tijah:score($qid, $node) } </score>
    </result>

   let $total := count($result)
   let $tmpRes := <results total="{$total}">  {
      for $result2 in subsequence($result, $start, $end)
          for $res in distinct-values($result2/file)
          let $cs-group := $result2[file = $res]
          let $file := string($cs-group/file)
          let $num := number($cs-group/num)
          order by $file, $num
          return <out id="{$res}">{ $cs-group }</out>
      } </results>
    return
       for $a at $b in $tmpRes
          for $r in distinct-values($a//file)
          let $cs-group2 := $a[//file = $r]
             for $xs at $relevance2 in $cs-group2//out[@id = $r]
             let $score := sum ( for $x in $cs-group2//out[@id = $r]
             return $x//score)
             where $relevance2 <= 8
             order by $score descending
             return <results total="{$total}">  { $xs } <br /> </results>
 };
```

Figure 2.9: Part 2 of function readme:retrieveAllAMC. XQuery code that illustrates the retrieval of elements according to relevance, grouping of results by file name, and re-ordering of the results given the original document hierarchy.

To deal with the problem, we introduce an aggregation-based approach. This approach goes a step further than the standard element relevance ranking, so we introduce the *Archival Material in Context* (AMC) interface in Figure 2.8, and see the XQuery code in Figure 2.9. It takes relevance `<rel>` into account.

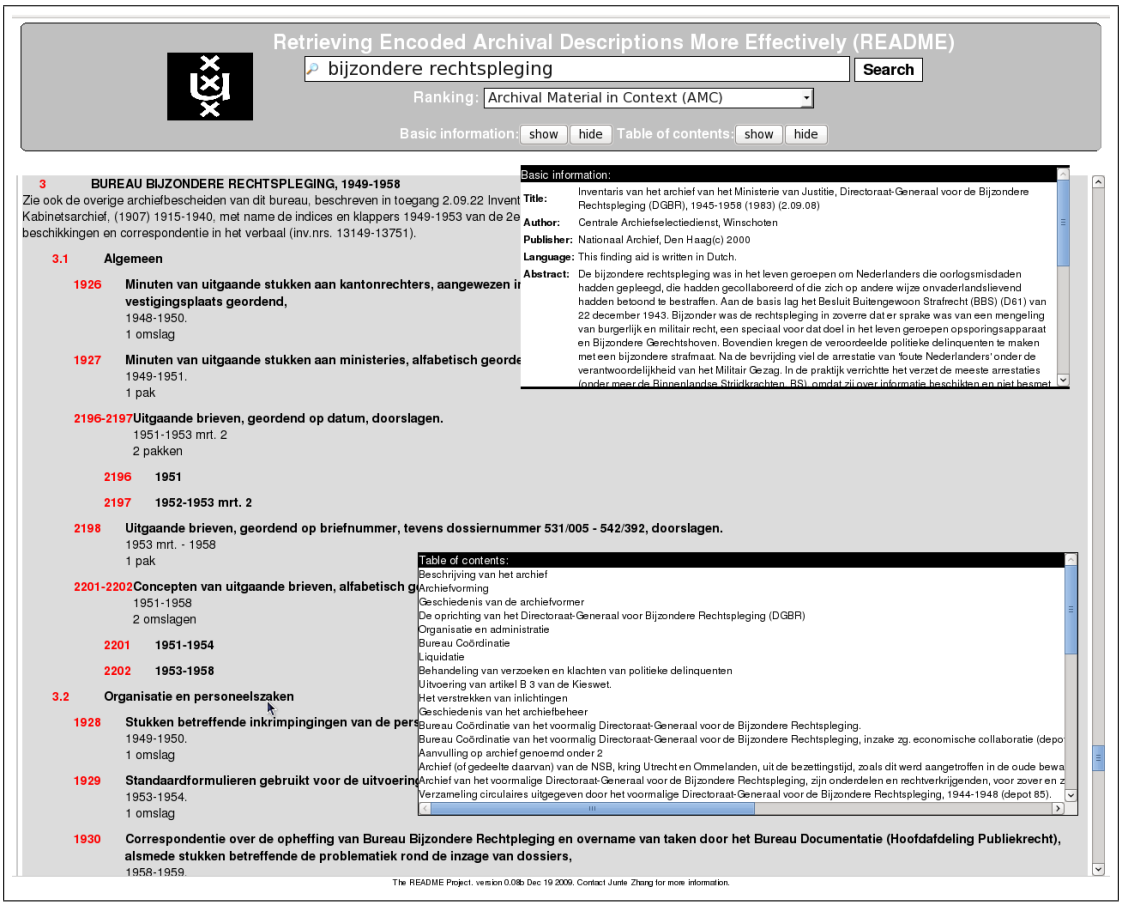

FIGURE 2.10: Deeplinking to the Result Display from the result list with a Basic Information (BI) pane and a Table of Contents. Both the BI and ToC can be dragged, and turned on or off. When a user clicks on an element in the hit list of IAM or AMC, the user gets linked directly to the element in the result display, which the selected element on top in focus.

Any and arbitrary elements can be retrieved. We rank the elements by taking the sum of the top N retrieved element scores e of a file f, so $\text{score}_f = \sum_{e \in f}^{N}$, and rank the files (and its elements) with score_f accordingly, where in our system we set N to 8, but it can be made dynamic by allowing users to move beyond that threshold. The retrieved elements are returned in original order as in the XML file, by computing the distance of the retrieved element to the root node by counting all preceding elements in `<num>`. We group the retrieved elements by its EAD finding aid `<file>`. The aggregation-based approach optimally utilizes the context of the finding aids by aligning the archival principles of provenance and original order of the descriptions within a finding aid (see Chapter 4) to an archival search system using XML IR techniques. It preserves the context to support access.[11]

[11] The aim of respecting the *archival context* is to maintain the quality of information that can

2.4.6 Result Delivery

The hitlist is connected to the Result Display with HTTP parameters using CGI: the query, XPath, source, and file name are always stored in the URL for persistency and to facilitate the analysis of the search logs. The system can deep-link (with the element and aggregation approaches) by rendering HTML anchors for each element using its (unique) XPath as anchor identifier. We deliver a result by physically linking a result to its file, and render its result display with the 'Basic Information' (BI)—e.g. core information of the finding aid, such as the title, creator and summary—and the 'Table of Contents' (ToC) using the SAXON XSLT processor[12]—this is faster than retrieving everything again from the index. There is minimal transformation from the original XML file, because EAD is as much document-centric (directly view-able by users in a browser) as it is data-centric. We use the Yahoo! User Interface Library (YUI)[13] to make the BI and ToC dynamic and enable enhanced interaction, see Figure 2.10. The BI and ToC can be dragged and hidden—making them extra non-obtrusive tools to locate information within the retrieved file.

2.5 Conclusion

After a technical embedding of the background where we explained EAD and XML, we formally introduced our system that employs XML retrieval techniques to gain more focused archival access, effectively exploiting the structure to search and find valuable information in context. A result is the implementation of our vertical (domain-specific) XML IR system driven by archival finding aids: *README*—an online archival information system that is able to retrieve the representation information of the archives by exploiting this representation, specifically the granularity and structure in XML.

Access to archival finding aids in EAD is a two-tier approach. First, the EAD finding aid has to be found. Second, relevant descriptions within the found EAD finding aid have to be returned. We showed that XML IR can be fruitfully applied on archival finding aids in EAD with both tiers. We presented three systems that each provide a different level of detail using the two tiers (from collection to item level) for focused access with archival finding aids in EAD. We also translated archival description principles aimed to preserve the archival

be reused by ensuring its availability, readability, completeness, relevance, representativeness, topicality, authenticity, and reliability (Thomassen, 2001, p.382). Archival research is the study of this context (Thomassen, 2001, p.384). Archival research goes beyond 'information,' and focuses on the interpretation of the 'historical evidence' that consists of these relations (Gilliland-Swetland, 2005, p.236). This rounds off the interrelated argument: when archivists manage their archives by respecting the archival context, they enable their access and use, and make archival research possible, that consists of the study of that context. [12] Retrieved 2011/01/11 from `http://saxon.sourceforge.net/`. [13] Retrieved 2011/01/11 from `http://developer.yahoo.com/yui/`.

context, and projected on EAD finding aids, to an information system design with our aggregation-based approach. Fachry et al. (2008) conduct a user study with the systems, and show broad support for this approach.

This dissertation's research focuses on measuring and evaluation access to archival finding aids in EAD, and it does not aim to find out how to design the most usable interface for archival finding aids in EAD, which would be an area too broad to explore. For example, Allen (2005) describes information visualization approaches from the hypertext and visualization research communities for archival access. The retrieved information as expressed in a query should be juxtaposed with proper interfaces. How to make the information and related services *visible* to people with services and functions constitute what access is. The access approach of README is also archival with influence of the archivists who created the EAD finding aids (Hedstrom, 2002), because we aligned archival concepts and principles with the system. We test this influence on access in Chapter 4. More types of system based on README with different ranking algorithms are also possible, but the baseline has been established with the aggregation-based retrieval approach. The question arises: how well does this system work? This question will further be researched in the remaining chapters.