White Paper

Migration to Intermediate XML for Electronic Data (MIXED)

A strategy in digital preservation – A DANS software project

Data Archiving and Networked Services

DANS

MIXED

MIGRATION TO INTERMEDIATE XML FOR ELECTRONIC DATA

Dirk Roorda working with

René van Horik, Kris Klykens, Rutger Kramer, Laurents Sesink

Summary

MIXED is a digital preservation project. It uses a strategy of converting data to intermediate XML. In this paper we position this strategy with respect to the well-known emulation and migration strategies. We then detail the MIXED strategy and explain why it is an optimized, economic way of migration. Finally, we describe how DANS is implementing a software tool that can perform the migrations needed for this strategy.

Introduction

A fundamental difference between digital objects and most other objects is that the former are not accessible without using a digital computer. Accessing a digital object means running a set of appropriate programs on a suitable computer. The result of this process is output on some device that can be used by man or machine. There are two major problems related to *digital* preservation: storage of the digital objects in question, and accessing the digital objects in the future.

The first task is not very different from preserving non-digital objects. It is all about setting up organizations that preserve holdings, using technology designed for storing those holdings. For holdings on paper it means controlling the environment, so that the paper does not deteriorate; for holdings in digital form it means putting bit streams onto stable media, and refreshing them whenever the inherent stability of the media weakens. By saying that this task is not very different from traditional archiving, I am not saying that it is trivial, much less that it is already solved, or that it can be solved once and for all.

The second task is specific to digital objects. Between the digital object and the human end user is some form of digital processing. Whereas it is conceptually clear what is meant by preserving the digital object, the matter of processing is much more difficult. A digital process is very context sensitive. It relies on:

- The current digital devices (computers, input/output devices, media devices, ...).
- The current operating systems (Windows, Unix, ...).
- The current programming languages (Java, C#, SQL, XSLT, ...).
- The current related applications (databases, web servers, office applications, ...).
- The current standards, de facto and de jure (XML, UNICODE, JPEG, MPEG, PDF, RTF, ODF, OOXML, SOAP, ...).

All interesting software is functionally dependent on at least one of the above environment factors. Hence, software tends to become obsolete fairly quickly. Part of ensuring digital preservation is dealing with software obsolescence.

Contents

Strategies against software obsolescence

There are several viable strategies that overcome the problem of software obsolescence. Two of them have an outstanding intuitive appeal. They are diametrically opposed to each other: *emulation* and *migration*.

Emulation

Emulation is based on the view that archived digital objects should remain immutable. The idea is to preserve the digital processing that is needed to access the objects. One way of doing so is to archive the existing hardware and software, but that has several disadvantages:

- Over time, the collection of archived hardware and software becomes unmanageable. It is impossible to keep alive the knowledge needed to operate these devices.
- Over time, the hardware is no longer produced, so if parts stop functioning, they cannot be replaced.
- It is difficult to aggregate the archived content and make it uniformly accessible.
- Improvements in hardware and software do not apply to current holdings.

In many cases, these disadvantages prohibit a successful deployment of emulation.
An approach that overcomes some of these defects is emulating the hardware and software on present-day computers. Of course, this emulation should be repeatable on 'future-day' computers, and in order to make this feasible, the concept of Universal Virtual Computer has been introduced. In this approach, emulation requires a twofold effort:

- Compile existing data-accessing programs to the target UVC platform for each program you want to preserve.
- Implement the UVC on every platform on which the archive wants to deliver its results.

In this way, one does not have to maintain the physical hardware and the original software; moreover, the translation effort applied to programs is minimized.
Despite the potential genericity of this solution, practical difficulties have so far prevented its widespread adoption. The only cases where it has been adopted are those in which the material to be archived does not admit other strategies. This is all about data where form and content are not clearly separable, and where action is part of the message, for example in the case of electronic education with tests, training sessions or rich multimedia content.
This leads to one of the biggest advantages of this approach:

- The original look and feel of the digital objects is preserved.

Migration

Migration means adapting the digital objects to the ever-changing context. If an archived document that was created by a very old version of a word processor can no longer be opened on a current computer with current software, the document will be converted into a form in which it can be opened. However, here too there are disadvantages:

- The migration frequency is a function of the rate of change of the context. Since the dependency on the context is manifold, and since application software changes rapidly, it is very difficult to keep track of all the migrations that are needed. For example: in 2007 it might occur that Word documents created prior Word 6.0 are no longer openable by the new version of Word. So all pre-1993 documents suddenly qualify for migration.
- Migrating archived material can require a huge amount of work; in addition, it is risky, because it might introduce conversion errors. Often it cannot be done completely automatically. What makes it worse is that one is always converting old documents that are in nearly obsolete formats.

Despite the conceptual clumsiness of this solution, it is the most widely adopted one. It is especially applicable when the form of data is less important than the content, for example in the case of research data, where the interest lies in reusing data for new aggregation and analysis. Here, the original look and feel of data is not important.

Doing nothing

In the light of the disadvantages of emulation and migration, another practice has gained popularity: the null-practice. On other words, leave the material as it is, take no steps against software obsolescence. In many cases this is quite reasonable. Deciding factors are:

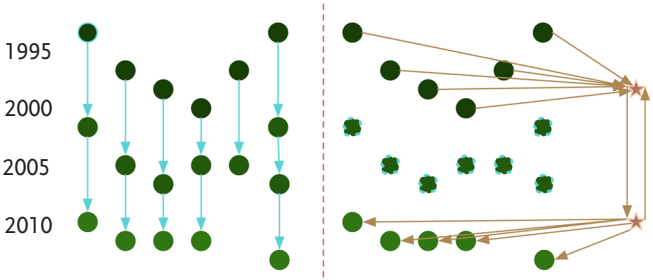
- The material does not need to be accessed for a long time.
- The material is in a more or less self-describing format, or in a well-documented format.
- Future improvements are at hand; for example there is a current trend of standardization in the formats used by the material, whilst this standardization is by no means complete.

The risk is that the data contains obscurities that cannot be resolved later on.

This approach highlights the fact that the question of preservation is always a question of cost. Who will pay for the sustained accessibility of data? More precisely, how is this cost distributed over the generation that creates the data and the future generations that will consume the data? If the present generation has to pay a lot for some preservation measure that can be done at much lower cost in the future, it might be an option to skip that measure now. On the other hand, if we, at limited cost,

can implement a preservation method that obviates enormous costs for future generations, there is strong reason to take that measure.

Towards smart migration



plain migration smart migration

In the emulation approach we saw the Universal Virtual Computer functioning as an efficiency aid to the general emulation approach. Can we do something of the kind for the migration approach? We think that there is indeed a migration booster: migration to an intermediate XML format. The general idea is sketched in the diagram to the left.

The plain migration strategy keeps datasets in the formats belonging to the applications with which they were created, and thus many different datasets follow many different conversion paths. Typically, one dataset will undergo several conversions over time. The smart migration strategy converts all datasets, upon ingest to the archive, into an intermediate, generic format. Upon dissemination of a dataset, it is converted from this generic format into a current vendor format of choice. It is likely that the intermediate format will also change, but at a much slower rate. The optimization is that conversions are split into many contemporary (or *synchronic)* conversions and a few time-bridging (or *diachronic)* conversions. This is a much more manageable situation, because the complexity of many different formats can be dealt with contemporaneously, and the complexity of bridging time can be dealt with by means of one well-defined format.

We are currently carrying out a project to demonstrate the viability of this idea by means of a working solution.

Migration to Intermediate XML for Electronic Data (MIXED)

Nowadays the word '*electronic*' does not have the modern ring that it had, say, two decades ago. *Digital* data may be carried by many media: magnetic, optical, chemical, elementary particles, structures in quantum space, holistic networks, and what not. The key idea is that the data is digital and requires some form of digital processing before it is useful. Nevertheless, the acronym MIXED is slicker and looks smarter than MIXDD, so we have traded some accuracy for style.

The founding observation for MIXED is that applications often use specific, non-standard, application-dependent formats to handle your data. And when such applications stop handling your data, they save your data in their proprietary formats. You find that your data is inside a straight-jacket.

The generic remedy here is: for every kind of application that handles data, find or define a standard format that fits that data. Then express this format in XML, which is the formalism of choice to make hidden structure explicit. Some applications already

work with standardized formats, while others are evolving into ever more standardization. One area in which the lack of standardization is most keenly felt is tabular data, as in spreadsheets and databases. There are several reasons for the acuteness of this feeling:

- There is an enormous amount of legacy material in spreadsheets and databases.
- Spreadsheets and databases are almost as old as digital computing itself, which means that these applications come in many (aged) versions.
- There is a huge discrepancy between the simple, logical nature of tabular data and the convoluted representations of it in applications.

There are other reasons why it is particularly appropriate to subject tabular data to a MIXED-like approach:

- Lots of research data, outcomes of experiments, interviews, surveys, field studies, etc. are in tabular form.
- The importance of this data lies in its content, not in its initial look and feel.
- Current trends in research require the mass aggregation and reuse of data collected in the past.

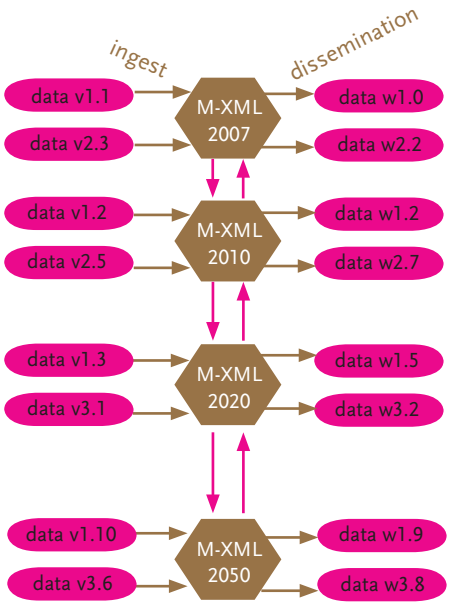
This means that the preferred strategy for preserving tabular data is *migration*. So if we make migration feasible for this kind of data, we contribute to the preservation of research data.

How MIXED optimizes migration

For MIXED we are defining an XML language that expresses the generic structures of databases and spreadsheets. This language – M-XML (pronounced 'mix-em-el') – serves as an interchange format for all databases and spreadsheets (we will discuss limitations shortly hereafter). More precisely: we are developing an XML schema, called M-XML, in such a way that we can express databases and spreadsheets as valid M-XML documents. So M-XML is just another format in which spreadsheets and databases might be encountered. *This format is a non-proprietary representation of the data*. We are building converters from existing application formats to M-XML, and vice versa. On ingest of tabular material into an archive, we will convert the data to M-XML, and on dissemination, we will convert the data from M-XML to any spreadsheet or database format the end user requires.

We expect that, over time, the M-XML format will change much slower than the individual application formats. We expect that we will have to make future changes in M-XML, no matter how comprehensive we think our definition of M-XML is. This is because M-XML has to represent the data contained in spreadsheets and databases, and it is to be expected that there will be new concepts introduced in the functionality of databases and spreadsheets that must be expressed in the underlying file format.

All we will have to do is maintain converters from the applications of the day to M-XML, and vice versa. We will not have to migrate the material once it is in M-XML. This



represents a huge reduction in work compared to the raw migration strategy.

There is another important advantage: having all tabular data in one generic format will facilitate data aggregation from heterogeneous sources, with respect to the applications in which the data was created and has been maintained.

Let us look more closely at what happens, in this strategy, to data over time. On the left are files of data made by various applications, from different vendors and in different versions. These are the files to ingest. On the right are files of data in various applications; these are the files to disseminate. In the middle are the ingested files in the intermediate M-XML format. On ingest, a file is converted from its original vendor format into M-XML. Upon dissemination, a file in M-XML is converted to a vendor format of choice.

Situation in 2007

Suppose a nice XML schema has been established for archival records: M-XML 2007. The data formats of the time, defined by applications of vendors v1 and v2, in versions 1 and 3 respectively, can be converted for ingest by means of MIXED modules. The required data formats, say w1.0 and w2.2 (which are chosen from the currently available applications and versions), can be achieved as well, by means of MIXED modules that convert from M-XML 2007 for dissemination. So far so good.

The *simple view* is that MIXED has achieved its objective, namely that an eternal archival format has been defined and that, as time goes by, only new conversion modules have to be added by MIXED. So, in the simple view, what is the work that MIXED must do?

- Define the schema M-XML 2007.
- Write conversion modules from v1.1 and v2.3 to M-XML 2007.
- Write conversion modules from M-XML 2007 to w1.0 and w2.2.

The *realistic view* is that M-XML 2007 itself will undergo evolution. This is due to:

- Initial weaknesses of M-XML 2007 as a data format.
- The arrival of new applications and versions with new features not supported by M-XML 2007.

Situation in 2010

Here, we think in the realistic view. New applications, versions and features have arrived, for ingest as well as dissemination. The format M-XML 2007 has to be upgraded to cater for the new features: M-XML 2010 is born. The older applications and versions for ingest and dissemination are not yet obsolete. The things to do are:

- Write new conversion modules for ingest from v1.2 and v2.5, and for dissemination in w1.3 and w2.7.
- Write an upgrade conversion from M-XML 2007 to M-XML 2010; this is needed

in order to ingest new material that arrives in older formats: the existing modules convert the material to M-XML 2007, while the new upgrade conversion brings it a step further to M-XML 2010. (Obviously, we could have written new modules to convert from the older formats directly to M-XML 2010, but that looks less efficient).

- Write a downgrade conversion from M-XML 2010 to M-XML 2007; this is needed to disseminate new material in older formats: the new modules convert new material directly from M-XML 2010 to new formats, in order to disseminate it in older formats by means of older modules, the new downgrade conversion brings it to M-XML 2007 first.

It could very well be the case that M-XML 2010 only adds features, so that any document in M-XML 2007 is also in M-XML 2010. In that case, the upgrade conversion is merely an *embedding* of unchanged M-XML 2007 elements in an XML document with an M-XML 2010 header; the downgrade conversion is merely a *projection* of unchanged M-XML 2010 elements into an XML document with an M-XML 2007 header, thereby omitting all M-XML elements that are new in 2010.

Situation in 2020

Still taking a realistic view, we assume that the intermediate M-XML format evolves quite slowly. This is because M-XML expresses the semantic core of data, and not all the frills needed by authoring applications. So here is the next upgrade, and we do all the work we needed to do in 2010. We add a slice to the diagram, consisting of a new set of ingest and dissemination modules, and an upgrade and downgrade conversion between M-XML 2010 and M-XML 2020.

But in a realistic view, an extra complication might occur: suppose that in 2020 the upgrade/downgrade conversions between M-XML 2007 and M-XML 2010 become obsolete. What must we do? It depends.

- Case 1: the software tools on which the conversions rely are no longer supported. Remedy: re-implement the conversion by means of current software tools.
- Case 2: M-XML 2007 can no longer be supported, for one of the following reasons: XML has undergone incompatible changes, UNICODE has undergone incompatible changes, or new software tools stumble over aspects of M-XML 2007. In this case we may assume that the old ingest formats (v1.1 and v2.3) and the old dissemination formats (w1.0 and w2.2) are obsolete as well. The only problem then is that we have archived data in M-XML 2007 that we can no longer disseminate. Solution: migrate all such data from M-XML 2007 to M-XML 2010 or higher (the first version that is not obsolete).

The need to migrate archived data is regrettable in view of the fact that we wanted to avoid any such migration, but:

- The up-conversions are likely to be easy and straightforward.
- The conversion is between open, generic, well-documented formats.
- The conversion is between a single source format and a single target format.

So, in 2020 a new layer is added to the diagram, and the oldest layer is stripped from it.

Situation in 2050

The same exercise has to be repeated: a new layer is added, one or more old layers are deleted. Deletion of a layer necessarily involves migration of the legacy content in that layer to a newer layer.

Evaluation of this scenario

The advantage of MIXED within the migration strategy can be summarized as follows: the many migrations over time are replaced by a set of *synchronic* conversions and a set of *diachronic* conversions. The *synchronic* conversions are between current application formats and M-XML. The effort reduction here is that these conversions do not have to be durable: they are needed only as long as the relevant versions of the associated applications are in general use. The *diachronic* conversions are between successive versions of M-XML itself. These have to be much more durable. The effort reduction here is that this durability is not very costly, since the conversions work between the versions of a single, well-defined, open XML format, where the version differences are likely to be just additions.

It seems that this scenario is a case in which, by implementing inexpensive measures, we save future generations much trouble.

Yet there is further potential for cost reduction. There are emerging standards for office documents, among which are spreadsheets. There are already converters operational between rival open standards (OpenOffice's ODF and Microsoft's Office Open XML) and between current application formats and these standards. If we choose our M-XML prudently, we will be able to utilize a lot of this already performed conversion work. This means that in the realm of *synchronic* conversions we expect that preservation by MIXED can be done routinely with hardly any added cost.

An additional cost reduction is to be expected if we take *limitations* into account. We are not concerned with preserving the initial look and feel of data, and it is a great relief not to be under the obligation to preserve exact form. Moreover, we are not interested in preserving the possibility of data creation and updating. So a lot of the application details in databases and spreadsheets can be ignored. They will not be reflected in M-XML. This means that whole areas of the applications and their associated formats will not have any trace in M-XML. This is another reason why the update rate of M-XML will be much slower than the application update rates.

If we work according this *economic view,* we harvest the power that is already there. In that way we envisage a practical solution for the mass preservation of digital records, especially in the realm of research data. By seeking the connection with emerging open standards we will contribute to current practices that are already strong, but not yet up to user-friendly preservation of archival records. We expect that this approach will lead to a situation in which DANS and other archives will have a practical way to add preservation to their records as a matter of routine.

MIXED as a software development project

Having positioned a new preservation strategy, we are now setting out to build the software tools needed to carry out that strategy. The software will not be able to implement the strategy on its own. In order to implement an archiving strategy, also an organizational process is needed. Moreover, once the software to carry out the MIXED migrations has been developed, the conversion landscape will need to be monitored for new file formats and other developments. Here, we restrict to ourselves to just developing the software, but we keep in mind that the software must be able to accommodate a changing conversion landscape. There are two focuses:

- The core functionality of converting application formats to M-XML, and vice versa.
- Interfacing with data repositories.

We will set up the software in such a way that these two focuses remain separable. It is to be expected that the conversion functionality will also be useful in contexts other than archiving and preservation, so we should not tie this functionality to a repository context.

On the other hand, the main reason for building this software is to upgrade our repositories and those of other organisations with preservation measures. Implementing the MIXED strategy for a repository poses a bunch of requirements:

- The conversions must act on behalf of data producers or on behalf of data managers, which leads to different scenarios and different interfaces.
- The conversions must maintain provenance metadata, which will be repository dependent.
- The conversions must contain extra levels of safety:
  - independent testing of the conversion output
  - preserving the conversion input, and linking the converted output to the input by metadata.

Core functionality

The core functionality of MIXED is performing conversions from application formats to M-XML and vice versa. In order to cater for a wide range of formats, we have to work economically: using and reusing what is already there. Suppose we want to con-

vert the formats *a1,...,an* to M-XML. Also suppose that there are existing conversions from the *ai* to format *c* and *c* is close to M-XML. Then we want to reuse these conversions *ai-c* and only write a conversion *c*-M-XML. As an example, think of the file formats that OpenOffice can handle. Files in these formats can be converted to ODF (Open Document Format), calling a module from OpenOffice. We only have to write a conversion from ODF, which is essentially XML, to M-XML to complete the conversion for all these formats.

Generalizing from this example, we envisage a whole landscape of file formats, with conversions linking them. We call the implemented conversions '*atomic conversions'; these* are the conversions for which a converter exists. If we have file formats A and C, and no converter exists to convert a file in format A to a file in format C, there is no atomic conversion between A and C. But if there is a format B, and there are converters from A to B and from B to C, then we can in fact convert from A to B to C. We call such a path of conversions a '*conversion route'*. In the ideal case, we have an ordinary problem of route planning here. For each conversion task from A to Z, the application plans a conversion route, consisting of a number of atomic conversion steps, leading from A through intermediate formats to Z. After planning this route, the application executes it by calling the appropriate atomic conversions. The planning of routes must be guided by additional parameters on the atomic conversion, specifying:

- Performance indication.
- Additional requirements (e.g. OpenOffice must be installed, or Microsoft SQL Server must be present, ...).
- Restrictions (e.g. does not work with multiple worksheets, cannot handle binary large objects, ...).

Quality

Finding an optimal conversion between a file format and M-XML requires computing an optimal conversion route, with maximal performance, where all requirements have been met in the actual situation, and with minimum restrictions. If we do the planning dynamically, it is quite possible that the actual conversion routes will change considerably over time. The conversion landscape will evolve, new atomic conversions will be possible, and old ones will become obsolete. But how is the quality of the conversion between a given file format and M-XML to be guaranteed?

Our response is not to make route planning dynamic. It will probably need human interaction anyway, because in many cases there will not be just one optimal route. What we can do, however, is make a helper application – a so-called conversion planner – that can be used whenever the conversion landscape has changed. What it will do is find optimal conversion routes with or without user input. Once the user is satisfied, he or she will be able to save the planned routes in a configuration file, from

which plug-ins can be assembled. A plug-in will take responsibility for a complete conversion route, not just an atomic step. So from the point of view of the MIXED program, all routes will be statically known and laid down in plug-ins, and quality will be assured per plug-in. But from the point of view of the MIXED operator, it will be possible to have their routes dynamically planned, tested and documented.

So the final scenario here is: whenever the conversion landscape changes, a conversion expert re-computes all routes, and possibly adds some new ones and deletes other ones. The changes in routes are tested thoroughly and the updated routes are coded in new plug-ins. The documentation of these new plug-ins is created. This documentation includes guidelines for data producers and managers, stating dos and don'ts specific to file formats, and guaranteeing certain results if the guidelines are followed.

Interfaces

Repositories

We are going to make MIXED interfaces for OAIS-compliant repositories. OAIS compliant does not mean that there is a standard protocol in dealing with some repositories. The OAIS standard only states how AIPs (archival information packages) can be manipulated by the management of the repository, how they can be checked out as DIPs (dissemination information packages) and checked in again as SIPs (submission information packages).

In the MIXED software we will make a layer to deal with repositories on the most abstract level, namely that of AIP, DIP and SIP. This layer will be implemented by lower-level layers that actually transport information, using standard protocols, eventually.

Web services

The MIXED service will be a framework with plug-ins. The plug-ins are basically the full conversions from file formats to M-XML and vice versa, but there are also plug-ins for configuration, logging, management and file format detection. The business rules will be expressed in a formalism most suited to expressing business rules, for example BPEL. All these plug-ins and formalisms are connected loosely, using SOAP as the messaging glue between them.

We have chosen an architecture like this to facilitate scaling, coping with change in the conversion landscape and adapting to multiple use cases. Take logging. All the plug-ins emit logging messages if they have something to notify. These messages will be intercepted by the logging module and stored in a database. Each message will be decorated with a process identification, plug-in identification, severity class, verbosity class, and perhaps more. Depending on the wishes of the operators of MIXED, all

kinds of interfaces can be built on the logging database. If you need to tailor logging to your needs, you only have to adapt a logging interface, not the MIXED framework itself.

This architecture lends itself excellently for providing MIXED as a service over the Internet or an intranet, rather than using it as a stand-alone application. But given a web server like Tomcat and a database like MySQL, it should be straightforward to get MIXED to work on a stand-alone computer.

It is the changing nature of the conversion landscape that induces us to set up MIXED and its conversion plug-ins as services. In this way MIXED can incorporate conversion resources that are located and developed anywhere. These services will be quality assured, because the conversion landscape will be monitored and the conversion routes will be optimized ever so often. This is the only burden. Once you have optimized the conversion routes, no extra effort will be required to deploy software updates. It is like the model Google employs for its Internet applications.

Use of standards

The use of standards in MIXED is not just lip service. We are following standards at all levels of programming and organisation provided there are standards available. We are considering not only de jure standards but also de facto standards. Standards make things recognizable to others, and thus facilitate cooperation. They embody best practices and proven design choices, and thus enhance quality. They make explicit what otherwise would remain implicit, and thus aid preservation – which is the core business of MIXED. The following is an overview of standards to be used by MIXED.

XML and UNICODE

MIXED's intermediate format is XML, the configuration files that connect the plug-ins is XML, the open document standards ODF and OOXML are XML, and SOAP is XML.

Every XML document uses UNICODE, one of the biggest blessings of XML. UNICODE defines a universal character set, which does away with all character coding problems. UNICODE itself does not specify bit strings for characters, only numbers. The link between number and bit string is done by encodings. In the western world UTF-8 is the most popular encoding, since it coincides with ASCII in the first 128 characters and nearly with Latin 1 in the first 255 characters. Yet it is a kind of legacy encoding, optimizing the encoding of the first 255 characters at the expense of the transparency of coding all the remaining characters. UTF-16 is transparent, and hence better suited for preservation. Probably UTF-32 is the only reasonable long-term choice of character representation.

OAIS

While OAIS not the only reference model for repositories, it is the most referenced one. It is not a technical standard; it only describes processes from an organizational perspective. It is quite hard to prove that a repository is OAIS compliant. For MIXED we draw on the following parts:

- Positioning of preservation management (task for repository managers, not for data producers).
- Rules about handling AIPs, DIPs and SIPs.
- Requirements for provenance metadata.

ODF, OOXML and UOF

There are several existent and emerging standards for office documents, of which spreadsheets form a subset. These standards are supported by Office applications, and they are more and more interoperable between different vendors. There are lots of good and poor conversions between older binary formats and older interchange formats and these new standard formats. M-XML draws on them for inspiration and details, but will steer an independent course. Probably ODF, OOXML or UOF will be chosen as a starting point; most probably, it will be ODF, as ODF is already a standard, UDF is based on ODF and there are more tools for ODF than for OOXML. OOXML is more tied to MS Office than ODF is to OpenOffice.

JAVA

Java is our language of choice for a number of reasons: well supported, good frameworks, well studied, well employed, good performance and still, after all these years, a centre of development activity.

SPRING

Spring is a well-known, clean framework for binding plug-ins together into frameworks.

ESB

ESB is not a standard but a design pattern. It is a middleware design, in which heterogeneous software modules exchange messages and in that way cooperate with each other. Heterogeneous means: not belonging to the same application, not running on the same computer, not having the same geographic location, not using the same platform, not being programmed in the same language.

SOAP

SOAP is a standard for application interaction by means of messaging in XML. Used by SPRING, SOAP.

BPEL

BPEL is a formalism for expressing and executing the business logic of applications.

Software development methods

Extreme programming

Although our project has strictly planned work packages, within these packages we are working with agile development methods, with many traits from extreme programming.

Patterns

We are designing our software with certain design goals: adaptable to many use cases, extendable by third parties, executable over networks. So it is very important to reduce coupling between subsystems, to separate interfacing, business logic, management and core conversion functionality. We are doing so by carefully selecting the design patterns that have the required characteristics. Whenever it appears that we have misselected, we re-factor our solutions towards the right pattern.

Related projects

DExT

Quote from the DExT website:

> *The project aims to develop, refine and test models for data exchange for both survey data and qualitative research data based on XML/RDF schema and will develop tools for data import and export. The work will also research the feasibility of developing automated conversion procedures for legacy formats. The data formats to be included are those that are commonly used in research such as SPSS, STATA, XML, Atlas-ti, MaxQDA and Nvivo. The test data selected for this project are from the social sciences, but these formats are typically found across all domains of primary research. A small scale evaluation of the models and tools will be carried out in order to inform JISC of the most viable options for future development in this area.*
>
> *A longer-term aim of this work would be to build a fully functional and scalable facility or service where data formats can be submitted and seamlessly returned in a chosen, desired format. The work in this proposal aims to lay the foundations upon which such a sustainable service could be built.*

RODA

RODA is a Portuguese project. Quote from Miguel Ferreira's website:

> *The RODA project is being developed by the National Archives (**Instituto dos Arquivos Nacionais/Torre do Tombo**) and the **University of Minho** and aims at raising awareness within public administration institutions on the issues of digital preservation. From this project will result a digital repository system capable of preserving authentic digital objects. After its conclusion the National Archives will be able to ingest digital objects (e.g. still images, relational databases, text documents) produced by associated public institutions.*

CHRONOS

Chronos is a commercial product from Germany, stemming from SIARD, which is also mentioned here. A quote from their website:

> *Long term archiving for relational data bases: 1. less resources needed with simultaneous increase of efficiency, performance and reduction of costs; 2. open systems archiving: independent from producers of data bases and platforms and nevertheless fast access to data; 3. transparent and high-performance access to archive data and output in any form: excel charts, PDF, HTML, XML; 4. time independent archiving by open archive format, data type library and archiving of meta data; so old archives are still readable after years and easy migration of data is guaranteed.*

SIARD

SIARD is a completed project for the Swiss Federal Archives. A quote from one of the available papers:

> *We discuss long-term preservation of and access to relational databases. The focus is on national archives and science data archives which have to ingest and integrate data from abroad spectrum of vendor-specific relational database management systems (RDBMS). Furthermore, we present our solution SIARD which analyses and extracts data and datalogic from almost any RDBMS. It enables, to a reasonable level of authenticity, complete detachment of databases from their vendor-specific environment. The user can add archival descriptive metadata according to a customizable schema. A SIARD database archive integrates data, datalogic, technical metadata, and archival descriptive information in one archival information package, independent of any specific software and hardware, based upon plain text files and the standardized languages SQL and XML. For usage purposes, a SIARD archive can be reloaded into any current or future RDBMS which supports standard SQL. In addition, SIARD contains a client that enables 'on demand' reload of archives into a target RDBMS, and multi-user remote access for querying and browsing the data together with its technical and descriptive metadata in one graphical user interface.*

References

See the references page of the MIXED website at **http://mixed.dans.knaw.nl/node/4**

Project name: MIXED (Migration to Intermediate XML for Electronic Data).

Project leader: Dirk Roorda; Dirk.Roorda@dans.knaw.nl; +31 6 13665023.

Organization: Data Archiving and Networked Services (DANS); http://www.dans.knaw.nl; Anna van Saksenlaan 51; 2593 HW Den Haag, The Netherlands.

Funding: Ministry OCW (program PRIMA), and DANS.

Financial control: SenterNovem www.senternovem.nl

Project site: http://mixed.dans.knaw.nl/

September 2007