# A blueprint for Representation Information in the OAIS model

**David Holdsworth, Derek M. Sergeant**
The Cedars Project (ISS department)
The University of Leeds
LS2 9JT, UK
D.Holdsworth@leeds.ac.uk, D.M.Sergeant@leeds.ac.uk
tel +44-113-233-5402, +44-113-233-5698
fax +44-113-233-5411

**Abstract**
The CEDARS[*] project within UK academia seeks to develop a demonstrator system to recommend techniques for long-term storage of digital data primarily within the research library context. The intention is that this demonstrator will conform to (the spirit of) the OAIS (Open Archival Information System) model [1], and put some flesh on the model's generic bones. This paper describes our current scheme for representation information.

## 1 Overview

We are firmly of the opinion that data outlives the medium upon which it is recorded. Technology obsolescence is a bigger threat to data retention than is media drop-out. For example, failure to read a reel of 7-track magnetic tape is more likely to be due to non-availability of the tape drive than drop-out on the medium.

However, a bit-stream can be preserved indefinitely [2,3].

Our blueprint provides for transforming of a digital object into a bit-stream preserved indefinitely, and for subsequent access to the intellectual content of the original digital object. We believe that the approach is valid for plain ASCII (American Standard Code for Information Interchange) text files, or for emulation of preserved computer systems, and for a whole host of situations in between.

## 2 Ingest

We propose a 2-stage process of ingest which has the steps:

- First step is separation of the data from the medium
- Second step is to map to a bit-stream (i.e. make the *data-object* part of the AIP).
- Followed by preservation of the bit-stream (i.e. keep the AIP in an archival store).

The form of the data between steps 1 and 2 is the *underlying abstract form* (UAF - see below). These two steps are the parts of *ingest* process that involve the data itself. We find that the UAF concept enables us to structure the representation information, and to build representation nets which facilitate evolution with emerging technology.

This paper does not address the other meta-data aspects that must also comprise part of the ingest process.

[*] CURL (Consortium of University Research Libraries) exemplars in digital archives

## 3 Information Package

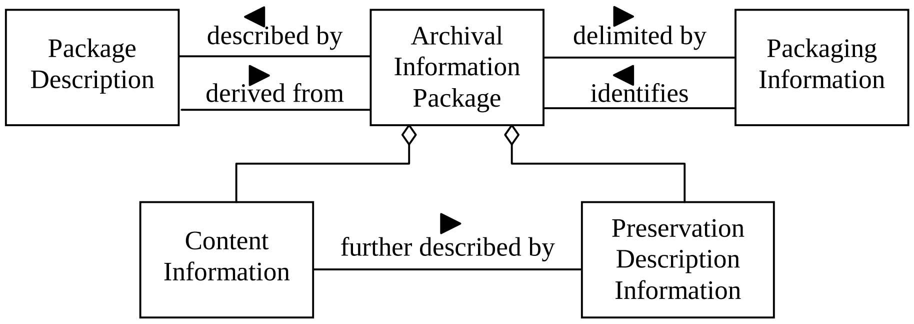

Figure 1. OAIS fig4-16: Archival Information Package (AIP)

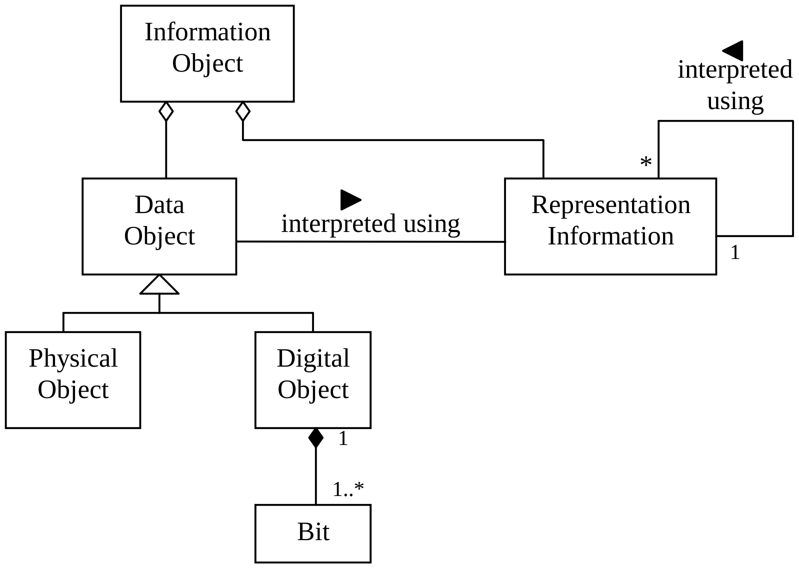

Figure 2. OAIS fig4-9: Information Object

We are concerned here with generating the representation information for inclusion within the content information component of an archive information package (AIP) (see fig 1 and fig 2 taken from the OAIS model). Each component of the AIP is packaged in a formalism appropriate to the data being stored, using an ASN.1 (Abstract Syntax Notation) wrapper to package the components together as an AIP:

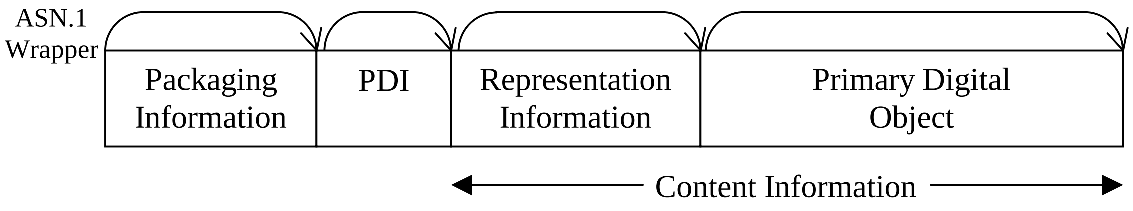

Figure 3. The package structure of an AIP

The packaging information is a CRID (Cedars Reference ID) (discussed later in this paper) which provides a version number for the internal structure of the AIP and references a specialised AIP whose purpose is to document the data format description (DFD) of each component. For example this latter AIP would provide an XML DTD for the PDI (Preservation Description Information) component, and a specification of the Java Properties used in the RI component.

The purpose of the *representation information* (RI) is to enable access to the preserved digital object in a meaningful way. Java property files are a simple formalism to store the information needed to do this. Their use integrates well with our demonstrator software written in Java, and yet the format is simple enough to admit of a brief description. As the Cedars' demonstrator evolves the RI will be migrated into an XML formalism which provides hierarchical structuring and also integrates well with our Java implementation.

Enabling meaningful access to the preserved object includes such processes as recreating the experience of viewing (or even interacting with) the original, or analysing it for a concordance. For particularly complex objects, emulation is likely to be involved.

At its most basic, the RI enables a reversal of the ingest process to deliver a copy of the original. This observation provides a useful minimum criterion for testing the acceptability of any scheme for RI and also for the RI of any particular information object. If this test is performed alongside the ingest process then a critical comparison of the copy produced by ingest reversal and the original can be made. This suggests that the standards for ingest should require that:

> The representation information must allow the recreation of the significant properties of the original digital object, if one assumes that appropriate hardware technology is available.

## 4 Significant Properties

Whoever takes the decision that a particular digital object should be preserved will have to decide what properties are to be regarded as significant. The submission agreement could usefully specify a list of significant properties. In a library context, the decision to preserve is taken by the collection management activity [4,5,6].

## 5 Underlying Abstract Form

We use the term *underlying abstract form* (UAF) to encapsulate the recognition that the data has an existence and a content separate from the medium upon which it is written. This underlying abstract form contains all the significant properties of the data, and is independent of the medium upon which the data is written. Any given digital object is likely to have a number of possible UAFs. Choice of the UAF for preservation is part of the ingest process (either in the *Receive Submission* box or the *Quality Assurance* box in fig 4 taken from the OAIS model). (There may be some difficulties here for the case of multimedia data objects, but we put them aside for the moment, so as better to develop the concept.)

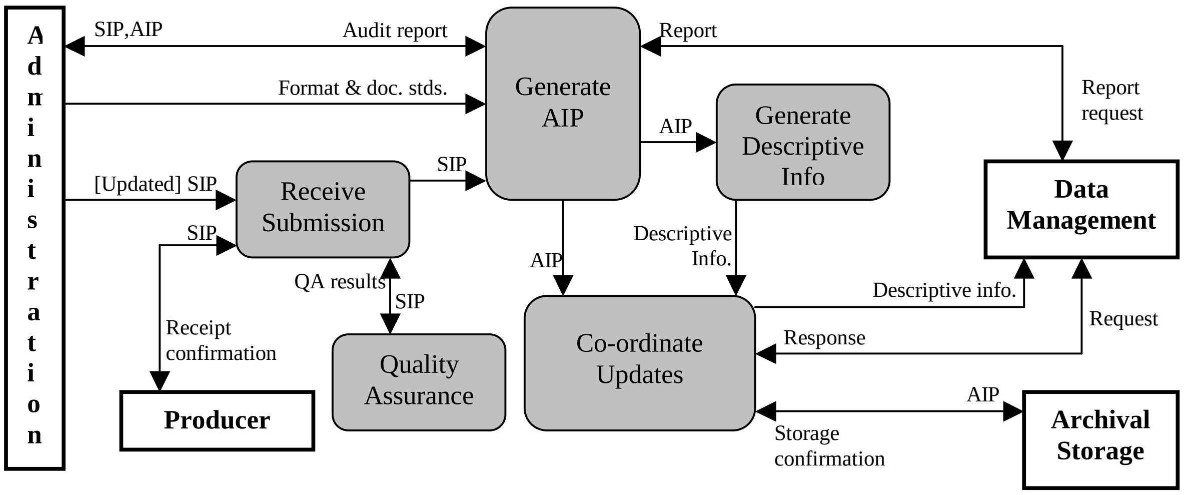

Figure 4. OAIS fig4-2: Functions of Ingest

Some examples:

- Many CD’s actually contain a file system, and successful operation only relies on that file system. Copying such a file system onto a partition on a hard disk delivers an equivalent working representation. File placement is unimportant. Thus the file system is a viable underlying abstract form.
- In some cases it is only important to have a file tree, and the CD contents can be copied into a directory within an existing file system.
- Data held in a relational data-base can equally well reside in a variety of data-base engines, and still deliver its original content. The comma-separated files holding the content can be used as a system-independent representation of that content.
- A plain text document consisting of lines of characters drawn from the ASCII character set is meaningful in a variety of environments. Internet RFC’s (Request For Comments) are typical of such documents.

*Access* involves realising the UAF on the technology appropriate to the time of access in such a way that the desired form of access (which may not necessarily be viewing) can be achieved. If we take the simple example of the Internet RFC, the same UAF is stored slightly differently on a UNIX system from a PC file system, because of the different conventions for line termination. However, the UAF of lines of text is the same in each case, and the UNIX `cat` command clearly displays the same information as the PC’s `NOTEPAD`. The same lines of text represented in an EBCDIC (Extended Binary Coded Decimal Interchange Code) system would be represented very differently in terms of binary digits. The same data would be displayed in the same form. The underlying abstraction of lines of text is the same, but the different platforms represent the information differently internally.

Fig 5 illustrates the way in which we perceive data being preserved and accessed. The original object is identified as having a particular underlying abstract form, and this is used in the creation of a bit-stream for indefinite preservation. Software access to the information will be via some API (Application Programming Interface), which may go through several layers of abstraction before mapping onto the original data via its underlying abstract form. Two such layers of intermediate abstraction are illustrated by the empty ovals in fig 5.

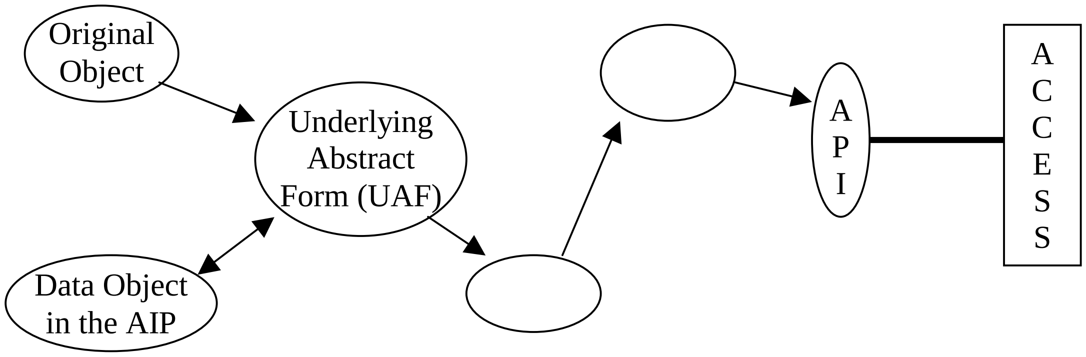

Figure 5. An Access path to the Preserved Data Object

## 6 Role of Representation Information

Clearly, the primary (sole?) purpose of RI is to enable access, i.e. to provide the "road map" for selection of a route such as depicted in fig 5. With this model of access via the UAF, the RI has two distinct roles.

1. enable generation of the UAF from the dissemination information package (DIP) (roughly equivalent to the AIP for those not familiar with the OAIS model), and

2. enable exploitation of the UAF on an available (and suitable) platform.

By *platform* we mean some computational facility which is capable of storing a representation of the UAF, and offers capabilities for running necessary software for accessing the intellectual content of the data in the desired manner.

A digital object may be configured for a variety of platforms (e.g. Many CD's will work with both MAC and PC), and the chosen UAF may well encapsulate this. It is up to collection managers to decide whether it is a significant property of the original digital object. The technology should give them that option.

## 7 Specifics concerning RI and UAFs

Fig 6 (taken from the OAIS model) splits the RI into 2 components corresponding with the division given above:

- **Structure Information** (StI) provides the information necessary for generating the UAF from the bit-stream that is the preserved data object.

- **Semantic Information** (SeI) which is usage information from the original release, modified to remove media dependency. There may be multiple entries in this, corresponding to multiple platforms.

- The UAF provides the formalism with which the horizontal line in fig 6 *adds meaning.*

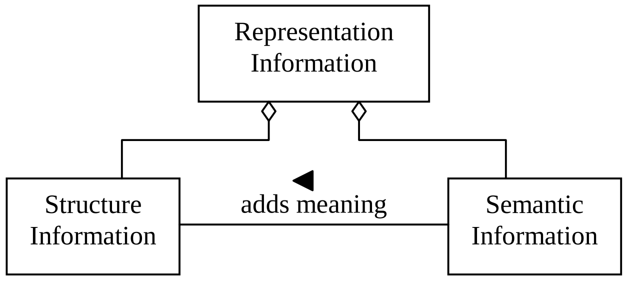

Figure 6. OAIS fig4-10: Representation Information Object

We gave examples of possible UAFs above. For many digital objects there is a set of possible abstract forms, and the choice of one that might be said to be *underlying* is not always trivial.

We propose a policy of choosing the highest level abstraction that discards no significant information. In a library context, the issue of what constitutes *significant information* is a collection management issue. The following examples illustrate the issue:

- A single PDF file held on a diskette can be considered as a file system, a file tree, a byte-stream, or a PDF file. It is our contention that the PDF file is the most useful UAF, as there is real prospect of finding several platforms that can make sense of it for decades to come.
- A set of PDF files might be treated as a set of objects, each of which has the above UAF, or we can decide that a set of PDF files is a *valid UAF*.
- A set of PDF files with a plain text READ.ME file, and perhaps a copy of the Acrobat reader on a CD raises more questions. The UAF is a file tree, but it is useful to include in the RI the information that a set of PDF files is incorporated within that file tree.

In introducing the notion of a *valid UAF* we imply that the management of the archive keeps track of all UAFs -- see below on *Gödel Ends*.

## 8 Platforms

A platform is thought of as a computational facility, though in extreme cases, it may be a human being reading plain text.

Dissemination involves delivering the UAF (or possibly something derived from it) in a form suitable for storage on the chosen platform. Better still, is the ability not only to store the UAF, but actually to render it (audio-)visually or to analyse it. The representation information in the dissemination package (RI in the DIP) provides access software, or information on which access software to use. Our proposals for management of Gödel ends (see below) address the issues of hardware and software obsolescence (referred to in OAIS section 4.2.1.3.2).

A platform might be:

- Browser
- SQL database
- PC file system
- application software (e.g. WordView)
- Khoros
- HDF software
- a very simple text viewer such as NOTEPAD
- or one of an ever growing number of options.

The archive information package (AIP) must hold in its RI all the information necessary to produce platform specific RI for inclusion in DIPs.

## 9 Names for Archived Objects

For the purposes of the CEDARS demonstrator we are allocating each object a Cedars Reference ID (or CRID). The CRIDs are used to form the links in the representation nets. We expect that a CRID will eventually be a URN (Unique Resource Name) [7,8], but at the present we have a simple name resolving technology using a simple server bound onto the socket port “crid:6386”. The machine name “crid” can be translated locally by the hosts file, or by the local DNS (Domain Name Server), so that the locations of objects need not be fixed, and local copies can be accessed preferentially.

The OAIS AIP-id is probably the tail end of our expected URN. Our CRID has a component indicating the participating member of the archive federation, followed by an identifier allocated by the archive that creates the AIP.

More detail of the architecture for the CEDARS archive is given elsewhere [9].

In summary, the architecture has a number of archive stores, each of which is fronted by a gateway with a Web interface. The gateway organises access to meta-data, and organises delivery of preserved objects, subject to the satisfaction of access rights. All reference to a preserved object goes via a CRID name-server which then redirects the request to the appropriate gateway. This structure allows the evolution over time, as preserved objects may well be transfered to new systems, getting a new URL while retaining their original CRID.

In Fig 7 the numbered lines show the stages involved in the retrieval of an object, from resource discovery to delivery. The first stage shows the Web interaction between the search engine and the end-user’s Web browser. Then an HTTP call is sent to the name-server (stage 2) where the object’s CRID is translated into the URL for the gateway of the desired object (this redirection occurs very quickly). The end-user then interacts with the gateway using their Web browser (stage 3). Once the manifestation and platform for delivery are selected the gateway responds to the end-user (stage 4), these could be instruction to use FTP, or await a package delivered by ordinary mail, it could also be a continued Web interaction with the browser providing the object-specific access facility.

The final stage is the delivery of the digital object (stage 5 shows the process by which the object, in the form of a DIP, is delivered and interpreted by the end-user). The remaining (un-numbered) lines show management data paths, especially with regard to meta-data.

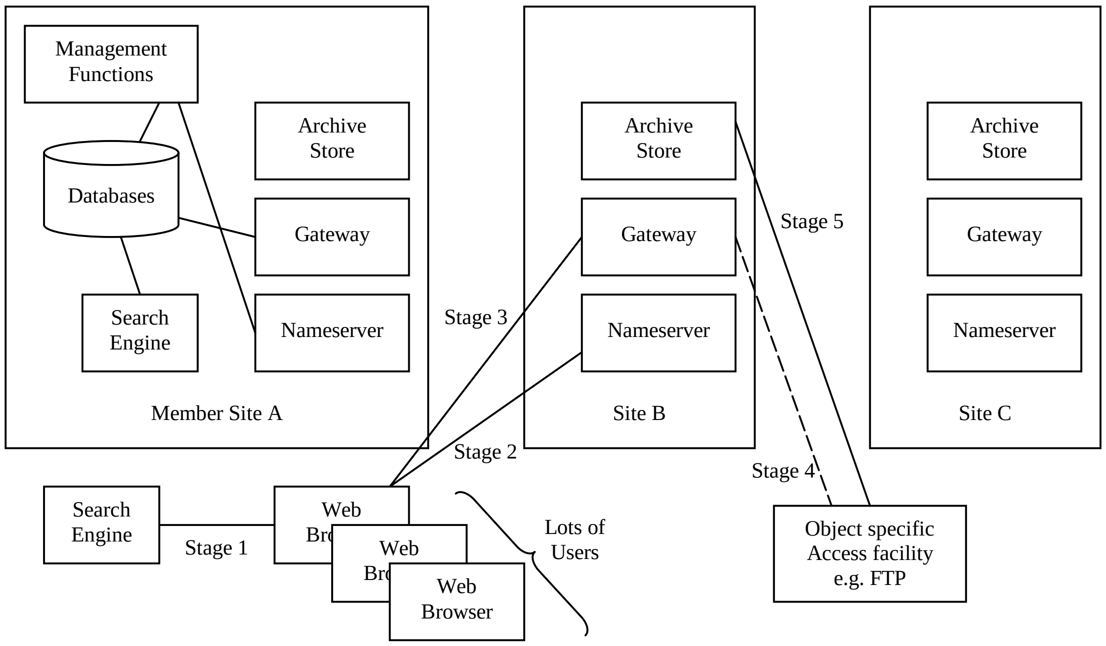

Figure 7. The Cedars Archive Architecture

Each member of the archive federation is seen as running a name-server, gateway and archive store, each with its management functions and databases (only shown for Site A for brevity). In fig 7 a three member federation is shown. Search engines may be independent or linked with the archive management.

## 10 Representation Nets

We are building a system of representation nets, and have implemented a tool that can be used for browsing such a net. In this section we use our own acronyms so that anyone who wishes to challenge our assertions of OAIS compliance has a language in which to do so.

Our representation nets involve 3 main types of node:

- **Data Format Definitions DFD** which define a data format, which is sometimes actual bytes and sometimes a more abstract entity such as an API. There is also a DFD for describing a real magnetic tape, or a human being interacting with a desktop WIMP interface.
- **Render/Analyse Engines RAE** take in data in one format, and deliver it in another. Deliver is interpreted very loosely, in that the output may take any of the forms described in a DFD.

- **Platforms** are typically computer systems, usually with storage, and are necessary for the execution of RAEs. They must contain or have access to storage suitable for storing the data that they are processing.

A *Data Format Definition* includes 2 lists of *Render/Analyse Engines*. One list enumerates those RAEs capable of accepting the defined format as input, and of delivering another format as output. The other list enumerates those RAEs capable of delivering the defined format as output.

An Underlying Abstract Form is a specialisation of DFD that includes RAEs that can accept the raw byte-stream of the *primary data object* as input and deliver the defined format as output.

Example:

> If a filesystem has been preserved as a tar file, it could be described as having a UAF of a filetree. Its UAF object could include a link to an RAE describing UNIX tar for generation of the UAF on a UNIX filesystem, and also an RAE describing WinZIP for generation of the UAF on a PC filesystem.

For any AIP the RI has two components at the top level, StI and SeI.

StI, the structure information, is concerned with describing (and especially regenerating) the UAF and consists of:

- CRID of the *UAF object* (i.e. a DFD including RAEs for building the UAF from the preserved data object).

- any parameters needed by the UAF object

SeI, the semantic information, is concerned with interpreting the UAF, and consists of a list of CRIDs of *render/analyse engines* (RAEs). With each CRID is held the parameter values needed by the particular RAE. Each RAE contains information (often in the form of software) for some particular processing of this AIP. In particular, the RAE includes a reference to the platform upon which it is to operate.

In some (many?) cases SeI may be null, and rely entirely on rendering facilities accessed via the UAF description. This depends on the extent to which the UAF is defined as a high level abstraction.

If we have a multimedia object, such as postulated in fig 8 taken from section 4.2.1.3.2 the OAIS model, we see the *multimedia mapping rules* as combination of the unpacking of the file-tree, and the association of file extensions with different data formats, which are themselves the references to representation information. The *multimedia operations and relationships* describe how to combine the different elements of the multimedia object to produce the intended rendition. A more formal example is given in the appendix.

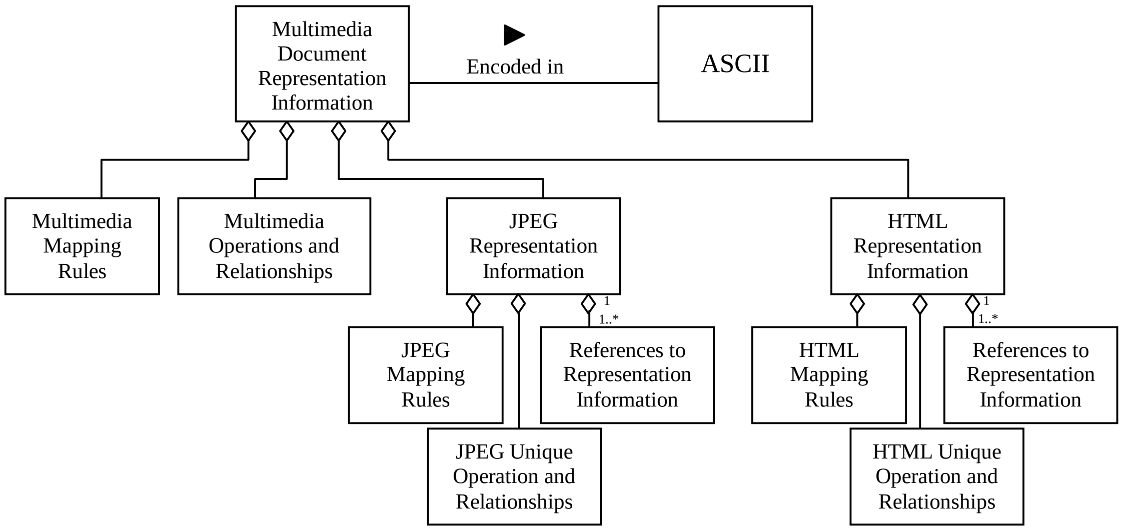
Multimedia Document Representation Information
Encoded in
ASCII
Multimedia Mapping Rules
Multimedia Operations and Relationships
JPEG Representation Information
HTML Representation Information
1
1..*
JPEG Mapping Rules
References to Representation Information
HTML Mapping Rules
1
1..*
References to Representation Information
JPEG Unique Operation and Relationships
HTML Unique Operation and Relationships

Figure 8. OAIS fig4-12: Example Representation Network

We see a value in attaching rendering capability at the data format level, because new facilities can be recorded at the data format level, and immediately become relevant to many preserved objects. This is particularly important in regard to following technological evolution. Our approach, means that the *references to representation information* of figure 9 (duplicated from the OAIS model), may be achieved via an indirection through an RAE with the capability to process the particular format.

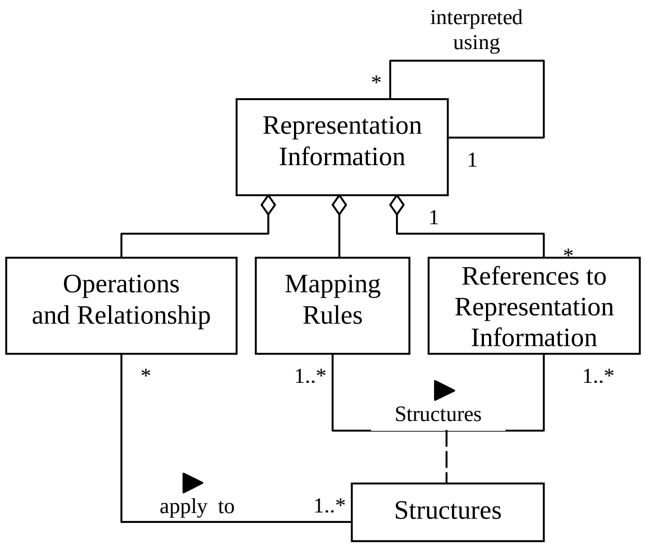
interpreted using
*
Representation Information
1
1
*
Operations and Relationship
Mapping Rules
References to Representation Information
*
1..*
1..*
Structures
apply to
1..*
Structures

Figure 9. OAIS fig4-11: Representation Network Object Model

## 11 True MultiMedia Objects

A CD with video and audio tracks, using the specific properties of a CD drive poses significant problems. The UAF would appear to be the actual spiral stream of bits on the CD, and the long term prospects for rendition seem to involve driver software that emulates this stream of bits on some other medium. It may be technically laborious, but the issue does not invalidate our approach.

## 12 Gödel Ends

As Gödel’s theorem tells us, any logical system has to be incomplete. There must be truths which the system cannot itself deduce.

The representation nets must have ends corresponding to formats that are understood without recourse to information in the archive, e.g. plain text using the ASCII character set, the Posix API. All references to such a format must be via the same CRID, and the management of the archive must have an inventory of such objects. As a format becomes obsolete, the object referenced by the CRID can then be updated to permit understanding of the obsolete format in terms of current practice.

In our scheme, the *platforms* upon which the *render/analyse engines* (RAEs) run are the things that become obsolete as a result of the march of technology. A data format becomes less and less accessible as the platforms of the relevant RAEs become obsolete. As a long stop there is the documentation of the format, which we might consider as a special RAE whose platform is human (e.g. a programmer). In the case of proprietary formats we may not have this information.

Thus the platforms (e.g. Win32) are the things that are outside the archive, and as such are the true Gödel ends of the system. As Jeff Rothenberg [10] has pointed out, emulation of the original computational environment, gives the very best hope for recreation of the experience of a preserved digital object. This can be a laborious process (see Emulation below), and technology shifts may render a true recreation impossible. It is quite possible that the technology of the time actually limited the access to the intellectual content, and a far better access to archived material can be achieved by implementing viewers that operate on the obsolete format directly.

We therefore conclude that the archive administration keeps an inventory of the platforms which occur in the archive’s representation nets, and that the Gödel end platforms each contain the IDs of the data formats that rely on that platform. The administration needs to keep this inventory under review, and to be open to the possibility of adding extra RAEs to data format objects in order to maintain accessibility.

Some platforms may not be true ends, as they may have been realised by emulations. An emulation is a type of RAE, and as such depends on a platform. This gives us a facility for representing a chain of emulations on the Rothenberg model.

## 13 Proprietary Formats

Any archive has an understandable reticence about keeping data which is held in undocumented formats. However, *reticence* must not be seen as a synonym for *rejection*. A current proprietary format may not have any publicly available documentation, but may have readily available render facilities (e.g. WordView) on current platforms. As obsolescence threatens, the commercial value of the documentation is minimal, and there is real prospect of adding it in later.

For instance, anti-virus companies have already reverse engineered the specifications of Microsoft formats.

## 14 Emulation v Format Conversion

We are led to the view that emulation does not always give the most useful rendering of preserved information.

We have looked in particular at two early systems:

- GEORGE3, a mainframe system common in the UK in the 1970s, running on ICL 1900 series machines, and

- The BBC-Micro computer, produced by Acorn and prevalent in UK schools and colleges in the 1980s

Several emulators for the BBC-micro are available over the Web, and provide a good platform for running much of the educational software that exploited the machine's facilities. The machine also had an early word processor (called VIEW) in which some quite large documents were produced (included the system's manual). Although one can run VIEW in emulation, a better access to the information would be achieved if it were to be converted to a more modern word-processor format.

Emulation of the GEORGE3 system is of a different character. The 1900 machine was character oriented with a 24-bit word. The system had a hierarchical filestore with integral management of a tape library. Our emulation of the system creates quite a lot of the feel of operating the real machine (although the modern PC screen fails to generate the ambience of the teletype operator's console, or the buzz of the 2000 cards-per-minute reader). Our CEDARS preservation of this system is a composite object, and holds its tape library. It can also function as a platform for other tapes from that system.

The source text of GEORGE3 was held on 2 magnetic tapes. In the real system this was a cumbersome object, and searching it was rarely a quick process. However, the tape format is not very complex, and a program to render these tapes as an ASCII file was easily written, and forms a much more convenient RAE for these tapes than does the GEORGE3 emulation platform.

On the other hand, the ICL 1900 was the platform for the world's first Algol68 compiler, and access to the intellectual content of this is at its best on the emulated 1900 platform.

We therefore argue that access to the intellectual content should take place at an appropriately chosen level of abstraction. Where emulation is appropriate, it is a powerful technique, with great potential for historical reconstruction and nostalgia. Elvis Presley on CD sounds better than on a 1956 78rpm wind-up gramophone.

We are trying to gauge the effectiveness of our preservation techniques, by pretending that they were available 15 years ago, and then using them on the material of that time. Providing access to the intellectual content of this material on today's hardware is enabling us to assess the merits of different emulation approaches.

## 15 Emulation Practicalities

The GEORGE3 operating system functions at different levels.

- The underlying hardware, where different members of the range have different interfaces to peripherals
- upon which is run the EXECUTIVE program, which implements
- the GEORGE3 executive interface with a uniform set of system calls for driving peripheral devices, upon which is run
- GEORGE3 itself which controls all the resources and provides various facilities for end users and operators, and implements
- the applications program interface (API), in which there are file access facilities and other features used by
- applications programs -- the ultimate *raison d'être* of the computer system.

Our emulation has taken place at the interface between GEORGE3 and EXECUTIVE, although some limited emulation has also taken place at the API level.

In terms of today's PC material, it may well be that the best future access to intellectual content is achieved by emulation at the BIOS level, whereas the Windows API may be a better level of emulation in other cases. We doubt that emulation of the raw hardware under the BIOS will ever be valuable, beyond study of the BIOSes themselves.

## 16 Preservation Format

The above discussion of emulation as opposed to format conversion raises the question of what format to use for long-term preservation. Our approach is to preserve something close to the original digital object (which we have called the *underlying abstract form*), and to have representation information that gives access to the ability to convert yesterday's obsolete format into whatever form is appropriate for the required form of access.

We attempt not to pre-judge the nature of future accesses, and believe that retention of original data maximises the options open to future researchers. We believe that our benchmark of *recreation in principle of the significant properties of the original* addresses this need (see section 3 above).

We readily accept that this criterion is not the same as recreation of the original, but only its significant properties. The discipline of deciding at ingest time what these properties are provides a focus for selecting the preservation format, and aids in the generation of really meaningful representation information.

This accords well with our decision in 1991 that in preserving the data from VM/CMS on our Amdahl system we should retain the data in EBCDIC and provide a converter utility, rather than convert to ASCII as we moved onto UNIX and NetWare platforms. We can now render the full EBCDIC character set using UNICODE, an option that would have been lost by any conversion to ASCII.

## 17 Conclusion

We believe that we are well on the way to a functioning demonstrator, adhering to the overall architecture of the OAIS model, and with useful example material showing possible directions for Representation Information within such an archive.

Key to our approach is the identification of the significant properties and the correct underlying abstract form. We contend that representation information should contain reference to appropriate rendering software. On the other hand we suspect that data whose format is only described textually may well be ignored by future generations because of the labour involved in achieving meaningful access to its intellectual content.

## 18 Acknowledgements

This work was conducted as part of the Cedars project which is funded by JISC/e-Lib programme [11,12] and has three lead sites, the universities of Cambridge, Leeds, and Oxford.

Colleagues within the CEDARS project are addressing issues of Preservation Description Information.

## References

[1] Consultative Committee for Space Data Systems. Reference Model for an Open Archival Information System (OAIS). (CCSDS 650.0-R-1, Red Book, 1999) (http://ssdoo.gsfc.nasa.gov/nost/isoas/ref_model.html),

[2] D Holdsworth. The Medium is NOT the message OR Indefinitely long-term file storage at Leeds University. (Published in: proceedings of Fifth NASA Goddard Conference on Mass Storage Systems and Technologies, NASA publication 3340, 1998)

[3] J Lindner. Toward a MediaLESS Archive. (proceedings of Paris 2000, 5th Joint Technical Symposium, Image and Sound Archiving and Access : the challenge of the 3rd Millennium).

[4] C Jenkins (editor). Collection Management in Academic Libraries. (Particularly chapter 9 by N Elkington) (1999)

[5] S Lee. Oxford, UK. (http://info.ox.ac.uk/localnet/dwp/priority.html) Answering the question of collection management decisions.

[6] N Beagrie and D Greenstein. A Strategic Policy Framework for Creating and Preserving Digital Collections. (1998) (http://ahds.ac.uk/manage/framework.htm)

[7] R Moats. URN Syntax. (RFC 2141 – 5/1997) (http://sunsite.icm.edu.pl/pub/doc/rfc/rfc2141.txt)

[8] A E Walsh. Draft : URN Namespace Definition Mechanisms. (10/1998) (http://www.web3d.org/WorkingGroups/media/hypermail/1998/0175.html)

[9] D Holdsworth. Draft: Proposed Architecture for CEDARS demonstrator (1998). (http://gps0.leeds.ac.uk/~ecldh/cedars/architecture.html)
[10] J Rothenberg. Avoiding Technological Quicksand: Finding a Viable Technical Foundation for Digital Preservation. (01/1999). (http://www.clir.org/pubs/reports/rothenberg/contents.html)
[11] Two web sites for the Cedars Project. (http://www.curl.ac.uk/projects.shtml) (http://www.leeds.ac.uk/cedars )
[12] K L Russell and D M Sergeant. The Cedars Project: Implementing a Model for Distributed Digital Archives. (RLG – Diginews 3:3, June 1999) (http://www.rlg.org/preserv/diginews/diginews3-3.html#feature)
[13] D Holdsworth and D M Sergeant. Representation Network Example. (http://www.leeds.ac.uk/cedars/blueprintAppendix.html)

## Appendix

Here is an example of one of our AIPs [13], with the data object holding a PDF book and a colour GIF image of the original front cover. This AIP exists in our archive store and also has an access page on the gateway. The PDI component of the AIP (see fig 3 in the main text) is only a title and the object's CRID, which is the minimum necessary for our exploration of representation nets. The CRIDs are the arcs of the representation net.

The contents of Packaging Information are not shown.

| PDI | RI | PDO |
|---|---|---|
| CRID=01trav<br>Title=Tigers | StI=↑31fset<br>SeI={ ↑51pdfr(tiger.pdf),↑52gifr(cover.gif),↑53webb } | BitFile001=tiger.zip |

The structure information (StI) in the RI is referenced indirectly (i.e. via a CRID for) and is an AIP which describes the data format (in this case a set of files) and facilities for interpreting it. The semantic information (SeI) in the RI is a list of CRIDs for RAEs that can give access to the intellectual content.

The StI has the following specialised fields in its data object: UAF, a description of the underlying abstract form; TOI, a *transformer object instance* (a special RAE which can generate the UAF from the preserved byte-stream); RAE, a list of render analyse engines that can generate other representations (or provide APIs).

| PDI | RI | PDO |
|---|---|---|
| CRID=31fset<br>Title=FileSetUAF | StI=↑32uafp<br>SeI={ ↑50gate,↑54asciir } | UAF=↑71fset RAE=↑80flist<br>TOI= {↑81pczip,↑82maczip,↑83unixzip } |

The representation net node ( ↑31fset ) conveying information on how to deal with the UAF of the "Tigers" AIP also carries RI to interpret its own digital object component. The object referenced by ↑32uafp conveys this information, and only deals with the UAF specialisation of a DFD (see section 10). Although two choices are provided to render our fileset AIP in the SeI, while the Cedars gateway ( ↑50gate ) can trace through the internal indirections of the representation net and present an overall decision framework, the ASCII renderer ( ↑54asciir ) can only display the Java property file in the PDO verbatim.

| | | |
|---|---|---|
| PDI CRID=71fset Title=filesetDesc | RI StI=↑33asti SeI={ ↑54asciir } | PDO Fileset_description.txt |

| | | |
|---|---|---|
| PDI CRID=81pczip Title=PCunzip | RI StI=↑34raep SeI={ ↑50gate,↑54asciir } | PDO Platform=↑91pc engine=winzip.exe Params=extract OutForm=↑72ftree |

Examining the contents of the fileset AIP ( ↑31fset ) the UAF field references an AIP containing an ASCII description of a fileset ( ↑71fset ), whose RI shows how to render the ASCII file. The RAE field provides access to a tool ( ↑80flist ) that displays the structure of the UAF, in this case by listing the filenames (tiger.pdf and cover.gif) which make up the set. The TOI list field provides methods for transforming the content file of the "Tigers" AIP into these two files on different computational platforms. Only the RAE for the PC platform has been shown (the others are similar). This RAE ( ↑81pczip ) has a dedicated StI (similar to ↑32uafp). ↑91pc references an AIP containing an ASCII description of the PC platform, and ↑72ftree is the DFD for a filetree.

| | | |
|---|---|---|
| PDI CRID=32uafp Title=uafUAF | RI StI=↑32uafp SeI={ ↑50gate,↑54asciir } | PDO UAF=↑73uafpf RAE=↑80flist TOI= {↑84asciicp } |

| | | |
|---|---|---|
| PDI CRID=51pdfr Title=PDFreader | RI StI=↑35rael SeI={ ↑50gate,↑54asciir } | PDO InForm=↑74pdf RAE={ ↑87pcpdfV,↑88macpdfV } |

| | | |
|---|---|---|
| PDI CRID=87pcpdfV Title=pcPDFviewer | RI StI=↑34raep SeI={ ↑50gate,↑54asciir } | PDO Platform=↑91pc engine=acroread.exe Params=none OutForm=↑75gui |

These last three nodes illustrate some important features of our representation net. ↑32uafp is a Gödel end for UAF nodes, ↑87pcpdfV is a Gödel end for the rendering platform. We show PDF as a platform ( ↑51pdfr ), which is one of a number of choices offered in section 8. As the Gödel end platforms become obsolete this formalism offers the choice of adding PDF rendering capability on new platforms or emulation of the obsolete platform upon which the existing rendering software will run. The parameter, shown in brackets in ↑01trav, is passed through to the software engine of ↑87pcpdfV via the ↑51pdfr. If desired, (tiger.pdf) could be rendered via ↑88macpdfV instead. All new rendering software for pdf is added to the list of RAEs in ↑51pdfr.