<XML/> AND ELECTRONIC RECORDKEEPING

Filip Boudrez

Antwerpen 2002 version 1.1

0. Table of contents

1. Introduction

XML, or eXtensible Markup Language, has really taken off since it was formally established. It owes its widespread and ever growing use to the fact that it can be used differently in a variety of IT applications[1]. After all, XML is more than just a file format. Within a fairly short space of time a whole field of XML technology has grown up around this markup language.

One possible area of application for XML is digital archiving. XML offers plenty of prospects for archiving digital documents, and, as a consequence, expectations are high in the archiving community. Despite its relative youth XML is already being recommended as standard in a great many guidelines and procedures.

In this article we will investigate how XML can be put to use for the purposes of archiving. We will give a general description of XML and its related technologies before going on to discuss the different ways in which archival institutions can use XML to preserve electronic records. We will also cover the aspect of practical implementation, making use of practical examples where appropriate[2].

---

[1] Examples of IT applications include: business-to-business, content providers, web content management, wireless applications, etc. XML is used in these for data exchange between different systems, dynamic on-the-fly transformations, information integration, and data distribution over different formats. For a list of XML applications see http://xml.coverpages.org/xmlApplications.html

[2] The meaning of technical terms not explained in this article can be found in the glossary on the DAVID web site (http://www.antwerpen.be/david → publications). The DAVID web site also provides on line examples (→ cases).

2. The need for XML: XML versus HTML and SGML

XML is a markup language. A markup language is a technology that uses tags to structure a document. SGML and HTML, two markup languages to which XML is related, proved inappropriate for a simple and user-friendly exchange of structured documents via the web, and so the new markup language was developed[3].

XML was developed as a joint project between the industrial and academic worlds. The computer industry (Sun Microsystems, Hewlett-Packard, Microsoft, Netscape, Adobe, Fuji Xerox) and SGML producers (ArborText, Inso, SoftQuad, Grif, Isogen, Texcel) were involved from the very start of the development phase. Later, IBM, Oracle and Omnimark came on board. The academic input came from the Text Encoding Initiative (TEI), NCSA and James Clark. The developmental work commenced in 1996. Two years later XML became established as a *W3C* Recommendation.

The XML specification sets a standard for adding markup to documents. The XML specification does not contain fixed set tags with specific meanings, but uses rules to compose an XML document. XML is therefore a meta-language, which enables the user to define the tags, their attributes and their mutual relations. In this sense XML differs fundamentally from HTML. In the HTML specification the tags, their attributes, and their meanings are fixed. In HTML the user cannot create his own tags or give these tags a meaning, and so cannot give a user-defined structure to a document when using HTML tags. However, with XML the user composes his own lexicon, as it were. Another basic difference is the function of the tags. HTML tags are geared towards visual presentation. They say nothing about the content or structure of the computer data. In other words, an HTML document has no real structure based on the data itself. As opposed to XML, HTML is not data or object oriented. Among other things this means that it is not possible to carry out structured or semantic searches of HTML source files (e.g. list all publications by author x sorted according to date of publication), which is a considerable disadvantage when it comes to accurate searches of the web. Therefore, HTML is not suited to information exchange or automated data management (search, manipulation, integration). Apart from tags and attributes becoming version-dependent this also means that content and layout cannot be kept separate when using HTML.

XML is just as distinct from SGML[4]. What both markup languages have in common is that they can be used to store and exchange data, and that the user can define his own tags and structure the documents himself; but SGML documents are too complicated to exchange data over the web. SGML requires powerful systems and cannot be easily implemented in a web browser. Therefore SGML tends to be used in large documentation projects. XML is a light version of SGML and is perhaps best considered as an application profile of SGML[5]. XML is simpler than SGML. XML has taken from SGML the possibility of identifying information, extensibility, and validation. As opposed to SGML, XML does not require the presence of a DTD. An SGML application is capable of reading an XML document, but the opposite does not apply necessarily.

The creation of XML should therefore be viewed against the backdrop of information exchange on the Internet. XML was initially intended to replace HTML and gained a reputation as the new "ASCII of the web". XML already plays an important role on the web, but, in the meantime, it is also being used for other purposes in other applications. XML is all about digital information in general.

3. XML Documents

3.1 Composition

XML is a markup language based on text characters, which can be used to establish the structure and meaning of documents. An XML document consists of no more than three parts: the header, the body and the epilog.

[3] For a more detailed discussion of HTML and SGML, see: *Standards for Digital Records* (http://www.antwerpen.be/david under "cases" or "publications").

[4] For a detailed comparison, see: J. CLARK¸ *Comparison of SGML and XML*, 1997 (http://www.w3.org/TR/NOTE-sgml-xml.html).

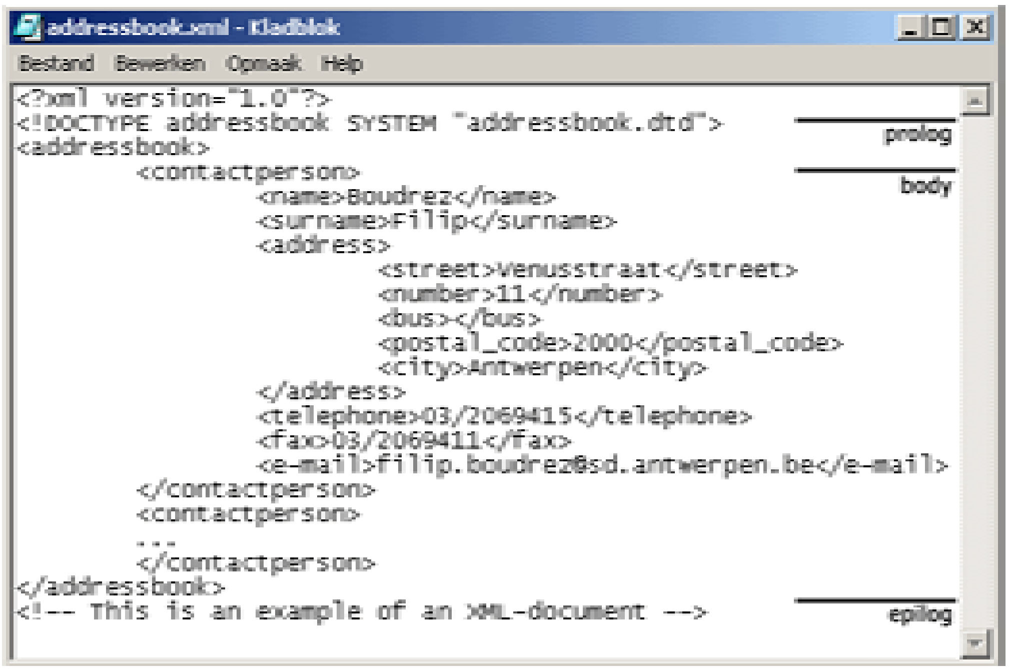
[5] This is amply illustrated by the length of the specification. The SGML specification is more than 600 pages long, whereas that of XML is barely 26 pages long.

An XML file starts with a header or prolog, which, among other things, contains the declaration. The declaration contains at least the version number of the XML specification used when the document was created. Optionally, the header can be used to specify the character table and establish whether or not the file is dependent on external sources[6]. The declaration can also be enlarged to include a reference to a DTD, an XML Schema, a style sheet or the namespaces applied (see 4. The XML Family).

The XML body starts with the root element and is closed by the end tag for that root element. The body consists of the elements from which the XML document is constructed. In a sense, the elements (or objects) are comparable with the data fields of a database. The length of an element can vary from one character to an entire book. The elements of an XML file are modelled hierarchically like a tree: one element can be part of another element (the parent) or consist of other elements (the childs)[7]. Elements with the same parent are known as sibling or sister elements. The method of arranging the elements is called *nesting*. The element at the highest level is known as the root element. The XML root element is not part of another element but contains the elements from which the XML file is constructed. Every element can be given attributes that provide more information (e.g. <personaldetails public="no">). The elements are always demarcated by means of one start tag and one end tag.

The epilog is optional and can contain processing instructions, white space and comments.

Illustration 1: structure of an XML document



```
<?xml version="1.0"?>
<!DOCTYPE addressbook SYSTEM "addressbook.dtd">
<addressbook>
        <contactperson>
                <name>Boudrez</name>
                <surname>Filip</surname>
                <address>
                        <street>Venusstraat</street>
                        <number>11</number>
                        <bus></bus>
                        <postal_code>2000</postal_code>
                        <city>Antwerpen</city>
                </address>
                <telephone>03/2069415</telephone>
                <fax>03/2069411</fax>
                <e-mail>filip.boudrez@sd.antwerpen.be</e-mail>
        </contactperson>
        <contactperson>
        ...
        </contactperson>
</addressbook>
<!-- This is an example of an XML-document -->
```

The markup and content characters of an XML document consist of text characters. XML supports Unicode with the exception of some control characters. The text characters are stored in a linear way. This does not mean that an XML document cannot contain binary file data. The XML document cannot itself contain the binary bitstream because this uses invalid characters. There are two methods of storing binary data in XML: either they are stored as unparsed entities (CDATA), or the binary characters are transposed into text characters. The latter is the most frequent solution. Macintosh platforms use BinHex-encoding for this, IBM

[6] Encoding: UTF-8, UTF-16 and ISO-8859-1, ISO-10646-UCS-2 or ISO-10646-UCS-4. UTF-8 is the predefined encoding, i.e. applications assume UTF-8 when no coding is specified.
standalone="yes": document contains all entity declarations; standalone="no": document needs an external DTD.

[7] The content of elements can be mixed, simple, element and empty.
(mixed content: contains text and other elements; simple: contains text only; element content: contains other elements only; empty: no content)

compatible computers use generally Base64 (rfc-1341, rfc-2045[8]). With BinHex- and Base64 encoding the binary data are converted to text characters using an established encodingscheme. The Base64 standard uses a 64 ASCII character subset (A-Z a-z 0-9 +/) to represent binary data and "=" for padding. The application itself converts binary data to Base64. In this way the bitstream of a TIFF file can be converted to text characters and stored in an XML file. To reconstruct the TIFF file the text characters are reconverted to binary data[9]. Recording the data of a binary file in an XML document has interesting consequences for data exchange and/or encapsulation.

All data in an XML document is pure text. Strictly speaking an XML document contains no other data types, such as dates and integers. Using text characters for markup and content means that an XML document can always be opened in a normal text editor and that the characters will always be legible. XML documents are therefore referred to as human-readable.

3.2 Document modelling

With the help of hardware and software we can save our knowledge of reality to a medium in the form of bitstreams. These bitstreams are in themselves nothing more than computer data with no real meaning or context of their own. However, electronic recordkeeping is not about storing bitstreams, but about transferring knowledge digitally over time in such a way that the documents can still be interpreted. In the face of technological obsolescence we must ensure that the digital representation of knowledge is infrastructure independent. We can achieve this by means of data or document modelling[10]. This is a calibrated means of storing knowledge in the form of computer data in a way that is open to human interpretation. XML offers us the possibility of modelling documents.

Created in the database environment, modelling is a method of linking computer data to an established knowledge model. The cornerstones of the knowledge model are "defined" and "related" computer data. By adding "identifying" and "characterising" information to data, we can clarify its meaning, but this is still not enough to extract knowledge. For this, the information still has to be structured. Only by establishing relations between the information can we obtain knowledge. In other words, knowledge is structured information. Modelling is essential not only as means of reconstructing knowledge in the future, but as a means of avoiding obscurities and misunderstandings when information is exchanged[11].

The process of modelling data for documents can be broken down into several steps: 1) naming the data, 2) identifying and defining the data by classifying them, 3) establishing the relations between the data, and 4) determining the properties of the data. This process results in a knowledge model, into which the computer data are fitted when creating documents. Document design therefore becomes a translation of that knowledge model into a set of rules to be followed on creation.

Essential to the XML document modelling mechanism is the addition of semantic tags to data (bitstreams) and the nesting of elements. The data objects correspond with the content of the elements from which an XML document is constructed. By allocating semantic tags to the elements we can identify the logical components of a document and give them meaning. For example, the data string "01012000" could mean anything. When <startdate_davidproject> tags are added this series of figures acquires meaning. The tags can be seen as attributes and they contain information on the data, through which interpretation is now possible. The nesting of the elements reflects the relations between the document's information components. The tags can be given attributes. The attributes provide extra information on the content of the tag or on the tag itself. The choice of tags and attributes is entirely free and so the user can adapt them to suit his document requirements.

---

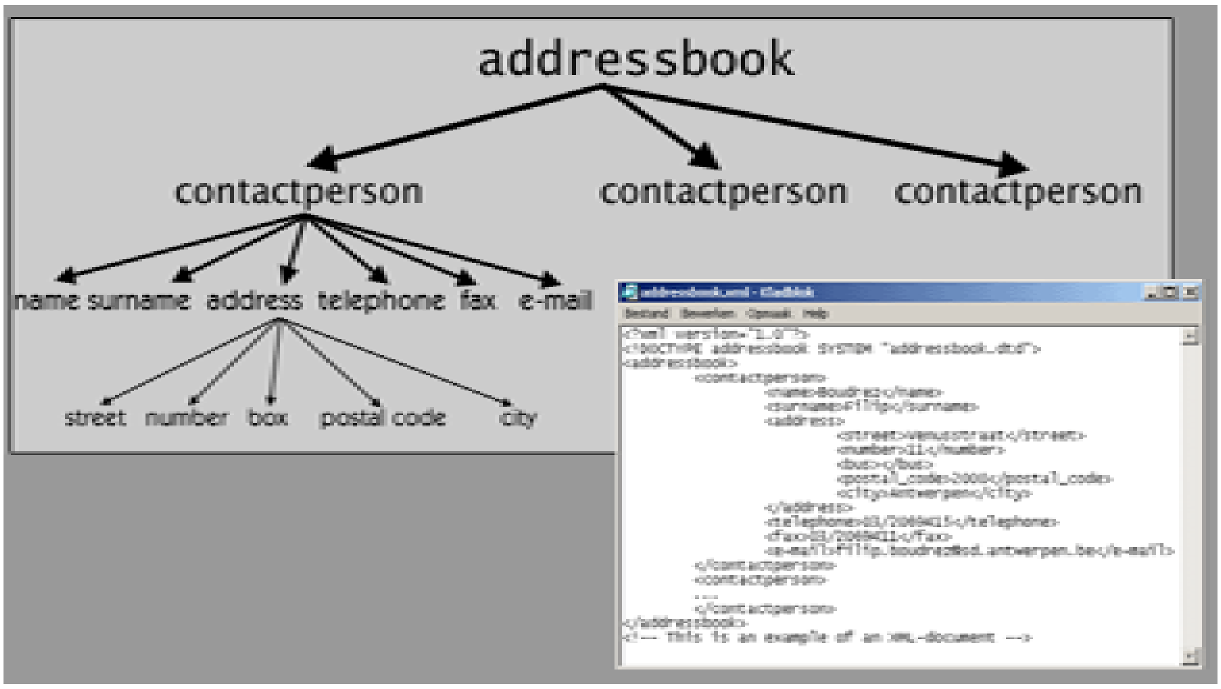
[8] Groups of 3 binary octets are converted to four text characters, with each text character presenting 6 bits of the original 24 bits. According to the MIME specification the length of the line is confined to 76 characters, but this does not apply in an XML document.

[9] This does not mean that, for the purposes of archiving, it would be a good option to store any binary format in an XML file. An application will always be needed to read the binary file.

[10] In IT the term "information modelling" is also used.

[11] R.W. MOORE, *The preservation of data, information and knowledge*, 2001.

Illustration 2: hierarchical tree structure in an XML document



```
<?xml version="1.0"?>
<!DOCTYPE addressbook SYSTEM "addressbook.dtd">
<addressbook>
        <contactperson>
                <name>Boudrez</name>
                <surname>Filip</surname>
                <address>
                        <street>Venusstraat</street>
                        <number>11</number>
                        <bus></bus>
                        <postal_code>2000</postal_code>
                        <city>Antwerpen</city>
                </address>
                <telephone>03/2069415</telephone>
                <fax>03/2069411</fax>
                <e-mail>filip.boudrez@od.antwerpen.be</e-mail>
        </contactperson>
        <contactperson>
        ....
        </contactperson>
</addressbook>
<!-- This is an example of an XML-document -->
```

The addition of tags (tagging) is a very important step in modelling an XML document. In document modelling the function of tagging goes a step further than simple annotation, through which information is added to the computer data and it acquires meaning. Elements can be related to a predetermined knowledge model by means of nesting or structuring. In XML this results in hierarchically structured information. Both steps provide a basis for using tags in an XML document. And this doesn't prevent XML tags from offering many other advantages:

- distinguishing the various atoms of the file as unique delimiters
- identifying or describing the information they envelop. The tags label, as it were, the data fields of a document and provide information on meaning, and even data type. When an XML-document contains also semantic contextual information besides its own content, we don't depend as much on external sources or documentation for the meaning of the data.
- adding semantics to the computer data as well as their attributes. The tags define the semantic context of the components of a dataset.
- allowing nesting of the document components and reflecting the structure and the relations
- enabling powerful and structured searches, resulting in more accurate search results
- enabling sorting actions
- ensuring that computers can process and interpret XML documents easily. XML applications determine how certain tags and their content will be processed
- functioning as nodes (e.g. addressbook/contactperson/address/street) in the XML structure and giving access to data for consultation and/or manipulation on the basis of scripting against the nodes
- acting as primary tools for understanding and managing documents. Documents need not be given specific headers or application-specific characters.

Document modelling yields a number of interesting advantages with a view to electronic recordkeeping. In XML the document and the organisation of the data are central, not the application that processes them. With XML the means of data storage is theoretically determined by the nature and requirements of the data, the record type, or the document model itself. In a normal desktop environment it is mainly the applications that determine the means of data storage.

One of the most important tasks of an archivist in a digital environment, is establishing records in an interpretable way and transferring them in time. Writing down information in the form of an electronic record is always influenced by the perception of the author and by the (implicite) documentmodel that he is using. This can cause problems when exchanging information and usage by others. The 'receiver' of electronic records is not just another person then the author, the archivist has to assume that the 'receiver' is living in another time and society and therefore is re-using the electronic records whithin his own context. Whit XML there is the possibility to record the documentmodel explicitly in a DTD or XML scheme (see further on). The DTD or XML scheme is the translation of the abstract data- or documentmodel. XML-recordkeeping of electronic documents offers the archivist the opportunity to record information and knowledge in a durable and re-usable way. XML is said to be a standard for describing data for a good reason[12].

The application of a document model on creation, combined perhaps with a DTD or XML Schema, provides options in the area of quality control and care. Thus the user can be steered into presenting a fixed structure for certain types of electronic records. It also gives the archivist an opportunity to guide the creation of records. Document modelling also plays an important role in archiving the same type of records from different platforms or information systems. In this way we can establish and apply one document model for each type of electronic record when storing documents in the digital archives, to ensure uniformity. In so doing we can avoid having different document models for the same type ofrecord. One practical application, for example, is the use of a single document model to follow when archiving e-mail, regardless of the application in which this e-mail was created or received.

By using a document model we can store more than just XML document data. In addition to content an XML document contains structure in explicit form. This provides an important difference as regards comma separated text files, for example. In comma separated text files separated by commas relations cannot be established by nesting. The third component of a traditional document, i.e. presentation, is always saved in a separate style sheet[13] (see: 3. The XML Family). Though the layout data are not stored in the XML document, this offers the advantage that XML documents are not technology-dependent. Finally, the nesting of semantic tags ensures that the XML document itself contains the information and knowledge needed to understand the electronic records afterwards (contextual information).

3.3 DTD and XML Schema

XML syntax enables the user to develop a document model from the elements, attributes and structure he has chosen himself. This document model is actually a further refinement of the XML specification, in which he composes his own lexicon for a document (or document type). The document model contains extra rules as well as the general XML syntax rules to be followed when creating the document. The model for an XML document can be formally set out in a Document Type Definition (DTD) or XML Schema. The DTD and/or XML Schema used to map out the document model can be communicated to the receiving computer or user, and used as a control mechanism.

XML specification 1.0 derived the option of composing DTDs from SGML. Therefore DTDs are strongly document-oriented and not even created using XML syntax. In a DTD the elements are defined along with their attributes, composition, relation and frequency. This essentially boils down to the structure and nesting of the document. With a DTD we can only check that the tags and nesting are correct. To some extent a DTD can define and verify the frequency of the elements.

Compared with DTDs, XML Schemas are not geared solely towards documents, but offer a number of possibilities derived from the world of databases. As opposed to DTDs, XML Schemas do offer the possibility of defining the data type, field lengths, and exact minimum and maximum element frequencies. With XML Schemas restrictions can be placed on the input options accepted, thereby offering some form of quality control on creation. The XML Schema is based on XML syntax and supports namespaces (see 4. The XML Family).

The function of a DTD or XML Schema is comparable to that of a database data schema. The DTD or XML Schema contains a definition of all valid documents or sets out the restrictions, pre-defined values, etc.

[12] For instance. M.J. HUDSON, *XML @ work*, in: *EAI Journal*, april 2001, p. 36.

[13] It is also possible to attribute semantics to SML data by means of a style sheet, but the XML document will then lose a little of its independence and its self-descriptive properties will not be used to the full.

Furthermore, a DTD or XLM Schema also documents the XML document. It provides explicit information on the meaning, use, and function of the elements so that author and reader understand them in the same way.

A DTD or XML Schema communicates metadata on the document to the parser[14]. This meta information contains the order and nesting of the tags, the attribute values, the data types and the default values of the elements, the names of external files to which reference is made, and the entities in the file.

Composing a DTD or XML Schema is essentially the same as composing the document model or prototype. By adding successive tags to the record the latter is, at it were, modelled on the DTD or XML Schema, and the document model is applied.

Table 1: DTD compared with XML Schema.

	DTD	*XML Schema*
GENERAL		
Syntax	Extended Backus Naur Form (EBNF)	XML
Support of namespaces	No	Yes
Extensible	No	Yes
Descending	No	Yes
Orientation	Documents	Documents - data object-oriented
Storage place vis-à-vis XML document	Internal/external	Internal/external
VALIDATION FUNCTIONS		
Record structure	Yes	Yes
Correct tags	Yes	Yes
Nesting tags	Yes	Yes
Data types	No	Yes
Field lengths	No	Yes
Frequency of elements	limited	Yes

DTD for addressbook.xml

```
<!ELEMENT addressbook (contactperson*)>
<!ELEMENT contactperson (name, surname*, address*, telephone*, fax*, e-mail*)>
        <!ELEMENT name (#PCDATA)>
        <!ELEMENT surname (#PCDATA)>
        <!ELEMENT address (street*, number*, PO box*, postal_code*, town*)>
                <!ELEMENT street (#PCDATA)>
                <!ELEMENT number (#PCDATA)>
                <!ELEMENT box (#PCDATA)>
                <!ELEMENT postal_code (#PCDATA)>
                <!ELEMENT town (#PCDATA)>
        <!ELEMENT telephone (#PCDATA)>
        <!ELEMENT fax (#PCDATA)>
        <!ELEMENT e-mail (#PCDATA)>
```

XML Schema for addressbook.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="addressbook">
    <xs:complexType>
      <xs:sequence>
```

[14] parsing: analysing the structure of an XML document

```
      <xs:element name="contactperson" type="contactpersonType" minOccurs="0"
      maxOccurs="unbounded"></xs:element>
    </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:complexType name="contactpersonType">
  <xs:sequence>
    <xs:element name="name" ref="name" minOccurs="1" maxOccurs="1"></xs:element>
    <xs:element name="surname" ref="surname" minOccurs="0" maxOccurs="5"></xs:element>
    <xs:element name="address" type="addressType" minOccurs="0" maxOccurs="2"></xs:element>
    <xs:element name="telephone" ref="telephone" minOccurs="0" maxOccurs="3"></xs:element>
    <xs:element name="fax" ref="fax" minOccurs="0" maxOccurs="2"></xs:element>
    <xs:element name="e-mail" ref="e-mail" minOccurs="0" maxOccurs="2"></xs:element>
  </xs:sequence>
 </xs:complexType>
 <xs:complexType name="addressType">
  <xs:sequence>
    <xs:element name="street" ref="street" minOccurs="0" maxOccurs="1"></xs:element>
    <xs:element name="number" ref="number" minOccurs="0" maxOccurs="1"></xs:element>
    <xs:element name="box" ref="box" minOccurs="0" maxOccurs="1"></xs:element>
    <xs:element name="postal_code" ref="postal_code" minOccurs="0"
     maxOccurs="1"></xs:element>
    <xs:element name="town" ref="town" minOccurs="0" maxOccurs="1"></xs:element>
  </xs:sequence>
 </xs:complexType>
 <xs:element name="box" type="xs:string"></xs:element>
 <xs:element name="e-mail" type="xs:string"></xs:element>
 <xs:element name="fax" type="xs:string"></xs:element>
 <xs:element name="town" type="xs:string"></xs:element>
 <xs:element name="name" type="xs:string"></xs:element>
 <xs:element name="number" type="xs:string"></xs:element>
 <xs:element name="postal_code" type="xs:integer"></xs:element>
 <xs:element name="street" type="xs:string"></xs:element>
 <xs:element name="telephone" type="xs:string"></xs:element>
 <xs:element name="surname" type="xs:string"></xs:element>
</xs:schema>
```

3.4 Main rules of the XML syntax

The main XML rules are[15]::

- every XML file must contain one basic XML element (the root element or the document element). All other XML document elements are nested in this basic element. An XML document can only contain one root element.
- XML tags enclose elements. The tags start with the character "<" and end with ">". Every element has a start tag and end tag. An element tag consists of <elementname>. The element name is identical in the start tag and the end tag. The end tag differs from the start tag through the addition of a backslash before the element name (e.g. <elementname></elementname>).
    - No restriction to the length of tag names
    - Valid characters for element names:
        - All letters, all figures and the following punctuation marks: colon, underline, hyphen and full stop.

[15] For good technical instructions to XML: http://www.w3c.org; http://xml.coverpages.org/; http://xml.com; E.L. MORGAN, *Getting started with XML: a manual and workshop*, 2003; http://www.w3schools.com/

  - Restrictions: spaces are not permitted. Tag names can only start with a character, underline or colon (not a figure, punctuation mark or "xml").
- tagelements always have to be closed in the correct order by endtags. The elementnesting has to be correct. There is no restriction to the tag levels or structural depth of the document.
- several text characters are reserved for markup purposes can cannot be used as characters to mark content. These characters are replaced by entities in the XML file content.

Text character	*Entity*
<	&lt;
>	&gt;
&	&amp;
"	&quot;
'	&apos;

- XML is case-sensitive: respect the difference between upper and lower case letters.
- the attribute values appear between standard inverted commas. The attributs ID and IDREF can be used for links.

3.5 Validation

XML specification 1.0 states that every XML document should follow the XML syntax rules given in the Recommendation. Any document that complies with this is *well formed*. Therefore every XML document is *well formed* by definition. If a document is inconsistent with one of the rules of syntax it is not an XML document.

The XML specification also distinguishes a second category of XML document, i.e. the *valid* XML document. Not only are these documents *well formed*, but they are composed in line with a given DTD or XML Schema. Not only do *valid* XML documents meet the general XML rules of syntax, but they conform with the document model defined in a DTD or XML Schema.

A *validating* XML processor or parser is used to validate XML documents. XML applications do not work on documents directly but work via a parser, which gives them access to content and structure. A parser deconstructs an XML document and whether or not this document complies with a given document model. Validation on the basis of an XML Schema is more powerful than on the basis of a DTD. A *non-validating* parser only checks the XML document for *well formedness*.

3.6 Properties

The list of XML document properties is fairly extensive. Below we give a list of properties that make XML suited to digital archiving.

XML:

... *is the result of a standardising process and has the status of public defacto standard*[16]

[16] In 1997 the DLM forum set out three levels of standards: 1. defacto standards, 2. publicly available specification, 3. legal standards. XML is classified in the second group (*Guidelines on best practices for using electronic information*, page 50).

XML was published as a Recommendation of the *W3C* (*World Wide Web Consortium*) on 10 February 1998. The second edition of the XML 1.0 specification was published on 6 October 2000. The second edition is a coordinated version of the first edition plus the errata list. Version 1.1 is currently a working draft (version of 25 April 2002). The *W3C* is a consortium that deals with standardisations for the web. The *W3C* is not an official standardisation authority. The *W3C* Recommendations have been published and are freely available at no charge. A set of specifications will run through a complete process before becoming a *W3C* Recommendation: working draft → last call working draft → candidate recommendation → proposed recommendation → recommendation. These steps are linked with the recommendations, comments and approvals of other *W3C* study groups, the *W3C* team, members, and the public. Any substantial change to a Recommendation must run through the entire Recommendation creation process (public comment, including recommendations and approvals)[17]. One of the aims when designing XML was compatibility with SGML. SGML became an official ISO standard in 1986 (ISO-8879). SGML applications must be capable of reading XML files.

➥... *is extensible*

XML is a meta language that enables every user to create his own XML lexicon. Everyone can make his own markup and composes personaly the language with which he describes the data.

➥... *is platform independent*

XML files can be exchanged between different computer platforms.

➥... *is software independent*

XML files are not linked to one particular computer application or version when files are created, opened or edited.

➥... *is free of patent rights*

XML is not the property of one particular manufacturer. No company or institution is entitled to call itself the owner of XML or to change the XML specification on its own initiative.

➥... *is not as version-dependant as other markup languages*

The user defines the tags and attributes of his choosing. Therefore tags and attributes do not become obsolete when versions change.

➥... *is transformable*

XML data can be shown in another structure and in any output format.

It is not strictly necessary to follow the structure of stored data when generating output. The transformation requires a style sheet (HTML: XSL(T); PDF: XSL:FO) or JAVA. Other actions can be performed during transformation: movement of the text elements, addition or deletion of text, sorting according to one element, numbering of items (e.g. headers), performance of calculations, etc.

The content of XML files can be published in various (future) file formats. At present XML files are mainly used to create (X)HTML and PDF documents, but it is possible to convert XML content to future, as yet unknown technologies.

The same XML source file can be converted to several output formats. XML is therefore a storage format from which a variety of renditions can be made.

[17] http://www.w3c.org/About the World Wide Web Consortium (W3C).htm ; http://www.w3c.org/5 Technical Reports.htm ; http://www.w3c.org/5 Technical Reports.htm

... *consists purely of text characters (Unicode)*

XML is a format that encodes all markup and content data as text characters. XML files are not binary files requiring a particular software application to interpret the binary data and reconstruct their content. As a result XML files can be opened in any application capable of reading text characters. XML files are therefore simple, easy to exchange and digitally durable. In this way XML meets the requirement of having the lowest possible level of platform and software dependence. The only dependence is a correct interpretation of the bits to Unicode characters.

XML uses Unicode. XML documents can contain characters in all languages supported by Unicode. This means that XML can be described as *universal*.

Binary files can be stored in an XML file but the binary data must first be encoded as text characters. The encoding must be documented.

Textual encoding is an interesting factor in data exchange and preservation. Computers can easily process XML documents, and, moreover, people can understand them easily. Keeping in mind long-term preservation, full textual encoding is safer than binary encoding. One corrupted bit can lead to an irreparable and illegible record with binary files. This is not the case with textual encoding.

... *separates content and layout*

XML allows separate archiving of content and layout. The content is stored in the XML file, and the layout of the record can be set out in a CSS or XSL(T) file. Separating content from layout offers advantages with regard to reusable information and large-scale information management. Thanks to media independent storage and the coupling of XML data to presentation late in the production process, XML documents can be transformed to any file format for output (XML, HTML, VoiceXML, WML, PDF, TeX, RTF, etc.). This media independent publishing is possible in "real time". The same XML source document can be used in combination with different style sheets. The lack of hard coded layout information in the data facilitates easy data exchange.

... *is self-descriptive*

XML documents are self-descriptive and so people and computers find them easy to interpret.

XML tags identify and label the components or fields from which the information is constructed. The tags indicate the start and end of each field, signal the information data type, and explain the meaning. The tag attributes can give a further description of the document atoms. The nested structure reflects the relationship between the different elements. XML also enables the encapsulation of metadata or contextual information thereby making XML files self-descriptive. Hence no external documentation or sources are needed in respect of semantics, interpretation or visualisation. An XML file can function as an envelope or container that holds its own description along with the electronic record. This is influenced by the notion of object-orientation, where the archived computer file contains the record as well as its metadata. Therefore, XML documents can be easily interpreted and anatomised.

XML documents are also self-descriptive in IT terms. All the information that computers need to deconstruct and interpret XML data is contained in the file itself. This makes processing possible without human intervention.

... *offers a variety of search options*

With XML the search possibilities are not restricted to full text searches; it is also possible to retrieve documents on the basis of their structure, metadata and semantic tags.

... *saves the structure of an electronic record in an explicit and formal manner*

XML is suited to a structured storage of information. The markup tags are nested according to a certain structure. The hierarchical structure of the elements reflects the relationship between the components of the record and is explicitly present in the document. A DTD or an XML Schema can further document the structure of a record.

... *is flexible, stable and re-usable*

XML document storage does not contain instructions on what to do with the data. The processing of XML data is entirely dependent on the application. As a result, data storage imposes no restrictions whatsoever on the future use of the computer data. XML-documents can be processed and used by several applications without the XML documents undergoing changes. The architecture in which the XML-documents will be re-used in the future, will evolve and change, but the underlying XML-documents can remain unchanged. Therefore, there is less danger for loss of information or corruption of electronic records.

... *is both data-centric and document-centric*

XML softens the boundaries between database-oriented (regular structure, small field lengths, management) and document-oriented (irregular structure, long field lengths, human consumption) applications. Ultimately both worlds are evolving towards each other, because in reality text information tends to be semi-structured. Documents are gaining more structure (transmission, standardisation) and databases must be capable of containing data with an irregular structure. XML is capable of spanning the full spectrum, from weakly to strongly structured information. It is possible to record in one and the same XML-document very structured information (for instance e-mailheader) as well as semi- or unstructured data (for instance e-mailbody).

... *has a great deal of commercial support, is easy to implement and has broad market penetration*

All major software suppliers support XML[18]. XML can be used in Windows, MacOS and Unix. There are now more XML tools available than there ever were for SGML (including JAVA and PERL-based tools)[19]. Furthermore, XML is integrated in existing software packages (e.g. web browsers[20], MS Office 2000[21], MS Office XP[22], OpenOffice.org[23], WordPerfect 9, etc.).

XML-softewaretools are available as freeware as well[24]. Seeing that XML is in essence nothing more than flat text, it's relatively easy to create your own XML-tools.

... *has its disadvantages too:*

- low performance:
  - o due to the full text encoding of the data, XML documents are sometimes long-winded, and they usually require more space than files with binary encoding The file seize can expand rapidly, especially in XML-documents with many XML-tags (for instance when archiving a database in which every field is tagged.)To improve performance XML documents are

---

18 http://wwws.sun.com/software/xml/; http://www.novell.com/products/edirectory/dirxml/; http://www-106.ibm.com/developerworks/xml/; http://xml.coverpages.org/xmlSupport.html

19 For a list of XML tools: http://www.w3.org/XML/#software;http://www.oasis-open.org/cover/publicSW.html; http://xml.coverpages.org/xml.html#xmlSoftware; http://www.xml.org/xml/resources_focus_programming.shtml

20 Internet Explorer: Version 5.0 is based on XML Recommendation 1.0 and on a working draft of the XSL specification. In IE, client-side XSL processing is possible.

Netscape: Version 6.0 is entirely XML 1.0 conformant, but does not yet support XSL because at the time of release the XSL specification had not yet reached completion.

21 MS Office 2000: use of XML data islands when saving Excel, Powerpoint and Word documents as web pages. This means the documents can be viewed using web browsers. Data islands are blocks of xml content in an HTML page. The non-official xml tag is used here.

22 MS Office XP: import of XML documents in Excel and Access, storage of spreadsheets, tables, queries and views in Excel and Access as XML documents, use of XML smart tags in Word, Excel and Outlook. In Outlook views are stored as XML files.

23 OpenOffice.org: with the exception of Math all applications in OpenOffice.org (Writer, Calc, Draw, Impress, Writer global document) use the same format, which is based on XML. OpenOffice.org files contain metadata (uncompressed) and the bitstream for the digital objects (ZIP-compression). OpenOffice.org files are compiled from: meta.xml, styles.xml, content.xml, settings.xml, meta-inf/manifest.xml and the pictures, dialogs, basic and obj. folders (http://www.openoffice.org).

24 Bijv. http://www.garshol.priv.no/download/xmltools/

normally compressed when exchanged between networks ((g)zip, http 1.1, modems, etc.). Very high compression rates can be achieved (up to 90%) with XML documents: it envolves textfiles in which the same tags appear several times. Compression is lossless. Specific compression methodes for XML exist. (for instance XMIL and XMLZip).

- o parsing and conversion operations from and to XML take time

– not many client computers are XML and/or XSL compatible as yet, and so most XML applications have to perform server-sided parsing and transformation operations.

– expandability of XML. XML is freely expandable and adjustable to the needs of the creators and softewareproducers. The expandibility of XML implies that everyone can develop freely their own application bounded schemes and XML-language. The consequences in the long run remain to be seen. Because of the future exchangeability and interpretability, it's in any case important that DTD's and XML schemes, possibly replenished with other documentation, are archived along with the records. This way future (archives) users can use the documentation model to beter understand the records.

3.7. Storage and retrieval of XML documents

XML documents can be stored in two ways: in the file system of a current operating system or in a database. The choice of storage affects the way in which XML documents can be located later on. With XML it is not just a case of finding the XML document in question, but one of gaining access to the data objects in the document too.

3.7.1 In a file system

Storage in a file system is simple and carries the advantage that the XML data can be accessed quickly. XML documents are simply stored in folders as separate computer files with the *.xml extension. The possibilities for tracing XML documents are restricted to searches based on folder and file names and to full text search commands. However, full text searches will not give particularly refined results since they do not distinguish between markup characters and content characters. This method of storage has the disadvantage of being limited by the operating systems: restrictions in maximum file size, difficult navigation through large files, no version control, no multi-user access, low levels of security, etc. It may be a simple matter to store XML documents in a file system, but this is not recommended for large archiving applications. However, the method may well be appropriate for storing smaller quantities of XML documents.

3.7.2 In a database

Database storage is a better recommendation for large applications. Databases offer a solution to the shortcomings of ordinary operating systems. They offer more search functions. XML documents are always stored in relational, object-oriented or native XML databases.

3.7.2.1 Storage in a relational database

Since XML documents have a hierarchical structure they cannot be saved in a relational database as they are. After all, there are some fundamental differences between the relational and hierarchical objectmodels[25]. Still

[25] Some differences between the XML-documentmodel and the relational datamodel

XML:	RDBMS:

it is possible to record XML-documents in a relational databasesystem. XML documents are stored in the database as objects, or their XML content is translated into a relational data model. Here the elements are mapped to tables and columns and their relations are expressed through different join types. On top of the relational engine many commercial relational databases contain an XML mapping layer to store and retrieve XML data. XSLT transformations, Xpath expressions and XQueries also have to be converted. XML applications also have to repackage the output as an XML document afterwards. All of these conversions on input, query and output take time and enlarge the danger that XML-documents undergo changes or loose information. Nonetheless, a relational database usually provides search results faster than an object-oriented database. Relational databases are best used if the XML documents are not too complex, or if they are well structured. Storing badly structured or unstructured documents in a RDBMS is not efficient, because much of the storage space is lost.

3.7.2.2 Storage in an object-oriented database

The hierarchical data model for object-oriented databases is better suited to XML documents than that for relational databases. XML documents are not stored as one entity in an object-oriented database either, but the nodes are stored as separate objects in a given hierarchy. In- and output is done on an XML basis, without preliminary transformation or mapping so the danger of corrupting the original XML-document reduces. XML documents are retrieved and manipulated on the basis of XML queries of the nodes. Every node can be processed separately, without having to load the intire XML-document. Processing on the data is possible as well as on the nodes (metadata). The use of object-oriented databases is recommended in cases where the XML documents have complex relations and a great depth of structure.

3.7.2.3 Storage in a native XML database

Native XML databases are specially designed to store XML documents. The main difference with other database systems is that their internal model is based on XML documents and not on a relational or object-oriented data schema. The XML documents constitute the basic unit of native XML databases. Native XML databases respect the order and internal structure of XML documents. The XML engine searches in query languages based on XML (e.g. XQL, XML-QL, XMAS) and returns the data in the form of XML documents. The XML documents in an XML native database are stored in linear fashion. Therefore no time is lost through mapping, join instructions, and XML output wrapping. XML documents can be stored directly in the database and queried. Therefore native XML databases usually perform better than relational or object-oriented databases. This database technology is still fairly young and so only a small number of tools are available at present.

4. The XML family

An XML document does nothing in itself because in essence it is nothing more than a set of Unicode characters packaged in XML tags. XML describes the data used to structure, save and send information. Other mechanisms are needed to add presentation and functionality to XML documents. All kinds of other standards have been defined around XML, and they are used in combination with XML. Together with XML these standards constitute the XML technology[26]. Most of the standards have been set by *W3C* and form the basis for the XML applications that manufacturers develop and implement in their products.

– data is stored in one hierarchical structure	– data is spreaded over several tables
– nodes have elements and attribute values	– cells have a single value.
– XML-documents are the basic unit, elements have an order and are nestable	– atomique datacells, no arrangement in tables and rows
– scheme optional	– schema needed

26 XML encryption, XML key management and XForms are still in the development phase.

CSS *Cascading Style Sheet* Version: Level 1 (CSS1): W3C Recommendation (17 December 1996), reviewed on 11 January 1999; Level 2 (CSS2): W3C Recommendation (12 May 1998)	CSS is the style sheet language for HTML, but can also be used in combination with XML.
XML Base Versions W3C Recommendation 1.0 (27 June 2001)	Facility for producing base URIs to indicate parts of XML documents.
XML Signature Version: W3C Recommendation 1.0 (12 February 2002)	Defines a framework based on XML to add digital signatures to sources on the web.
XML Query Version: W3C Working draft (30 April 2002)	Query language for XML documents, also known as XQL. Not yet formally established.
XML Schema Version: W3C Recommendation 1.0 (2 May 2001)	See 3.3 DTD and XML Schema
XPointer Version Candidate Recommendation (11 September 2001)	Application based on XPath for referring to certain components of XML documents.
XLink *XML Linking Language* Version: Recommendation 1.0 (27 June 2001)	Offers the possibility of adding hyperlinks to XML documents, such as in HTML pages, but XLink also facilitates more sophisticated links <link xml:link="simple" href="locator">text</link>
XSLT *XSL Transformations* Version: W3C Recommendation 1.0 (16 November 1999)	XSLT is mainly used for transforming data from one XML structure to another. Hence the XML data can be expressed in another format or structure. In this sense XSL is comparable to CSS, but XSL is more powerful (including selection of elements, filtering, sorting transformation of XML tree, data dependent layout). XSLT is also the most effective technology for transforming data recorded in an encoding language into a presentation language such as SVG or X3D.
XSL *eXtensible Stylesheet Language* Version: W3C Recommendation 1.0 (15 October 2001)	XSL gives XSLT the possibility of *formatting objects* and *formatting properties*. XSL consists of three parts which are usually combined with each other: XPath, XSLT and XSL Formatting Objects. XPath gives access to the nodes of an XML document, XSLT provides a link with templates and XSL deals with formatting objects. Among other things XSL-FO (Formatting Objects) is used to create PDF documents based on XML data. The elements in the XSL:FO vocabulary define aspects such as margins, headers, footers, lines, spaces, word wrapping, etc. First of all the XML content is converted to formatting objects, after which an FOP tool converts these to PDF. The presentation options for XSL-FO are comparable with those for HTML and CSS.
XPath Version: W3C Recommendation 1.0 (16 November 1999)	XPath is a string syntax based on URIs, for creating addresses for XML documents, elements or attributes. Xpath expressions are used to localise information in XML documents. XPath works from the hierarchical position of the components in the tree. The nodes are used as unique IDs. XPatch offers no other query functions, and so XPath is not a real query language. XPath is used by XSLT and XPointer
MathML *Mathematical Markup Language* Version: 1.0 (July 1999), 2.0 (February 2001)	Used to mark up maths on the web.
DOM *Document Object Model*	A communal *Application Programming Interface* (API: collection of standard functions) through which the elements, structure, events,

Version: Level 1 (October 1998) and Level 2 (November 2000)	etc., are approached separately from different program languages and/or scripts. DOM also provides a standard interface for other software to process XML documents.
RDF *Resource Description Framework* Version: W3C Recommendation (22 February 1999)	W3C's metadata standard, RDF, contains a standard lexicon and coding techniques for linking metadata to any source on the web. RDF is designed to improve searches on the Internet. For example, one RDF application is a file placed in the root map of a web site containing metadata on the web site. The user describes objects, and ascribes attributes/properties. He can also set statements on the objects. These statements may be relations between objects, for example.
XML Namespaces (January 1999)	With XML everyone can compose his/her own vocabulary. Thus there is an exceptionally high chance that elements with the same tags will have a different meaning. XML Namespaces was developed to prevent tab-naming conflicts and other exchange problems. The user's vocabulary can be identified to avoid ambiguous information. The namespace is identified via a URI (URL or URN). The tags that make up a given namespace have a common prefix. Different namespaces can be used in the same XML document. If a default namespace is declared it will be valid for the entire document. A namespace that is an attribute of a parent element also relates to the child elements.
SAX *Simple API for XML* Version: 2.0.1	Java API developed for use with XML. Advantages: simple, the entire document need not be loaded into memory. Disadvantages: not supported by all browsers, only for XML documents, complex searches can be difficult. SAX was not developed under the auspices of W3C[27]. SAX is currently used for environments other than Java.
SVG *Scalable Vector Graphics* Version: W3C Recommendation 1.0 (4 September 2001), W3C Candidate Recommendation 1.1 (30 April 2002)	Specification based on XML for the dissemination of vector graphics on the web.
XTM *XML Topic Maps* Version: TopicMaps.org specification 1.0 (2001), annex of ISO-13250 (October 2001).	XML namespace for composing a topic map. XML is used as a means of notation, XML based URIs are used for the links. A topic map consists of associated topics and reflects knowledge.

5. XML for electronic recordkeeping

5.1 XML as an exchange format

XML can be used to exchange structured information between different information systems. This has several major advantages: faster exchange, avoidance of errors, automatic exchange, etc. XML owes its description as a suitable exchange format to the text encoding of all markup and content, the separation of content and layout, the use of unique tags as delimiters, the explicit presence of structure, etc.

This means that XML documents are easily exchanged between applications, and the receiver has no need to know how the computer data are organised at the other end. There is no need for extra coding. The only requirement is that the rules of document creation be followed and that these be laid down in a DTD or XML Schema.

[27] http://www.saxproject.org/

XML has many areas of application as an exchange format. At present, content providers make full use of XML to disseminate digital information. They also use the option of sending the same source document to the user in different formats. Incompatible computer systems can interoperate on the basis of XML[28]. XML has been put in place as an intermediate format for exchanging data between different types of databases (e.g. from relational to object-oriented) or databases with different structures. In the database environment XML interfaces are gaining more and more ground as an ODBC replacement. At present IETF is working on an XML exchange format for e-mail between different applications as a variant of RFC822[29].

In the archiving context XML can be used as a transmission format between the archivist and the archive institution/department. For example, XML can be used if the archive institution/department does not support certain software applications, but archives documents with archival value nonetheless. The creator saves documents with archival value as XML data outside of the original application and deposits them in this form with the archivist. The archiving department can then transfer this XML data to its own application or archive it in XML document form.

5.2 XML as a preservation format

In the second case XML is used as a file format for long-term archiving. As a file format, the characteristics of XML are very close to those of the ideal preservation format. The ideal preservation format is:

– stable and standardised
– free of patent rights (public domain)
– as simple as possible
– fully self-descriptive, not dependent on external sources
– suitable for containing structured and self-selected metadata
– fully documented (open specification)
– not application or platform based, but easily exchangeable
– used and implemented widely
– easy to implelent and to check for errors (validation)

The XML file format meets the vast majority of these requirements. As a result XML is extremely well suited as an application-independent storage format. The user is no longer bound to versions, applications, platforms or even manufacturers to access his computer data. As opposed to manufacturer-based storage methods, the user is no longer dependent, for example, on closed formats or schemas when writing data to disk or opening files. XML applications are therefore open and transparent and allow for the easy incorporation of XML data in future developments. Furthermore, XML documents do not have to go through a process of migration when transferred to another information system. And so the chances of losing information or compromising authenticity and integrity as the result of migrations fall appreciably. XML is not an application-based format and so it offers the user greater independence. The same XML data can be used in desktop applications, databases, and in intranet and Internet applications. In this sense XML data are comparable to SQL data[30].

The autonomy of XML files is a major strong point. The independence of XML documents from external factors guarantees the reconstructability of digital records. XML files are and do remain digital. Software is always required to consult them, but a higher degree of software-independence is not possible or even

[28] An example of an XML application would be the CEVI E-Loket, whereby data received via on-line forms are sent to the administrative applications in XML format. (http://www.cevi.be/cevi/site/realisaties/cevigem/default.htm). Another example is the *Universal Message Engine* for exchanging structured messages between heterogeneous government applications (e.g. Population Register).

[29] G. KLYNE, *An XML format for mail and other messages*.

[30] An example from the museum sector is Adlib. XML was introduced in version 5.0 so that all descriptions of museum objects could be written to disk as XML data.

desirable. After all, we are dealing with electronic records here, and we will always need software to reconstruct the recordss from the bits that have been saved.

Compared to other file formats with defacto or open standard status, XML is better able to meet the requirements of a suitable archiving format. Adobe's Portable Document Format (PDF) is regularly put forward as an archiving format, but has a number of disadvantages compared with XML. PDF is property-based, has no semantic markup, offers fewer search functions and is not "human-readable". A PDF document cannot be interpreted simply with the aid of a text editor because the reader sees only incomprehensible signs. In other words, PDF files are more software-dependent than XML documents.

The ratio of XML file to archive documents can vary from one-to-one, one-to-many, and many-to-one. In XML it is all about the logical structure of the computer data. One XML file can contain one or more records and one records can be spread over several XML files.

In the first place XML is recommended as an archiving format for text documents. In addition to guarantees of digital durability, XML offers the advantage that the record structure can be archived along with the document in an explicit manner. The document model can be formally set in a DTD or XML Schema. Once set by the archiving department, and prior to ingestion in the recordkeeping system, the records can be parsed against the DTD or XML Schema as a sort of quality control. By adding a DTD or XML Schema we can increase the self-descriptive character of the XML document. Since XML documents can vary in structure from weak to strong, it is possible to archive ordinary text files, spreadsheets, e-mail messages and databases as XML files. The hierarchical tree of an XML document is just as likely to contain the relational, object-oriented and ordinary text data of a letter, e-mail or spreadsheet.

A migration phase will usually precede any archiving of text documents as XML files. At the present time the industry is working on the implementation of XML in computer applications, but as long as the applications themselves do not store the data as XML data, there will be a need to migrate native formats to XML. In many cases this means that a static version of the digital document will be saved and that a number of dynamic functionalities will be lost. These functions are typically application-based and seldom have archival value. The exportation of computer data as XML files is beginning to find its way into computer programs (e.g. MS Office). *OpenOffice* is probably the most advanced in terms of XML and uses XML as a native format for all applications in the package[31].

Despite the increasing spread of XML many existing applications do not offer XML functions. However, many of these computer applications have been used to create documents with archival value. To convert these records from their application format to XML, we need a solution that allows us to save the records in XML format. One option would be to use one of the many available conversion tools. However, even if we have at our disposal an export function or conversion tool[32], we should question the wisdom of using it. Existing, automatic export functions or conversion tools seldom allow the user to define the tags and structure of the document, or seldom follow an external document model. As a result, the tagging and structuring of XML documents is usually dependent on the export function or conversion tool used. This gives rise to difficulties when we wish to apply one document model to a certain type of record, while this is precisely one of the strong points of XML. Using different export functions and conversion tools with fixed DTDs for the same type of electronic record results in different document models that require different DTDs, style sheets and consultation applications. As a result, electronic records management becomes more complicated.

Before using an existing export function or conversion tool it is important, therefore, to check the extent to which an established document model can be applied. The ability to configure is an important factor in the choice of tool. If XML storage does not satisfy the established document model, there are two options. Either convert the XML files to the document model after exporting (e.g. via XSLT), or provide a customised XML wrapper that respects the document model immediately[33]. The second option is more efficient because it saves an intermediary step and avoids the problem of splitting XML elements.

[31] *OpenOffice.org XML File Format, 1.0*, July 2002.

[32] MS Word files can be converted to XML using XML Spy for example. Standalone converters for converting WORD → XML include, for example Majix (http://www.tetrasix.com) and UpCast (http://www.infinity-loop.de). Tools integrated in MS Word itself, such as S4/Text (http://www.i4i.com), create XML documents that can be re-opened and processed as Word files. Conversions from Word to XML can also be made using VBA, Omnimark or Perlscripts. Many tools convert Word files to XML via Rich Text Format (RTF).

[33] DAVID used this solution to archive e-mail in XML. On the basis of the e-mail document model a script was written to save e-mail directly from the client program as an XML file.

The XML wrapper must ensure text output, correct encoding, the replacement of reserved characters and the filtering out of invalid XML-characters such as certian control characters. Obviously Unicode is best used for encoding, although this is not always straightforward. Unicode is relatively new. Most textual digital information is not stored in Unicode and only the latest generations of operating systems and applications support it[34]. In the event of a retroactive migration to XML the chances are high that reserved characters (&, <, >, ', ") will be used as content characters. When migrated to XML these characters must be changed to an entity. The file can swell considerably in size when semantic tags are added. Finally, when converting to XML it is important to consider the storage place for DTDs or XML Schemas. To increase the self-descriptive character and autonomy of XML documents we might consider recording them as internal DTDs or internal XML Schemas in the XML document. However, this is inconsistent with the principles of re-usability and information source sharing. The disadvantage with storing the DTD or XML Schema in a folder (local or on the (web) server) is that the path has to be changed in the URI when the DTD or XML Schema is moved or the web servers' domain name changes.

The text content of databases can also be archived in XML files. In themselves XML files are not databases, but documents, because they only contain linear text, and no indices, datatypes, relations or other logic used to process or query data. XML does offer the possibility of structured data storage and the XML family provides query tools and programming interfaces. In any case, with XML the boundary between database and document has blurred. How database content is converted to XML depends on the type of database. Hierarchical databases already involve a structure and nesting with strong similarities to XML (root-parent-child). XML tags have to be added on export. The data model for a hierarchical database forms the basis for the DTD or XML Schema[35]. With relational databases the conversion to XML is a little more complex. The fields of one record are spread over different tables. XML elements or documents can be constructed from SQL. To the query in the main table we link sub-queries, which are performed in related tables. In the next step the output is tagged and given linear structure (nesting) thereby creating an XML document. The primary key can be used as (a unique) ID, so that all child objects fall under the right parent. With object-oriented databases there is a high level of agreement between objects and XML elements[36].

Keeping in mind the size of the XML-documents, which need to be archived, it's important to consider in advance if the databasefields or the databaserecords will be tagged. Tagging the fields offers the advantage that databasefillers can be removed and that the content of the individual files can be adressed directly. The disadvantage is, of course, that the same information (the tags) is repeated a lot of times and that a big amount of redundance is created. Using abbreviations in the XML-tags instead of semantic XML-tags can be a compromise, but this way XML -documents will lose part of their autonomy and self-describing character[37]. In the case of tagging records an XML-tag is placed between the records as a delimeter and the original record layout is preserved[38]. Within the database record, the usual field delimeters seperate the fields. No mather which path is taken, archiving databases as an XML-file results in large files. To manipulate or to re-use large files, it's evident to use SAX (Simple Api for XML: event-based). The other XML API, DOM (Document Object Model: tree-based), will load the entire document in the computers' memory, while SAX is more economical and loads only the nodes that are needed .

XML is not only suited to the archiving of text documents. Scalable Vector Graphics is a file format based on XML for graphics and images. XML files can also contain the text bitstream of binary files (graphics, sound, etc.). However, we should bear in mind that the right software will be needed to open the files once the text characters have been re-converted to binary signs. Consequently, it is best to opt for a standard format whose specification is freely available. In future it will probably be possible to store multimedia files as pure XML documents and not merely to pack them in an XML wrapper. After all, XML is extensible.

[34] Notepad in Windows 95 and 98, for example, does not support Unicode, which can cause problems with a few diacritical signs. This can be solved by entering appropriate coding in the XML declaration. E.g.: `<?xml version="1.0" encoding="windows-1252"?>` of: `<?xml version="1.0" encoding="ISO-8859-1"?>`. As from Windows 2000, Notepad does support Unicode.

[35] The city of Antwerp's archiving of its electoral roll and population register was an application involving the XML archiving of a hierarchical database (for more info: http://www.antwerpen.be/david under "cases").

[36] R. BOURRET, *XML and databases*, Febr. 2002.

[37] This strategy was applied for the preservation of the population register of the city of Antwerp (see http://www.antwerpen.be/david --> cases --> population register.)

[38] The Swiss Bundesarchiv (ARELDA) applies this strategy of XML-preservation when archiving relational databases (http://www.erpanet.org/www/products/bern/bern.htm; http://www.heuscher.ch/PaperXmlUrbino2002).

In the archival community XML is generally labelled as one of the best file formats for long term archiving. In many countries XML has been formally declared as an appropriate format for archiving. XML meets the requirements of the MTIC and is the recommended standard of the French government[39]. In the Netherlands XML is listed with PDF in the *Regeling geordende en toegankelijke staat archiefbescheiden* (art. 6) as the standard for text documents. The *Public Record Office* recommends XML when following the path of migration[40]. In 2000 the Antwerp Records Department applied XML for the first time when archiving the electoral register and population register. XML has also been used in Antwerp to archive e-mail[41]. XML plays an important role in the *Persistent Object Preservation* archiving strategy of the NARA and the San Diego Super Computer Center (SDSCC). DTDs were set up for the different types of electronic records, depending on documentary form. When ingested in the digital archives the records were marked with XML tags on the basis of these DTDs.

5.3 XML as a metadata format

In the world of electronic records, the metadata constitutes the administrative, technical and archival descriptions used to ensure future reconstruction and interpretation. Metadata will be needed whatever archiving strategy is followed. Just like the records themselves, metadata has to remain accessible too. As a result, metadata is usually printed on paper or stored in a pure ASCII file. XML offers several possibilities for storing metadata. Firstly, through the addition of semantic tags, the XML document itself contains identifying and characterising information on the data. Secondly, the metadata on electronic records can be stored in an XML document.

By storing the metadata in an XML document we satisfy the legibility requirement to some extent. Moreover, we can store the metadata in a structured way. The semantic nature of XML documents makes XML a suitable format for the storage, querying and exchange of metadata. A metadata model can be easily created in XML. In theory, any kind of descriptive system can be fed from XML files containing metadata. XML is clearly a better option than storing metadata on paper or in an ordinary text file.

The METS, Metadata Encoding and Transmission Standard[42], was developed for libraries. This is a standard in XML for storing and exchanging metadata for compiled objects (multimedia). For example, the Dutch *Regeling geordende en toegankelijke staat archiefbescheiden* (art. 6) prescribes that documentation on the database structure or metadata of a text document in TIFF or PDF be stored in an XML file. The Antwerp City Archives stores the metadata on archived websites in XML files. The archiving prototype developed by CEDARS uses XML to set out *Preservation Description Information* (PDI) as a component of the *Archival Information Package* (AIP). In France XML is recommended as the most suitable format for the digital storage of metadata[43].

Not only does the *Persistent Object Preservation* method of the NARA and the SDSCC make use of XML, DTDs, and style sheets, but it uses XML Topic Maps to reflect knowledge. The XML Topic Maps[44] establish the link between the records and the operating process in which they are received or created. A DTD is created to lay down the (hierarchical) structure of the records collection. In this way the relations between the archive components are formally and explicitly archived. The logical relationship between the attributes of the entire collection are described in Data Definition Language (DDL) and mapped to the DTD for the whole[45].

One simple application here is describing a folder structure in an XML file. Traditionally, digital documents are organised into folders. The folder layout is important in maintaining the archival bond between the

---

39 MTIC, *Guide to the conservation of data and electronic documents for teleprocedures, intranets and Internet sites. Formats, media*. Paris, 2001, page 7.

40 *Sustainable electronic records: strategies for the maintenance and preservation of electronic records and documents in the transition to 2004*, page 23.

41 For more information see DAVID publications the DAVID web site (under "cases").

42 http://www.loc.gov/standards/mets/

43 C. DHÉRENT, *Les archives électroniques. Manuel pratique*, Febr. 2002, (under IV. 2.Constituer les métadonnées).

44 http://www.topicmaps.org/xtm/index.html. See also the DAVID contribution *XML Topic Maps voor digitale archivering*. (Dutch only)

45 K. THIBODEAU, R. MOORE and C. BARU, *Persistent Object Preservation: Advanced Computing Infrastructure for Digital Preservation*.

documents, and between the documents and the operating process. Since in current operating systems folders are nothing more than a sort of index, we might consider describing the structure and composition of the records collection within an XML file in a more formal and explicit way. An XML file of this type would then describe the "*respect de fonds*". It would be relatively easy to keep the DTD for a folder structure[46]:

DTD for a folder structure:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!ELEMENT folderstructure (folder | file)*>
<!ELEMENT folder (foldername, content)>
<!ELEMENT content (folder | file)*>
<!ELEMENT foldername (#PCDATA)>
<!ELEMENT file (#PCDATA)>
```

5.4 XML as an encapsulation format

Encapsulation is an archiving strategy in which the records and the information needed to store, interpret and retrieve it are grouped together in a new digital object. The new digital file functions as a document archiving format and metadata format combined. The metadata are encapsulated so that this information is inextricably bound to the electronic record. In this way the link between the records and metadata is retained and the record's dependence on external sources is restricted. The latter is not unimportant since with digital archiving we must work on the basis that a loss of the external source can result in a loss of the record. By recording that source or its building blocks along with the record we make sure the information is available, or make it possible to recreate the external source. In this way, for example, we can reduce dependence on the records management application. At a given point the records management application will be faced with technological obsolescence. An alternative to converting the data from this application would be to recompile the external sources, such as databases and indices, based on the metadata held in the digital records themselves. After all, with encapsulation the metadata are stored together with the archived computer files. The new digital object corresponds physically with the logical record unit and the metadata.

The principle of the encapsulation format can be found in the *Open Archival Information System* (OAIS)[47], the standard model for a recordkeeping system. In an OAIS, an AIP consisting of the record components, information on the archive, and the information needed for presentation, is compiled when recording to the digital depot. XML is perfectly consistent with the encapsulation format requirements described in OAIS. One XML file can correspond to one AIP and combines the functions of the archiving format and metadata format. Furthermore, recording the metadata in the XML file ensures that the information is archived in a long-term and legible manner. Hence, XML is used as a combined archiving and metadata format.

The type of metadata recorded in the XML file can differ greatly: contextual information (archival code, reference to case or dossier, related records), compression algorithm, hash code, key words, references to documentation, specification file format, index data. The XML file can even include the bitstream for the viewer needed to view the record. It would be possible, for example, working from this encapsulated metadata, to reconstruct a viewer or external index afterwards, if it were lost. If the same metadata were related to a large quantity of records it would be best to check in advance whether to record the metadata itself or just a reference to commonly shared metadata (e.g. specification of a file format).

The encapsulation of records in XML files can be applied to any type of computer file. Packaging electronic records in an XML file therefore restricts the number of different types of items that can be stored in the digital warehouse. Encapsulation in XML files makes the electronic archives fairly homogenous in composition, which simplifies the task of records management.

Encapsulation can be applied in several ways for the purposes of electronic recordkeeping. The metadata can be added to the bitstream of the record itself. Encapsulation can also be applied to store several versions

[46] A fine example of this application is available at the DAVID web site (http://www.antwerpen.be/david → cases).

[47] www.ccsds.org/documents/pdf/CCSDS-650.0-R-1.pdf

of the same record as a single digital file. The same XML file can contain both the text (XML) and bitmap (TIFF) presentation of the record.

XML is used by VERS (*Victorian Electronic Records Strategy*[48]) to encapsulate records. The XML file contains the archive object itself, the digital signature and the metadata. In VERS, the archive object is archived in the PDF format. The Dutch *E-Archive* project uses XML files as containers for the metadata, the original bitstream, the viewer list (file names for AIPs with source codes for appropriate viewers), and alternative representations for bibliographic items. The AIPs for the viewers are also XML files[49]. The XML application of the *Persistent Object Preservation* method can also be considered as encapsulation.

6. Conclusion

XML is not the ultimate solution for all eventualities in the field of electronic recordkeeping, but goes a long way in assisting the archivist. In the world of archiving XML is gradually gaining currency as a metadata and archiving format. For the most part this answers the problem of technological obsolescence and has the advantage of enabling the structured and semantic storage of electronic records. Currently no technology can offer better guarantees of long-term legibility. Moreover, XML documents can be easily converted to new technologies. Document modelling is not yet commonplace. By applying XML as an encapsulation format we can limit dependence on external sources and further increase the autonomy of electronic records.

7. References

http://www.w3C.org

http://www.xml.com

http://xml.coverpages.org

http://www.covax.org/

http://www.topicmaps.org/xtm/index.html

http://www.w3schools.com

R. Bourret, *XML and databases*, Febr. 2002. (http://www.rpbourret.com/xml/XMLAndDatabases.htm)

J. Clark¸ *Comparison of SGML and XML*, 1997 (http://www.w3.org/TR/NOTE-sgml-xml.html).

S. Gilheany, *XML for records managers*, (http://www.archivebuilders.com)

D. Martin et al, *Professional XML*, Birmingham 2000.

R.W. Moore, *The preservation of data, information and knowledge*, 2001.

T. Shinkle and R. Marciano, *Using "XML" Standards for the preservation of records*, lecture at the *ARMA 2001 Annual Meeting*, 3 October 2001, Montreal.

Testbed Digitale Bewaring White Paper, *XML en digitale bewaring, 2002.*
(http://www.digitaleduurzaamheid.nl/bibliotheek/docs/white-paper_xml-nl.pdf)

K. Thibodeau, R. Moore and C. Baru, *Persistent Object Preservation: Advanced Computing Infrastructure for Digital Preservation*, in: *Proceedings of the DLM-Forum on electronic records. European citizens and electronic information: the memory of the Information Society*, Brussels, 18-19 October 1999, pages 113-118.

E. van der Vlist, XML for document preservation, 2001. (http://www.xmlhack.com/read.php?item=1030&r=1)

K. Williamson, *XML: The complete reference*, Berkeley, 2001.

[48] http://www.prov.vic.gov.au/vers/welcome.htm

[49] http://www.library.tudelft.nl/e-archive/