OPEN DOCUMENT FORMAT AS A NEW STRUCTURED FORMAT STANDARD IN LONG-TERM ARCHIVATION

Rudolf Vohnout
Ústav systémového inženýrství a informatiky, Fakulta ekonomicko-správní,
Univerzita Pardubice

***Abstrakt**: Pro požadavky dlouhodobé archivace je nezbytné aby uchovávané elektronicky podepsané dokumenty byly ve strukturálním formátu. Protože se jedná o archivaci po dobu několika desítek (až stovek) let každý musí být explicitně dokumentován a musí se jednat o dlouhodobý standard. Extensible Markup Language (XML) jako jeden z nástrojů jednoznačně definované struktury a syntaxe vytvoření takového standardu umožňuje, avšak pouze za předpokladu, že tyto podmínky budou striktně dodrženy. Touto cestou se snaží jít OpenDokument formát (ODF) a stává se tak důležitým nástrojem nejen pro složky veřejné správy, ale také pro podnikatelské subjekty.*

***Abstract**: For Long-Term Archivation requirements it is necessary to preserve electronically signed documents in structural format. Every one has to be explicit documented because we act with archivation for tens (even hundreds) of years and it has to be long-term standard. Extensible Markup Language (XML) as one of the tools of exact structure and syntax defined tools allows us to make such standard, nevertheless providing that conditions will be met. OpenDocument format (ODF) tries to go this way and it becomes important tool not only for public administration authority, but for business subjects as well.*

***Klíčová slova**: dlouhodobá archivace, XML, ODF, dokument, formát, standard*

***Keywords**: Long-Term Archivation, XML, ODF, Document, Format, Standard*

1 Introduction

The main idea for long-term archivation (tens of years) is to archive documents for a long period of time. These documents have to be electronically signed to prove their authenticity and eligible owner(s). This can be realized by usage of long-term electronic signature[1] services. Long-term archivation should take-over today archives, but without tons of papers.
For example in the Czech Republic companies have to archive accountancy documents up to 30 years. The difficulty is the short life of the software and hardware environments. You need a new computer every third year and have problems reading documents older than ten years, because today's software can only import a limited selection of file formats. Regarding to these requirements long-time archivation (e.g. for 50 years period) has to force following problems:

- ✓ *Hardware* (today top hardware will be museum exhibit after 50 years).
- ✓ *Software*
  - ➢ *Operating system* (no one of today's computers will be able to run operating system which will be current in the year 2056).
  - ➢ *Program itself* (it has to be able to run under OS mention above and to handle 50 years old file formats).
- ✓ *File Format* (universal file formats must exist, which won't be affected by matter of time).

[1] Trustworthy Time-Stamp Authority (TSA) proves that electronic signature in electronically signed document existed in time electronically document was created (modified). Simply it verifies electronic signature existent at instant of time in the past.

In the begging there were 3 draft solutions:

- ✓ *The technological museum*[2]
- ✓ *The migration*[3]
- ✓ *The emulation*[4]

Last few years solution of the problems above is long-term trustworthy archive which takes the best of migration solution above but with many new features and it focuses on security as the highest priority.

2 Long-term trustworthy archive

2.1 Basic description

The basic idea of the trustworthy long-term archive is very easy:

- ✓ Before archivation, the archives verify the validity of the document.
- ✓ Ensure that the document will not be lost or modified in the archive.
- ✓ When the document is picked up from the archive, the archive may confirm the validity of the document. The archive can do it because it already verified the document validity when it archived it. And logically because the archive is trustworthy thus the picked up document must be genuine.

Trustworthy archive authority (TAA) is an authority into which we give our digital documents and we trust it that it does not change them, lose them or does not allow any unauthorized person to access them. If the trustworthy archive verifies the digital signatures of the document before archiving it, the digital signatures should remain valid also after picking up the document from the archive. The archive confirms the user the acceptance and archiving of his document. If the document is picked up from the archive, the archive gives to the user a confirmation of the digital signatures validity and authenticity.

From the top level point of view, the TAA is a bounded system, which provides its functionalities to producers and consumers, and which behaves as they require. The basic functionality is controlled sending of documents, which are required to archive, from the side of documents´ producer to the side of system TAA. This process is preceded by the process, called „pre-archiving phase“, when metadata and digital signature are added to the relevant document. After producer authentication and authorization, received documents are archived - if valid, or rejected. The TAA system issues a confirmation of successful archiving. Providing of a document from the archive is based on usage of the document metadata for concrete producer, which are presented to consumers (persons requiring access to archiving documents).

The trustworthy storage should have three basic features:

- ✓ It should guarantee that the archived document will not be lost or modified.
- ✓ It should guarantee that no unauthorized person will be able to access the document.
- ✓ It should guarantee the sequence of the documents stored into the archive. It means if someone wants to cast doubt on the date and time when the document was stored, he should have to doubt also the date and time of documents stored to the archive before

[2] It contains old computers and software. Soon you will run out of spare parts and the cost to uphold the knowledge about it.
[3] It is conversion of the file to current format, when the old one is outdated. Drawback is that sometimes you lose some information or functionality of a document when it is converted from one software to another.
[4] It means reading old files by writing new software in your current computer environment, emulating the old programs or at least the reading part of them.

and after this particular document. It is more complicated and other persons witnessed the storage process. The sequence of stored documents could be done with the help of e.g. linking hashes.

2.2 Supported formats

The definition of long-term formats, which would be easily migrated to more advantageous format, is another inevitable condition of the system operation. If we look at present file formats, we can declare that an universal file format, useful for long-term archiving, does not exist (even if some foreign studies and projects assert the opposite …) So, it is necessary to make some compromise for these supported formats. In the TAA, the long-term formats are:

- ✓ TXT for text files.
- ✓ XML for documents.
- ✓ PDF for formatted documents.
- ✓ TIFF for graphical files (converted formatted documents can be also in this form).

Only data in these formats can be migrated to new formats in future if some reasons for their migration arise, in other words the TAA systems will not migrate unsupported formats to supported formats for their presentation to researchers. The list of supported formats can be either enlarged or reduced, according to requirements of future systems, technologies, and file systems.

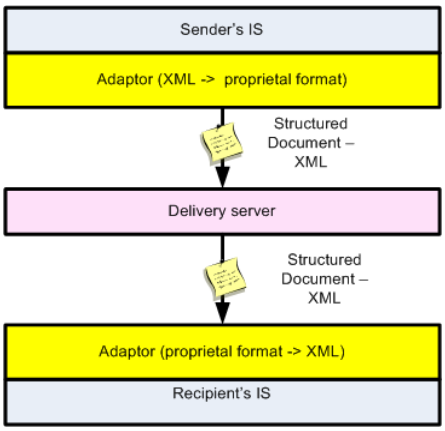
The "ideal" office document format what a government requires should be:

- ✓ *Open*: Valoris only looked for openness in the sense that a public, royalty-free specification is available. For example, PDF and MS XML both met this requirement.
- ✓ *Non-binary*: Binary formats get in the way of neutrality. A format that depends on Windows components will be hard to support on GNU/Linux and Mac OS.
- ✓ *Cross-platform*: An obvious requirement for neutrality.
- ✓ *Preserve format fidelity*: This refers to both presentation and structure. For many applications, format fidelity is an absolute imperative.
- ✓ *Modifiable*: This excludes formats like PDF.
- ✓ *Support current word processor features*: This list includes Unicode support, bi-directional (Hebrew), and scripting, among others. Abiword and KOffice failed this requirement.
- ✓ *Support emerging requirements*: Digital signatures, access rights, version control, etc. Almost every format failed this requirement.

In the next chapter we will focus on XML file format.

3 XML file format definition usage in long-term archivation

Each from Information Systems (IS) family works with some data formats which are not unified with each other, so-called *proprietary formats*, for example different version of MS Office, different database file types etc. To make the transaction and work up process between particular IS possible, they have to be converted to universal format – *structural document*. Some proprietary document formats have implemented a minimum level of functionalities assuring the document can be opened on different computers. They however remained incompatible between versions (e.g. different Word versions) and platforms (e.g. Mac & PC).



Source: Author

Figure 1: Adaptors (conversion bridges)

To solve the issue of incompatibility, word processors provide import and export filters, which allow to import and export documents to other formats. Newer versions of the software tend to be backwards compatible. However, as technology evolves word processors offer more functionality resulting in formatting information loss, and hence format degradation, if documents are converted to an older format. One of the structural document file type is XML.

3.1 Short XML description

In the beginning we have to explain important thing: XML in itself is NOT a format in its own right. XML only enables the definition of formats. Many formats are based on XML and yet they are not interoperable.

Primary purpose of XML is to facilitate the sharing of data across different systems, particularly systems connected via the Internet, allowing programs to modify and validate documents in these languages without prior knowledge of their form. XML provides a text-based means to describe and apply a tree-based structure to information. At its base level, all information manifests as text, interspersed with markup that indicates the information's separation into a hierarchy of *character data*, container-like *elements*, and *attributes* of those elements. By leaving the names, allowable hierarchy, and meanings of the elements and attributes open and definable by a customizable *schema*, XML provides a syntactic foundation for the creation of custom, XML-based markup languages. The general syntax of such languages is rigid — documents must adhere to the general rules of XML, assuring that all XML-aware software can at least read (*parse*) and understand the relative arrangement of information within them.

The schema merely supplements the syntax rules with a set of constraints. Schemas typically restrict element and attribute names and their allowable containment hierarchies, such as only allowing an element named 'birthday' to contain 1 element named 'month' and 1 element named 'day', each of which has to contain only character data. The constraints in a schema may also include data type assignments that affect how information is processed; for example, the 'month' element's character data may be defined as being a month according to a particular schema language's conventions, perhaps meaning that it must not only be formatted a certain way, but also must not be processed as if it were some other type of data. With XML, it is much easier to write software that accesses the document's information, since the data structures are expressed in a formal, relatively simple way.

What makes XML unique is that it can represent unstructured, semi structured, and structured data with equal ease. With XML there remain no barriers between document processing, handling or presenting. With XML we shift away from the paper-based document paradigm to

one where documents and data are intertwined. XML makes no prohibitions on how it is used. Although XML is fundamentally text-based, software quickly emerged to abstract it into other, richer formats, largely through the use of datatype-oriented schemas and object-oriented programming paradigms (in which the document is manipulated as an object). Such software might treat XML as serialized text only when it needs to transmit data over a network, and some software doesn't even do that much.

XML as a heavily used format for document storage and processing, both online and offline offers several benefits, e.g.:

- ✓ its robust, logically-verifiable format is based on international standards,
- ✓ the hierarchical structure is suitable for most (but not all) types of documents,
- ✓ it manifests as plain text files, unencumbered by licenses or restrictions,
- ✓ it is platform-independent, thus relatively immune to changes in technology.

4 Short ODF description

ODF, short for the OASIS OpenDocument Format for Office Applications, is an open document file format for saving and exchanging editable office documents. It is XML-based file format originally created by OpenOffice.org, and ODF was approved as an OASIS standard on May 1, 2005. A draft for the ISO ISO/IEC 26300 was approved on May 3, 2006. This standard was developed by the OASIS industry consortium.

The purpose is to create an open, XML-based file format specification for office applications, such as text documents, spreadsheets, databases, charts, and presentations (that means DOC, XLS, PPT file formats). The resulting file format must meet the following requirements:

- ✓ It must be suitable for office documents containing text, spreadsheets, charts, and graphical documents.
- ✓ It must be compatible with the W3C Extensible Markup Language (XML) v1.0 and W3C Namespaces in XML v1.0 specifications.
- ✓ It must retain high-level information suitable for editing the document.
- ✓ It must be friendly to transformations using XSLT or similar XML-based languages or tools.
- ✓ It should keep the document's content and layout information separate such that they can be processed independently of each other.
- ✓ It should 'borrow' from similar, existing standards wherever possible and permitted.
- ✓ It must have perfect documentation about its structure and syntax.

If it meets all requirements written above (the last one especially) and so we will be able to work with the format in future (tens of years) we should accept it as a *structured document format*. That means there will be no need for adaptors which provides conversation between proprietal and structured formats.

MS XML still stays proprietal document format unlike ODF. It offers these benefits over MS XML format:

- ✓ More open: No legal constraints, and support from OASIS.
- ✓ Reuse of existing open standards when possible (SVG, Dublin Core, MathML, etc).
- ✓ Higher format fidelity.
- ✓ Friendly to XSLT and other XML-based tools.

It is obvious that ODF should become new structural document format standard for usage in long-term archivation

One objective of open formats like ODF is to guarantee long-term access to data without legal or technical barriers, and some governments have come to view open formats as a public

policy issue. ODF is intended to be an alternative to proprietary formats, including the popular, undocumented DOC, XLS, and PPT formats used by MS Office. Another perceived competitor to ODF is the MS Office Open XML format, which has licensing requirements that prevent some competitors from using it.

Several concerns about MS XML:

- ✓ Doesn't cover all of MS Office (for example, it doesn't cover PowerPoint).
- ✓ Doesn't support some of the advanced MS Office features.
- ✓ Can contain binary objects that depend on MS Office and Windows (e.g. OLE and VBA) and those lack complete documentation.
- ✓ MS did not commit to make future changes to MS XML available to the public, only the current one.

5 Conclusion

Almost all office documents are in proprietary formats of individual office software products. Who wants to work with document of this type is forced to get application (and version in many cases) the document was created with. Adaptors (filters) between different proprietary formats are imperfect, because most of the developer companies are protecting their know-how properly. If company starts using explicit office software, it is forced to henceforth buy products of specified producer only.

The standard was publicly developed by a variety of organizations, and is publicly accessible, meaning it can be implemented by anyone without restriction. The ODF is intended to provide an open alternative to proprietary document formats. Organizations and individuals that store their data in an open format such as ODF avoid being locked in to a single software vendor, leaving them free to switch software if their current vendor goes out of business, raises its prices, changes its software, or changes its licensing terms to something less favorable.

ODF is the first standard for editable office documents that has been vetted by an independent recognized standards body, has been implemented by multiple vendors, and can be implemented by any supplier. This includes proprietary software vendors as well as developers using free software or open source licenses.

When ODF becomes ISO standard, the situation will be much easier for public administration. European Union and the US State of Massachusetts in particular have been examining the ramifications of selecting a document format. It was announced on 31 March, 2006, that the US National Archives had settled on ODF as their choice for a cross-platform/application document format. Other governments around the world are also considering the adoption of the format.

On 23 June 2006, the Belgian federal administration decided to exchange all documents in ODF from September 2008. It is the first and very important step for the whole European Union. All federal administrations should be able to read ODF documents one year earlier. Only ODF is accepted as such a standard in the proposal. Earlier drafts of the Belgian proposal had treated ODF and Microsoft's own Open XML format (which is to be included in Office 2007) on equal footing. Microsoft representatives announced that their new MS Office with its MS XML format won’t support ODF. Anyway ODF Foundation is already working on a plug-in for MS Office that would add ODF support.

References

[1] DECKMYN Dominique: *Belgian government chooses OpenDocument* [online]. ZDNet Belgium, last updated June 23, 2006, c2006, [cit 2006-06-25]. URL: <http://news.zdnet.co.uk/software/applications/0,39020384,39276978,00.htm>

[2] BUDÍNOVÁ Jitka: *VNITŘNÍ ORGANIZAČNÍ SMĚRNICE – OBEC SKÁLY* [online]. Skály, last updated 06.02.2003, [cit 2006-06-26]. URL: <http://www.municipal.cz/skaly/dok_povinne/smernice.pdf>

[3] MANNERHEIM Johan: *Preserving the Digital Heritage of the World* [online]. Last updated 01/2000, c2000, [cit 2006-06-25]. URL: <http://www.hb.se/bhs/ith/1-00/jm2.htm>

[4] *OpenDocument - Wikipedia, the free encyclopedia* [online]. Wikipedia, Florida, USA, last updated 26 June 2006, [cit 2006-06-26]. URL: <http://en.wikipedia.org/wiki/OpenDocument>

[5] RIZK A.: *Comparative assessments of Open Document Formats Market Overview* [online]. VALORIS, Last updated 23.05.2003, [cit 2006-05-26]. URL: <http://europa.eu.int/idabc/servlets/Doc?id=17982>

[6] *Comparison of OpenDocument and Microsoft XML formats - Wikipedia, the free encyclopedia* [online]. Wikipedia, Florida, USA, last updated 26 June 2006, [cit 2006-06-26]. URL: <http://en.wikipedia.org/wiki/Comparison_of_OpenDocument_with_Microsoft_XML_formats>

[7] *Extensible Markup Language - Wikipedia, the free encyclopedia* [online]. Wikipedia, Florida, USA, last updated 25 June 2006, [cit 2006-06-26]. URL: <http://en.wikipedia.org/wiki/XML>

[8] CARRERA Daniel: *OpenDocument Fellowship – Articles - History of OpenDocument* [online]. OpenDocument Fellowship, Australia, last updated 05 December 2005, [cit 2006-06-25]. URL: <http://opendocumentfellowship.org/Articles/HistoryOfOpenDocument>

Kontaktní adresa:

Ing. Rudolf Vohnout
Univerzita Pardubice
Fakulta ekonomicko-správní
Ústav systémového inženýrství a informatiky
Studentská 84
53210 Pardubice
rudolf.vohnout@upce.cz