Project Number	IST-2006-033789
Project Title	PLANETS
Title of Deliverable	Report on tool and service approach
Deliverable Number	PA/4-D1
Contributing Sub-project and Work-package	Subproject PA WP PA/4 (Tools for objects)
Deliverable Dissemination Level	Internal PA - Restricted to other programme participants (including the Commission Services)
Deliverable Nature	Report
Date	28th February 2007
Author(s)	BL, KB-NL, SB, KB-DK, MRL, ONB, BAR

Abstract

This deliverable describes the analysis and discussions performed by the PA/4 groups (Tools for objects). We describe the proposed preservation architecture of PLANETS as we understand it, and our area of responsibility. Specifically, we concern ourselves with preservation actions and quality assurance (QA).

We discuss several fundamental questions regarding preservation of digital objects from the point of view that the goal of digital preservation is to preserve fidelity (quality of representation of some information in the object) and inclusiveness (preservation of information in the object). We identify several challenges, mainly that automated QA is currently very limited, both in coverage of formats, in thoroughness of the QA, and in that no known QA systems perform comparative QA, i.e. compare the fidelity of a conversion from one format to another. The concept of digital object complexity is identified as a core factor in estimating preservation and migration difficulty, but we are unable to devise a general measure, or a general way to measure it.

Further, we identify the “many interpretations” model of digital objects as desirable, as it reduces risk of loss of fidelity and inclusiveness reduces reliance on few, complex formats and allows for a new model for performing comparative QA, i.e. by comparing only some interpretations of the objects considered, rather than all aspects of it. This conclusion is

validated by the fact that the XCL working group independently reached much the same conclusions.

We then investigate the requirements for reusing current tools by “wrapping” them for a Service Oriented Architecture (SOA). We conclude that it is likely possible, but that it presents several technical challenges, which we identify and briefly address. Our proposal is a very “slim” interface that aims to be very narrowly defined in terms of input and output format. Presumably, for more capable programs, that e.g. accept several input formats, an action can be created for each possible input format.

Finally we report on actual tool usage at the institutions, and topics opened, but not thoroughly investigated during the work period.

Keyword list:

Preservation actions, preservation plans, digital archive architecture, quality assurance, QA, automated QA, SOA, XCL, digital objects.

Contributors

Person	Role	Partner	Contribution
Anders S. Johansen	Editor/contributor	KB-DK	Editor of document. Written sections 1, 2, 3.1 – 3.3, 3.5 – 3.7, 6, 7, 8, 9. Contributed to general analysis.
Caroline van Wijk	Contributor	KB-N	Written sections 3.4, 4.3, 5. Reviewer of document. Contributed to general analysis.
Lars Clausen	Contributor	SB	Co-author of sections 3.8, 4.7 and 8.1. Reviewer of document. Contributed to general analysis.
Toke Eskildsen	Contributor	SB	Co-author of sections 3.8, 4.7 and 5. Reviewer of document. Contributed to general analysis.
Hamid Reza Mehrabi	Contributor	KB-DK	Written section 4.1. Contributed to general analysis.
Peter Bright	Contributor	BL	Written section 4.2. Reviewer of document. Contributed to general analysis.
Eleonora Nicchiarelli-Bettelli	Contributor	ONB	Written section 4.4. Reviewer of document. Contributed to general analysis.
Jean-Marc Comment	Contributor	BAR	Written section 4.5.
Natasa Milic-Feryling	Contributor	MRL	Written section 4.6.
Niels H. Christensen	Contributor	KB-DK	Co-author of section 3.4. Contributed to general analysis

EXECUTIVE SUMMARY

Objective

This document aims to present promising ways to improve the current state of the art in tools for objects practices in relation to the current best practices at the institutions involved in the PA/4 workpackage, the current state of the art of digital preservation approaches concerned with transformations of objects in general and the challenges involved for transformation tools.

Scope

This deliverable contains best practices at the participants of the PA/4 workpackage. Furthermore it states the current state of the art in the area of tools for objects, the challenges that tools for objects offer and novel approaches to improve the current state of the art. The context for the novel approaches is formed by issues such as characteristics of a preservation action, transformations and preservation planning and is briefly outlined.

Best practices at institutions

The archives described vary greatly in architecture and strategy. The number and class of tools used by the participating institutions are somewhat lower than one would assume, given the wide variety of mission and data types stored and processed.

A common problem appears to be a dearth of tools for automated QA. In the cases where automated QA is employed, is must be described as shallow and spotty, and not comparative. In effect, only some types of objects are subject to automated QA, and they are only tested for validity, not for whether they are a satisfactory conversion of the source object. The only known cases of comparative QA have been performed entirely as manual operations, and appear not to have been exhaustive.

State of the art

The concept of file format - a standard for encoding some (related) items of information as a sequence of bytes - is a convenient way to devide digital objects into categories or ‘families’. However, even the most precisely described file format might contain specification variations. It is important to analyse which variations to expect, how to notice them, and automatically validate the compliance with the standard.

The current approach to transformation of digital objects for preservation purposes is to convert an individual file in one file format to another format. One approach for this type of transformation is to convert to a more generic ‘superformat’. The superformat contains as many features as possible from one class of digital objects (e.g. text documents). However, most superformats would fail in a file format risk assessment because of their complexity when digital preservation is the objective.

Quality assurance (QA) is performed manually for most of the transformations. Automated QA is only possible in a few specific cases.

Challenges

Containers are objects that contain several other digital objects, or should be approached as such. A crosscutting concern for all these containers is that the individual container may not preserve information about the objects it contains, be that their type (which it may or may not be possible to infer from the contained object), or the boundaries between the contained objects. If no metadata for this is present in the container, and no outside source is available to provide it, the contents are essentially lost until such time as the situation is remedied, which may be never. Container formats tend to present several challenges other formats do not that they are often more complex to process.

There are two fundamental models for encoding information in digital objects: sampled objects and interpreted objects. Sampled formats present fewer challenges in terms of presentation, as they have fewer external dependencies, and are inherently quantified. This is not so for interpreted objects. They are inherently dependent on an, often complex, interpreter.

We have some reservations with regards to wrapping existing tools for SOA use, mainly based on the fact that this is a new and unproven concept, at least for the participants in PA/4. We do however feel that this is feasible, and advocate a "thin" approach, i.e. strongly typed, and functionally narrow preservation actions exposed from existing "fat" and capable programs. We note that while migration of tools to the data, rather than moving the data to the tools as proposed in PLANETS may be more efficient, it is not a feasible strategy due to the challenges of migrating live programs in a heterogeneous environment.

Specific issues concerning file formats should be taken into consideration when picking a file format to transform to and when considering different transformation tools and when building new tools. A few of these issues are concerned with the lack of specifications when the format has evolved over time, specifications may be proprietary or so large that they are nearly impossible to implement. Also transformation programs may have implemented only a subset of the specifications. It also goes to show that merely validating a file against a formal syntax specification, even when possible, does not necessarily determine whether or not the file can be transformed.

Novel approaches

We identify a multiple stream/multiple interpretations object model as desirable. The multiple stream model relies on the “aspects” abstraction, and views a digital object as the sum of its potential renderings. The multiple interpretations model suggests that a logical digital object may be constructed from several digital objects, that each store one or several of the streams/aspects of the logical object. This facilitates an archive architecture that allows for a much more flexible and risk-reducing preservation and migration strategies that the traditional one object/one interpretation approach. Further, this approach facilitates a layered model that neatly partitions the preservation architecture into smaller, better-defined steps, that are easier to solve and (perhaps more importantly) reuse and combine into more complex operations. This is a classic “divide and conquer” approach. Layered models are a staple of computer science, and have proved to be a superior model in many other areas e.g. networks

An aspect is an abstract view of (a subset of the) information in one or more digital objects. Example aspects could be ICC profiles in JPEG images, metadata in MP3 files, or page breaks in Word documents. The problem that aspects attempts to address is this: Unlike physical objects, digital objects may contain layers of information that are not immediately obvious when the object is rendered in a way that emphasizes other layers of information in the object. So the concept of aspects provides a more precise and flexible view of a digital object than "file format". Hence, the aspect concept maps very well to the multiple streams object model and the multiple interpretation model for object conversion.

We investigate the areas of digital archive preservation that XCL address in the context of our independent analysis of the problems with current approaches and our proposed solutions. We note that we have independently arrived at many of the conclusions that the XCL group have, mainly the multiple stream object model, the multiple interpretation model for object conversion, and the concept of performing "fuzzy" QA by comparing simpler interpretations of the source and destination object of conversion. We conclude that the XCL approach has merit as a general solution to the problem of performing comparative QA.

Next steps

A native tool encapsulation web service prototype has been developed by PA/4 . The aim has been to implement a simple web-service. The prototype –implemented in Java- among others includes input form and output to local file system and no security handling. Dia is used for the prototype as

native tool (multiplatform, capable of migrating between different vector graphics formats), but generic native tool interface

Conclusion

We conclude that transformations should be non-destructive, e.g. only be allowed to add to the current data associated with a logical object, be that metadata or digital objects. Removing old, unused formats rightly belongs under administration or data administration, nor preservation.

The primary current operational challenge is automated QA. Currently, automated QA is spotty (e.g. that it is only possible for a few formats), and often not thorough (e.g. often only cursory validation is performed). Worse, even in the cases where automated QA is in use, it only provides “isolated QA”, i.e. validation of whether the objects produced can be processed. What we really need is automated “comparative QA”, that actually compares the source object with the destination object after conversion.

The secondary operational challenge is, as always, to find or develop tools for providing the required transformations from existing formats to new ones with acceptable fidelity and inclusiveness, but reports of current practice from the institutions indicate that while this is a common concern, all institutions are currently able to fulfil their mission at least partly.However, improvements of these current practices is very much needed..

Object complexity is mainly a practical problem of interest to cost/benefit projections for preservation, and not our area of responsibility when considered as a static problem (“the current state of an archive”), but we must point out that relying on a superformat migration strategy will likely increase the average complexity of objects, and hence the price of maintaining logical access (“the implications for future archive complexity of a superformat strategy”).

As for SOA wrapping of existing tools, we conclude that it is possible, but requires that several fundamental decisions with regards to implementation and policy are made and agreed on. Examples are error handling and file access. We point out that while migration of live preservation actions will often be desirable in a distributed SOA, as the data to process will usually be significantly larger than the live code needed to process them, it presents almost insurmountable problems when the preservation actions consist of a wrapped tool. The problem is, simply stated, that these tools in the general case rely on a runtime environment that consists of the entire computer they are installed on, which makes migration of live programs extremely hard.

We propose that XCL is a promising technology to enable comparative QA in a general way. Unfortunately there are limits to the level of support that can realistically be expected for complex formats, and therefore XCL can not be expected to solve this problem entirely.

TABLE OF CONTENTS

1. Introduction

1.1. Purpose

The PA/4-D1 deliverable aims to identify:

- Current best practice in the use of tools for objects at the institutions that are participant of the PA/4 workpackage
- Common problems with the current state of the art of tools for objects
- Promising ways to improve the state of the art of tools for objects, be that in concepts, nomenclature, tools or infrastructure, and which define a tool approach for tools for objects in the context of digital preservation

1.2. Scope

This deliverable consists of the description of the tool and service approach for tools for objects within PLANETS, the current state of the art and challenges, which both are starting points for the tool approach. This document presents overviews of the current best practices in tools for objects, of the common challenges that institutions face when using tools for objects and of new approaches that can improve the current state of the art on tools for objects.

Other areas of interest for the PA/4 workpackage that are not dealt with in detail in this deliverable, but are part of the scope of the PA/4 workpackage and form the context for the tool and services approach, will be mentioned in the Context paragraph.

1.3. Context

1.3.1. PLANETS objectives

The Planets proposal presents some scenarios that our work should hopefully eventually facilitate. Beyond the obvious (e.g. improving characterization, better file format registries...) the goals are very far-reaching and visionary, that is, they require that the state of the art is ultimately improved on during the PLANETS project.

According to the PA/4 group's analysis, the main concrete, technical goals for the project as a whole are:

1. Preservation action decision support ("Here is my archive? What happens if I convert the Word documents to format <X>? Should I?")
2. Preservation action formulation and development ("How do I convert my archive of images to format <X>")
3. Preservation action dissemination ("I need to convert my images to format <X>. Can I download something that will do that for me using best practices?")

The goals 1 and 2 (decision support) can be broadly viewed (informally) as "How do we help computers to help us better in this area?".

The goals 2 and 3 (development and distribution) can be broadly viewed (again informally) as "How do we develop a platform that enables us to develop and disseminate 'preservation actions', preferably with decision support from computers?"

Note that goal 2 is both concerned with decision support, development and distribution, as it is assumed that one likely way to improve the state of the art in this area is computer-based decision support on several levels.

1.3.2. Characteristics of a preservation action

The current PLANETS definition of a preservation action is as follows:

"The execution of an action to ensure the continued accessibility of an authentic digital object across time and changing technical environments that transforms the digital object itself, the technical environment required to support access to the object, or a combination thereof"

Some of these preservation actions will result in code being executed in the archive, and some will not, as the definition is not restricted in this regard.

To set the preservation action context for this document, we will first put focus on the concrete realities of a single "migration action", the act of migrating one digital object. We want to answer the following questions:

- What data is required in order to predict the quality of the result of the action?
- What is the relevant measure of quality for such a prediction?

The answers to these questions will help PA/4 to understand which “characteristics” of (existing) tools are important for evaluation and/or development of new tools.

Furthermore, the input from PA/4 on this topic is of importance to work package PA/2 as this work package is to deliver a generic model for tools and services and of importance to PA/3, the work package that will develop a tools & services registry. Synergy between the PA/2, PA/3 and PA/4 work packages consists of the development of user scenarios for migration tools and the use of a tools and services registry.

It's all about the bits, really

At the end of the day, if you're not in the business of manually rewriting digital objects, some pre-existing piece of code will have processed one or more pre-existing streams of bits. As a result, one or more new streams of bits will have been delivered to some device—probably a disk or some other persistent storage device. (Strictly speaking, migration need not generate actual bit streams, but that behaviour is so common that we shall ignore other possibilities here).

As the people who are going to perform that action, we want a registry to tell us whether it is at all applicable, and what we can expect of the output.

- **Example 1:** We are considering converting the (pre-existing) bit stream "tucan.gif" using the (pre-existing piece of code) "tifiptc2dcxml". Before diving into that, we would like to have an idea about whether this is applicable (does "tifiptc2dcxml" accept a digital object like "tucan.gif" as input?) and what sort of output can we expect to get out of it?
- **Example 2:** We are considering converting the (pre-existing) bit stream "paper.ps" using the service "ps2pdf.com", which actually just wraps the (pre-existing piece of code) "ps2pdf". Before performing that conversion, we would like to have an idea about whether this is applicable (does "ps2pdf" accept a digital object like "report.ps" as input?) and what sort of output can we expect to get out of it?
- **Example 3:** We have a PDF document, version 1.4 with all Document Restriction items on "allowed" except for Printing. We would like to migrate our 1.4 version to a newer version: 1.6. We expect the registry to give us a list of tools that are able to do PDF version migration to 1.6. We also expect to see the evaluation of the tools: what have been the experiences of others with the tools for example. We do not have any requirements about the viewer that we will use for the newer version. The only thing

we want is that the Document Restriction item Printing will still be on "not allowed" in the new version.

- **Example 4:** We have a WordPerfect document, containing tables and simple paragraphs that we would like to convert to PDF/A. We look for a conversion tool or service that creates PDF/A and also takes in bit streams that are formatted as a WordPerfect document. The only requirement that we have for the migration is that the tables and paragraphs will be intact after the conversion.

The examples listed above are all based on having a concrete file. A different example scenario could be a lot less concrete, e.g. somebody is planning a repository and wants to know what potential issues arise if standardizing on accepting "PDF" files (no specific version provided). Presumably a system with extensive knowledge of file formats, sub formats, migration/conversion actions and their consequences could provide decision support for such a query. This goal—high-level decision support for archive planning—is outside the scope of PA/4, and will not be discussed further here.

And good intentions, too

The quality predication of our migration action should be fitted to our expectation of the action's final result—the registry should actually predict our satisfaction with the output of the migration action! The nature of our expectations will be whether the output will render satisfactorily with our intended viewer/view path. (In some cases, we may even want to know if the output is applicable for another conversion program).

- **Example 1 continued:** We intend to view the output of "tifiptc2dcxml" applied to "tucan.gif" with the program "dublincoreviewer". On doing this, we expect (at the least) that the "Publisher" field to contain something meaningful.
- **Example 2 continued:** We intend to view "ps2pdf"'s output from "report.ps" using the program "acroread". On doing this, we expect to the text to be readable and nicely laid out, but not necessarily searchable or containing a navigable table of contents.
- **Example 3 continued:** To evaluate our migration, we would like to use a metadata extraction tool (such as JHOVE) that extracts the Document Restriction items of the original object and the converted one. Then we would compare the two extractions (perhaps using a comparison tool). We could view the original and converted object both in Acrobat Reader 7.0 to see if there are different behaviours/appearances etc.
- **Example 4 continued:** In an ideal world we could just have an extraction tool extract the tables and paragraphs from the original WordPerfect document and from the newly created PDF/A document. Then we would compare the two extractions to see whether the tables and paragraphs in the migrated version were still intact. Another way to evaluate the migration is to open the WordPerfect document in a WordPerfect viewer and do the same for the PDF/A document in a PDF viewer. We would (manually) compare the tables and paragraphs.

Summing up the examples

For example 1 and 2 the summary of the scenarios can be as described below.

Our query to a registry is likely to be composed of at least the following components:

- An actual bit stream such as "tucan.gif" or "report.ps".
- An actual program (in fact also a stream of bits) for conversion like "tifiptc2xml" or "ps2pdf".
- Another actual program or entire view path for rendering the output.
- A stated intention, our expectation to What We See when the output is rendered.

This list is not complete—for instance, concrete parameters for the programs (like "--strict" on a command line) may very well be needed. Note also that in querying the registry we don't send all of these bit streams; most likely essential properties will be derived locally and only these will be sent on the network somewhere.

For example 3 and 4 the summary is a little bit different.

This is where we need to make use of a tools registry:

- An original digital object (bit stream)—some technical/functional knowledge (characteristics or otherwise named and/or file format)
- An idea of what we want to do with the original digital object—conversion to x (is of course variable)
- A list of requirements for which characteristics of the converted digital object will be as they are in the original object (example 1: all Document Restrictions be kept and example 2: tables and paragraphs be intact)

The difference between the first two scenarios and the last one is the role of a viewer of the digital object in the migration process in the first two examples and the mentioning of specific file formats and versions in the last examples.

As a result of these queries, we would expect some sort of quality prediction along with any tools suggested. This prediction could be as simple as “No” (it will not work), or it could be more complex, like a human-readable listing of how well various parts of the files can be expected to be preserved, what issues we can expect to encounter, and what the system requirements are.

Consequences

- *Formats and aspects can never be a goal, just a means***.** There is **no** mention of format in the above discussion in the first two examples. If formats help the registry to answer our query, such as in the last two examples, that's all well and good, but the end goal is to be able to say something meaningful about a specific combination of bit streams, converters and view paths.
- *Non-standard features of viewers (and converters)* ***must*** *be part of a digital object's characterization.* Example: We are converting an HTML page that was intended to be rendered with Internet Explorer. If the page actually uses (in a significant way) features of IE that are not part of the HTML specification, this information **must** be available to the system that evaluates the migration action—otherwise it might recommend a conversion tool that does not render the page like IE would.
- *We need to understand the dimensions of intentions.* Our expectations to the migrated object are important. We cannot get away with just wanting the "best possible". In Example 1, the expectations can be clearly defined by the Dublin Core standard and thus easily verified, whereas in Example 2, we might be in a situation where no known conversion preserved both layout and search ability of the original document. On helping us, a system must provide us with a **reasonably short** list of **understandable** choices.

1.3.3. Archival Information Packages AIPs and abstract digital objects

The “atoms” of digital archives are Archival Information Packages (AIPs). The Digital Preservation Coalition describes AIPs such in their introductory guide to the OAIS model [1].

“The Archival Information Package, or AIP, is the version of the information package that is stored and preserved by the OAIS. The AIP consists of the information that is the focus of preservation, accompanied by a complete set of metadata sufficient to support the OAIS’s

preservation and access services. The archived information and its associated metadata represent a single logical package within the archival system: there is, however, no requirement that any form of physical association be maintained, such as embedding the metadata in the information object itself and storing the combined object as a single bit stream. Arrangements for storing archived information and its metadata are left to the OAIS's implementers; possible solutions might range from complete physical integration, to storage in separate yet logically related databases."

In a digital archive, the individual AIP supposedly references a "digital object", which is the focus of the preservation effort.

It is worth noting that this may be an "abstract digital object", which in actual fact can be stored in several files, which may or may not be of the same format. These files might be termed "concrete digital objects", but are usually referred to as just "digital objects".

1.3.4. Transformations

We refer to the actual pieces of code that manipulate AIPs as "transformations". To avoid confusion, we note that when transforming data in the digital realm, the source object is usually not replaced with the destination object, unless it is a specific requirement of the environment in which the transformation is performed. We feel that the act of removing the source object from the archival storage is not a preservation action. It does nothing to enhance or preserve the access to the original digital object (abstract or concrete), and will in fact in most cases decrease the authenticity of the accessible digital objects, as some authenticity will almost always be lost in the transformation.

Hence, we advocate this definition of a transformation in the context of a preservation action:

"An action that creates new data from existing data in the archive, with the intent of preserving or increasing access to information stored in the archive, without modifying the existing data".

The new data produced is usually categorized as one of the following:

1. **(Preservation) metadata**. This is data that aids in characterization, describes requirements for rendering or executing the object, or records actions taken to otherwise process, archive or itemize the object. Most of the metadata that might be changed by a transformation will be preservation metadata, i.e. metadata that is specifically maintained to provide a record of requirements for preservation and actions taken for this purpose. For a detailed description of a real-world recommendation for a state-of-the-art preservation metadata scheme see the PREMIS preservation metadata model.[2]
2. **Digital objects.** These are new versions of the original object(s) generated for the purpose of preservation. Usually the new format(s) are generated to preserve or enhance access to the content of the object. Generating new interpretations of digital objects will usually also result in changes to preservation metadata.

This generation of new data is presumably for the nebulous goal of "preservation", but it is not the purview of the individual transformation to determine if it is acting to "preserve" anything—the concrete translation of a perhaps nebulous preservation goal info concrete transformations is performed at a higher level, and is expressed in the choice of which transformation(s) to apply to which objects, and in which order.

There are three cases where data may need to be deleted:

1. Intermediate data generated during a transformation may be deleted once the transformation is successfully completed, or it is aborted and examination of possible error causes has been done. Some intermediate data or extracts thereof may be useful as preservation metadata, though.

2. Referential metadata that needs to point at the transformed data rather than the original may be overwritten. It is preferable, though, that the metadata is kept as-is and a layer is inserted that resolves the mapping from old to new objects. Such a layer also makes it easier to assess transformation quality later, or use the originals in ways that the transformation process did not allow for.
3. For economic reasons, some institutions might not be able to afford storage for both the originals and the transformed versions. Throwing away old versions is not a decision to be taken lightly, though, and alternatives should be considered first. Alternative ways to keep originals around include storing them off-line on archival-quality media, storing them with commercial storage providers, or cooperating with larger public institutions that can spare some storage space with simple access requirements. Throwing the originals away should always be the very last option.

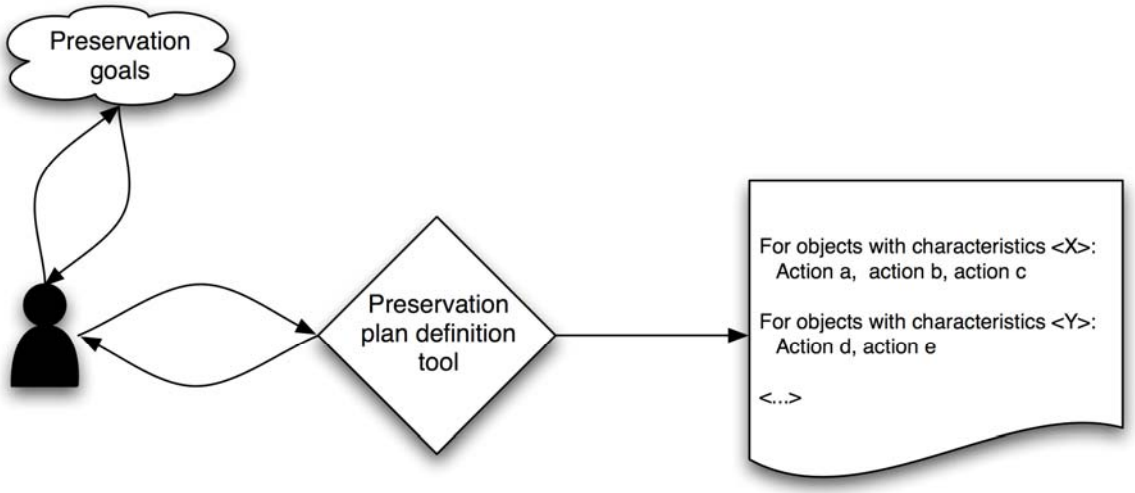
In general, we do not see it as part of the PA/4 task to decide when to delete materials, rather it is a preservation planning task. Thus we assume that transformations never delete data from the archive.

The PA/4 group interprets our brief (“tools for objects”) to mean that we are only concerned with transformations that input and output digital objects (“files”), not transformation of metadata, inasfar as the two can be distinguished.

1.3.5.Preservation planning

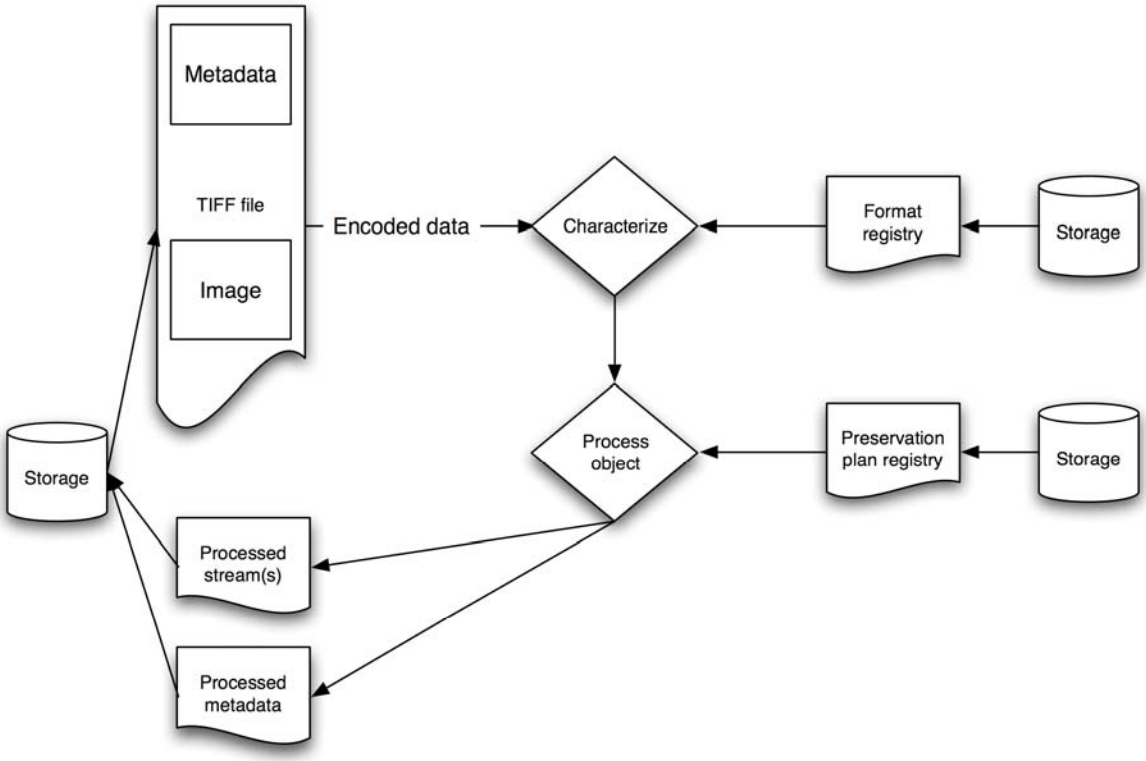
A plan is defined by a human from abstract goals, which are usually not initially defined in terms of concrete preservation actions. An example of such an abstract goal could be "keep scientific articles in the archive accessible without unacceptable loss of printing quality".

Using a plan development tool this plan is then converted from abstract goals to concrete preservation actions that should be performed on objects that have some specific characteristics, often in an iterative process. Goals may change based on the knowledge gained during the process. Some of these concrete preservation actions will resolve to transformations.

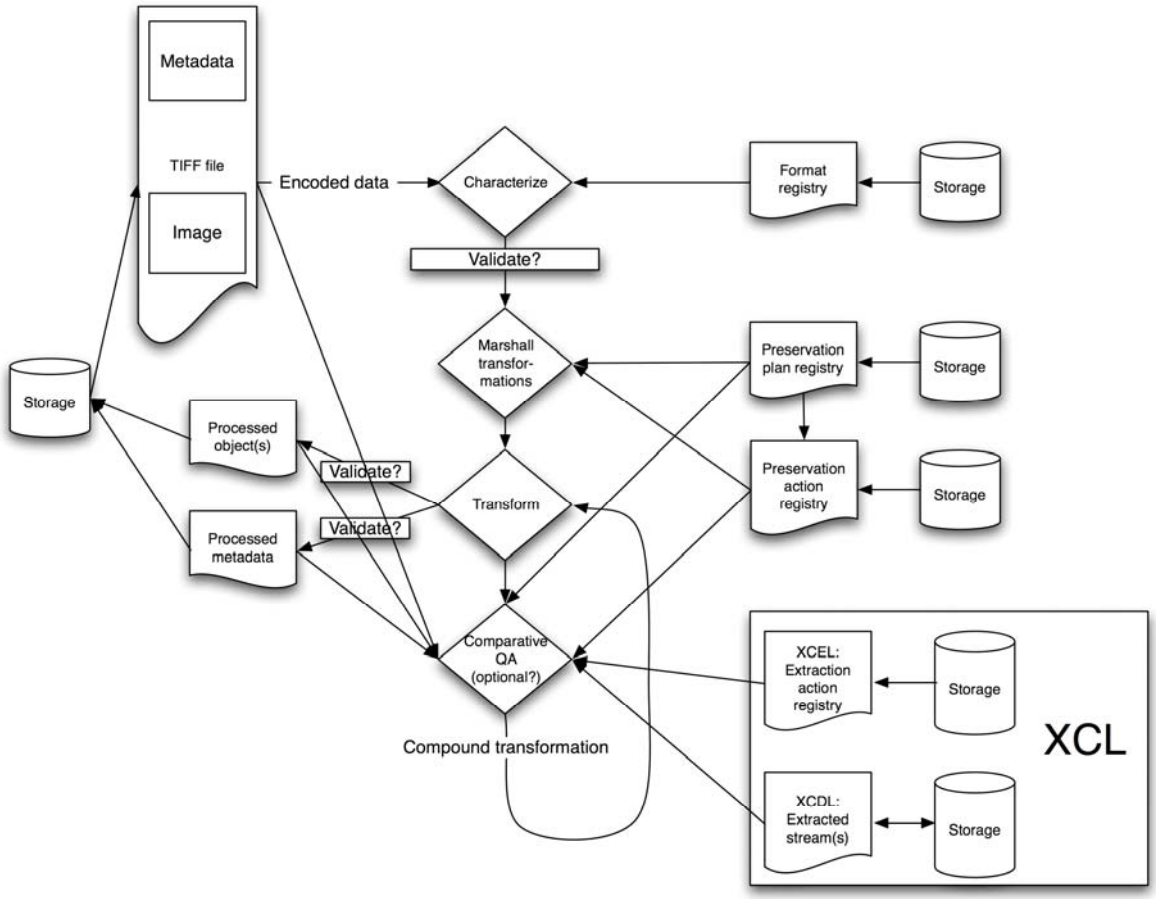


The executable end result is a "master plan” that consists of one or more sub-plans, defined as transformation that should be applied to objects that have a given set of characteristics (“file format, in traditional terms). These transformations may again consist of a chain of transformations, depending on whether the architecture is based on "fat", complex transformations or "slim" simple transformations that can be combined into more capable ones.

The traditional view of preservation planning is a "fat transformation" view: Essentially it devolves to "run tool X for all files in format Y with parameters Z".

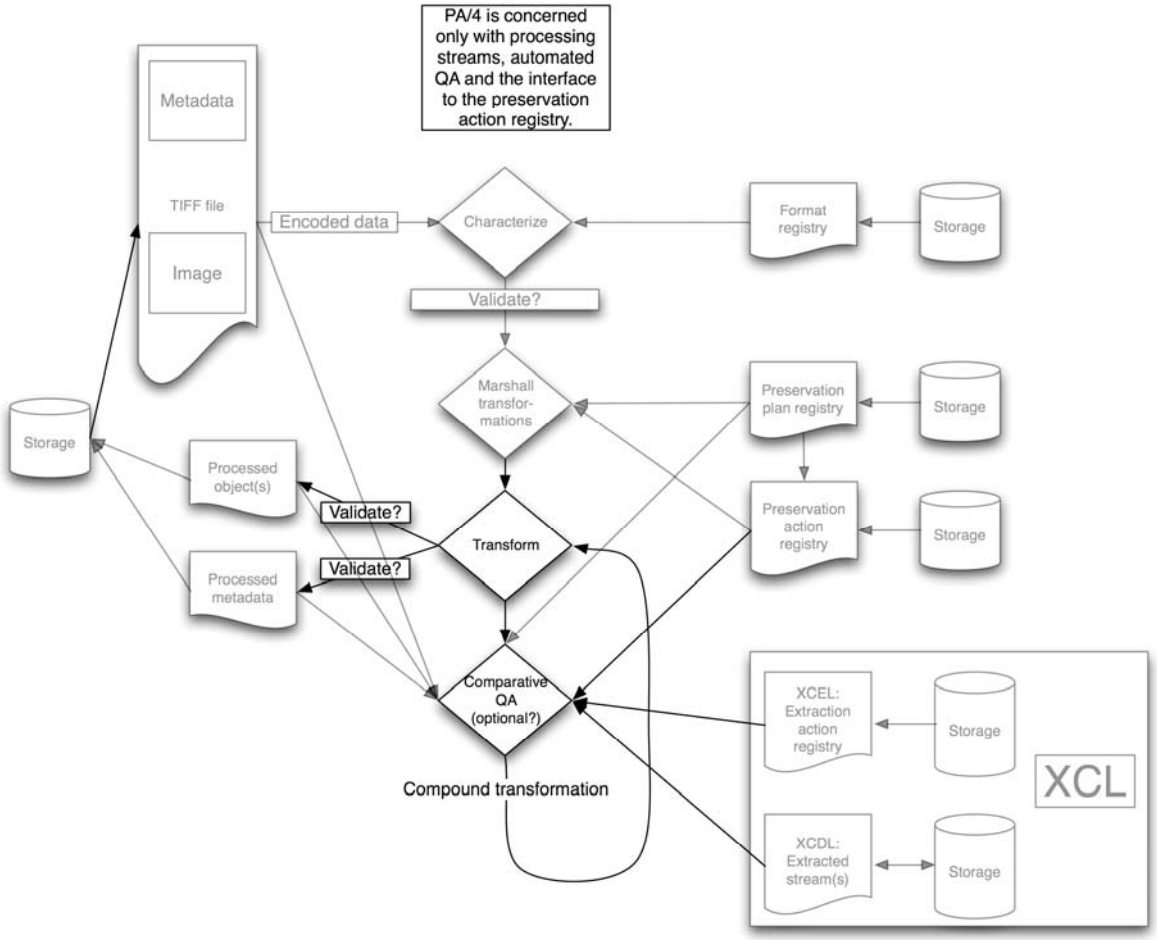


The architecture currently discussed in PLANETS is somewhat more complex, but the complexity allows for compound preservation operations, and a clear separation of preservation plan enactment, transformations, QA (Quality Assurance, or validation) and storage. It's worth noting that it will usually store multiple interpretations of the digital object due to the definition of a transformation in the previous section. Also note that the use of compound transformations amounts to using a combination of the design patterns "command" and "composite" [3].


Metadata
TIFF file
Image
Encoded data
Characterize
Format registry
Storage
Validate?
Marshall transfor-mations
Preservation plan registry
Storage
Storage
Processed object(s)
Validate?
Transform
Preservation action registry
Storage
Processed metadata
Validate?
Comparative QA (optional?)
XCEL: Extraction action registry
Storage
XCL
Compound transformation
XCDL: Extracted stream(s)
Storage

1.3.6. Analysis, conclusion and scope of PA/4

PA/4 is concerned with data extraction and conversion, and the QA process in as much at it can be automated. In the described architecture, this is very narrowly defined: This corresponds to the individual transformations and the associated QA.



QA should in principle be enforced, but due to inherent limitations on the scope of automated QA, it is unrealistic to require that the results of all operations must be subjected to automated QA. It is also unrealistic to assume that "perfect QA" is possible, especially in the face of nebulous goals for the preservation plan. Often the QA devolves to "I know what's acceptable when I see it", i.e. a value judgment performed by a human expert, which is currently hard or impossible to automate or quantify.

This is not to say that (automated) QA is pointless. Even limited QA is valuable, but one must be constantly and acutely aware that even if the results of enacting a preservation plan passes QA this is no guarantee that the intended result was achieved.

1.4. Overview

The remaining parts of the deliverable are as follows: in chapter 2, we present the experiences in doing transformations at the institutions. In the third chapter, an overview of the current state of the art and challenges involved in doing transformations is presented. This is followed by a chapter on novel approaches that will improve the use of tools for objects for preservation action purposes. Conclusions on best practices and novel approaches are presented in the Conclusion chapter. The last chapter of this document contains the references used in this deliverable.

2. Current best practices

2.1. Report on tool usage in PA/4

In this section we report on the tools used in the institutions participating in PA/4. This is intended as inspiration for practical, operational use of current tools, and as a way to gain insight and perspective regarding our deliberations and suggestions.

2.1.1. KB-DK

Introduction

At KB-DK we are currently at the verge of declaring our Digital Object Management System (DOMS) operational. Our DOMS is based on a strategy of maintaining high-quality digital masters, and deriving the formats necessary for various uses on the fly from these masters.

We expect to support several types of objects in the final version, but initially we focus on bitmapped images. We have chosen the TIFF format for our masters, as this format is stable, well-known and is already in use in several of the departments that produce or process images (mainly digital photography, graphical design and scanning of books and other materials).

The disadvantages of the TIFF format are that it is very complex, and does not support lossy compression. As we currently do not see a need to employ lossy compression for our image masters, that particular limitation is not a problem in our case. The complexity does have some undesirable secondary effects, mainly that it appears to be the reason some tools do not function correctly in our workflow, i.e. that the TIFF format allows for some features that some tools do not support.

Another common problem when using TIFF files is that the format allows for metadata to be stored along with the image(s) in the files. Unfortunately not all TIFF conversion and manipulation programs process these metadata correctly. Hence, it is not sufficient to perform QA of a conversion merely by inspecting the images produced – the metadata must also be investigated for correctness.

Common problems encountered with regards to metadata damage during conversion are:

- Encoding of Danish characters is inconsistent in a multi-platform environment. We use a wide variety of systems, including Mac, Windows and Linux. As there apparently is no standard for text encoding in TIFF metadata, we often find that manual editing is necessary to restore the Danish special characters æ, ø, å, Æ, Ø and Å.
- Various colour profiles and other technical metadata can be stripped during conversion. This is a major challenge, as these data are often vital for preservation of images, particularly in the case of digital photography. This has been a major factor in selecting the tools currently employed for TIFF processing at KB-DK.

TIFF format variations supported

The DOMS workflow currently requires that the following areas of the TIFF standard can be processed by the tools we employ.

Full name	TIFF, Group 4 Compressed Bitmap

Description	A tag-based file format for storing and interchanging raster images; in this subtype, TIFF wraps a bitmap compressed using ITU_G4 (ITU-T T.6. Facsimile Coding Schemes and Coding Control Functions for Group 4 Facsimile Apparatus), and is thus limited to bitonal imaging, suitable for the reproduction of typographic or line-art originals.
Production phase	Most often an initial-state or middle-state format; may serve as final-state format.
Relationship to other formats	
Is subtype of	TIFF_6
Contains	ITU_G4, ITU-T Group 4 FAX Compression

TIFF, Group 4 Compressed Bitmap support B/W images.

Full name	LZW (Lempel-Ziv-Welch) Image Compression Encoding
Description	A lossless compression algorithm for digital data of many kinds, named for the creators Abraham Lempel and Jacob Ziv, and a later contributor, Terry Welch. LZW is based on a translation table that maps strings of input characters into codes. Through its incorporation in the graphics file formats GIF_89a and TIFF_LZW, LZW has come to be strongly associated with image compression.
Production phase	Used for initial-, middle- and final-state (end-user delivery) purposes.
Relationship to other formats	
Used by	TIFF_LZW, TIFF with LZW compression
Used by	GIF_89a, Graphics Interchange Format, Version 89a
Used by	Other file or wrapper formats, not documented at this time

LZW (Lempel-Ziv-Welch) support B/W, grayscale and colour images.

Full name	TIFF (Tagged Image File Format), Revision 6.0
Description	A tag-based file format for storing and interchanging raster images. TIFF serves as a wrapper for different bitstream encodings for bit-mapped (raster) images. The different encodings may represent different compression schemes and different schemes for colour representation (photometric interpretation). The most recent version of TIFF is 6.0, published in 1992. Since TIFF images conforming to earlier versions are valid TIFF 6.0 files, the information in this description is also pertinent to earlier versions of the TIFF standard. Many TIFF

	files with uncompressed image data are still being created as TIFF 5.0 files.
Production phase	Most often an initial-state or middle-state format; may serve as final-state format.
Relationship to other formats	
Has earlier version	TIFF, Revision 5.0, not separately described
Has subtype	TIFF_UNC, TIFF, Uncompressed Bitmap
Has subtype	TIFF_G4, TIFF Bitmap with Group 4 Compression
Has subtype	TIFF_LZW, TIFF Bitmap with LZW Compression
Has subtype	TIFF/IT, TIFF/IT, for Image Technology
Has subtype	TIFF/EP, TIFF/EP, for Digital Photography
Has subtype	DNG_1_1, Adobe Digital Negative (DNG), Version 1.1
May contain	Bitstream encodings for other compression schemes not documented at this time.

TIFF, uncompressed bitmap, greyscale and colour images.

Additionally, we require support for **Pyramidal TIFF**.

Tiled Pyramidal TIFF is simply a tiled multi-page TIFF image, with each resolution stored as a separate layer within the TIFF. This is a standard TIFF extension and is supported by most image processing applications.

Tools employed

As we have worked intensively with TIFF files for seven years, we have extensive experience with tools suitable for processing such images. Our primary tools are currently:

- PhotoShop
- AcdSee
- Leadtools with imaging SDK
- Total image converter (only occasional use)

These programs are also used in-house for other purposes.

PhotoShop

PhotoShop is very simple to use. It allows for user-created macros (“action” in Photoshop nomenclature), which can be used for conversion. Photoshop supports all known variants of the TIFF format except multiple TIFF. Hence, it is not possible to convert from single files (in TIFF) to multiple TIFF and vice versa using Photoshop.

Processing time for conversion to pyramid TIFF is very high. It takes several minutes to convert high-resolution images (600 DPI in TIFF RGB colour space) to pyramid TIFF.

It is not possible to run the software from command line.

Some of the fields of the header file will be changed during conversion. The changes do not appear to be important for viewing of the images.

The software is very stable, which means that all the TIFF images, which are generated by Photoshop, can be viewed in many other TIFF viewer applications.

AcdSee

AcdSee is primarily a viewer, but it can also be used for file conversion. In terms of capabilities and quality of the files produced, AcdSee is similar to Photoshop.

There is a major benefit to AcdSee when compared to Photoshop, namely the price. The price is around $50, which is significantly cheaper than Photoshop

LeadTools

This Software can be provided with a Software Development Kit (SDK), which gives full access to integration of all the capabilities of the program in any in-house application. LeadTools supports all variants of the TIFF format in use at KB-DK, including multiple and pyramid TIFF.

Running time for conversion to pyramid TIFF is similar to Photoshop, albeit slightly faster.

It is possible to run the software from command line.

None of the fields of the header file are modified by conversion.

The software is very stable, in the sense that almost all the TIFF images generated by LeadTools, can be viewed in other TIFF viewer applications.

Unfortunately the software is very expensive. The price is approximately $2000 with a SDK license.

2.1.2. BL

The BL thus far has little experience with any migration tools. This position may change as more is learned about the operations of different collection areas within the library; significant migration tools may be found to be in use. Any findings of this sort will be recorded as they are discovered.

2.1.3. KB-NL

Introduction

The digital archive at the National Library of the Netherlands (KB-NL), called the e-Depot, has been operational since 2003. At the moment, its main content consists of 8 million digital publications in the Portable Document Format. Publications in other formats, such as TIFF, JPEG and CD-ROMs containing executables form a part of the e-Depot collection as well. However, this is a very small amount compared to the digital publications in PDF.

Since the end of 2006, publications in more heterogeneous file formats have been deposited at the KB as part of a national project. As a result of this project all Dutch universities will deposit their publications (e.g. articles and dissertations) in file formats ranging from PDF to MS Office Word or PowerPoint and even WordPerfect documents.

The deposit of publications in the MS Office formats and other closed formats (no specifications available of the format structure) is the cause of research into normalization at the KB. The main interest is in conversion tools that can create a more understood (specifications of the file format available) version of the MS Office and WordPerfect documents.

Objective of the tests

All tested tools have in common that their purpose is to convert MS Office documents and sometimes also WordPerfect documents into a version of PDF. The tests were run to determine how well the tools would perform when specific guidelines were applied.

- Guidelines for conversion of the original publication: the converted version should keep most of the characteristics or properties of the original publications. Content and structure (page

breaks, chapters, headings etc.) of the original publication should be preserved in the converted version.

- Guidelines for the target file format: the created PDF version should be a well-formed and valid PDF, fonts embedded, no security or permission settings. Preferably the PDF version that will be created is PDF/A -1b, or after that PDF version 1.6 or 1.4.

Tested tools

Conversion of MS Office and WordPerfect documents to PDF

Batch conversion tools:

Name Tool/Service	**Creator**	**Purpose**
MyMorph	Lister Hill National Center for Biomedical Communications - National Library of Medicine http://docmorph.nlm.nih.gov/docmorph/default.htm	MyMorph converts text-based documents and images to PDF. The KB is testing MyMorph for normalization purposes. PowerPoint, Word and WordPerfect documents are converted to PDF. Windows platform – freeware
Document2PDF Pilot 1.10	Two Pilots http://www.colorpilot.com/	Tested for normalization of Word, Excel, HTML, RTF, text, WordPerfect to PDF, batch conversion. Windows platform
Express Server	Adlib software http://www.adlibsoftware.com/uploadedimages/logo_express.jpg	Tested for normalization purposes. Converts multiple file formats to PDF.Batch conversion possible, command line tool or with GUI.Windows platform
Print2PDF SE 6	Software602, Inc http://www.software602.com/products/print2pdfserver/	Tested for normalization purposes. Converts multiple office documents to PDF Batch conversion possible, command line tool or with GUI. Windows platform

Single file conversion tools:

Name Tool/Service	**Creator**	**Purpose**
BullZip Printer	BullZip http://www.bullzip.com	BullZip Printer converts Word and PowerPoint documents to PDF. Windows platform - freeware The KB is testing BullZip Printer for normalization purposes.
Adobe Acrobat	Adobe Acrobat http://www.adobe.com/products/a	

Standard 7.0?	crobat/	

Challenges

- What are the significant properties? Can you define guidelines for a complete collection without characterizing all objects in that collection?

Perhaps PLANETS can assemble a list of significant properties (significant in a technical sense) for a few much used file formats. Creators of publications or other digital cultural heritage are in the best position to decide what features of the digital object are significant and should be preserved for the long term. If such information is unavailable, the institutions, which provide safe long term preservation and accessibility, should try to formulate a policy about what characteristics of the deposited digital objects are significant and should be preserved as good as possible.

- How to do QA of the look and feel? – JHOVE can be used for a QA, but it has flaws so sometimes you are testing the tool that supports you while testing!

The PLANETS PC sub project will develop some comparison tools that should allow automated QA of migration processes. However, these tools will work for a few file formats (as is the case with JHOVE). They are not a solution for testing whether the look and feel (the appearance) of digital objects is still intact, or at least acceptable, after migration of a digital collection.

- The PLANETS network of tools and services is based on a Service Oriented Architecture. Existing tools will be wrapped to fit into the network. However, preservation action tools that can only be used through a GUI could provide a challenge for wrapping. If an institution needs to use a certain service (or tool) for a large volume of digital objects, it will be more practical to use a local version of the service or tool. What ideas does the IF subproject have on local implementation and the use of web services for large volume processing?
- No tools discovered yet that can create a PDF/A document according to the PDF/A-1b specifications.

The KB-NL has contacted the National Library of Medicine about a possible joint development of a conversion tool that converts several file formats to a PDF/A format.

2.1.4. ONB

ONB does not face the same issues as other partners, since the content providers are asked to convert the material themselves before sending it to the library. As a result, only 10% of the documents present in the ONB digital archive are in a MS Office format.

This is also the reason why no guidelines for migration have been developed at ONB, as well as no consideration on possible automation of QA for migration.

The only migration program currently used at ONB is Acrobat Distiller, for migration of MS Office Word documents to PDF/A. It works well, apart from some problems with the embedding of the fonts. Regarding this issue, PitStop Server has also been tested for conversion from other PDF to PDF/A; not all files were correctly converted, but as it worked perfectly well with embedding of fonts, it was considered better that Acrobat Distiller. It was not adopted as a solution because it is too expensive.

2.1.5. BAR

The contribution from BAR was unfortunately not formatted in a way that allowed for direct inclusion. It has therefore been included as an appendix in the submitted format (Excel spreadsheet).

2.1.6. MRL

Microsoft is a software company with a broad offering in content creation and management software. Its flagship product, MS Office suite of production tools is widely used. Therefore strategies and tools that enable backward compatibility and preservation of created content are of paramount importance.

With the latest release the Office 12, the Office product suite comes equipped with converters for storing MS Word, MS Power Point and MS Excel documents into corresponding Office Open XML formats.

Office Open XML Formats

Office Open XML Formats are open, standardized file formats designed to provide interoperability, transparency and compatibility for Microsoft Office documents that already exist, and those that will be created in the future.

Open XML formats are offered under an open, royalty-free license from Microsoft, and as an ECMA standard. Open file format licenses ensure that any technology provider can incorporate the Open XML formats into their technologies without financial or other consideration to Microsoft.

Open XML formats are compatible with the most widely used business productivity software today, including Microsoft Office, Corel Word Perfect, Open Office, and many other desktop applications and content management systems. Customers can migrate to the Open XML Formats with the confidence that the data within their documents won't be destroyed, and the layout, calculations or other important aspects won't change.

Office File Converter Pack

Office File Converter Pack provides file converters and image filters for Microsoft Office programs, from Microsoft Office 97 to Microsoft Office 2003. These additional converters and filters are for older or seldom used documents or image formats.

The Office File Converter Pack supersedes all previously released versions of the Office Converter Pack.

Note: Additional converters for Corel WordPerfect are available optionally in Microsoft Office Word 2003. To access these converters one must customize installation of Office 2003 to include these optional file converters available under the Add or Remove Programs option in Control Panel.

System Requirements

Supported Operating Systems:

Windows 2000; Windows 2000 Professional Edition ; Windows 2000 Server; Windows 98; Windows 98 Second Edition; Windows Server 2003; Windows XP; Windows XP Home Edition ; Windows XP Professional Edition ; Windows XP Professional Edition Service Pack 2 ; Windows XP Service Pack 1

The Office File Converter pack works with the following Office applications:

- Microsoft Office 2003
- Microsoft Office XP
- Microsoft Office 2000
- Microsoft Office 97.

2007 Microsoft Office Add-in: Microsoft Save as PDF or XPS

This Microsoft software add-in allows the user to export and save to the PDF and XPS formats in eight 2007 Microsoft Office programs. It also allows sending as e-mail attachment in the PDF and XPS formats in a subset of these programs. Specific features vary by program.

The MS Save as PDF or XPS Add-in for 2007 Microsoft Office programs supplements and is subject to the license terms for the 2007 Microsoft Office system software. One may not use it without the license for the software.

System requirements
Supported Operating Systems:

Windows Server 2003; Windows XP Service Pack 2.

This download works with the following Office programs:

- Microsoft Office Access 2007
- Microsoft Office Excel 2007
- Microsoft Office InfoPath 2007
- Microsoft Office OneNote 2007
- Microsoft Office PowerPoint 2007
- Microsoft Office Publisher 2007
- Microsoft Office Visio 2007
- Microsoft Office Word 2007

Further requirements

In order to facilitate effective conversion of a large number of documents, it is important to create programs for collection processing. In either case it is critical to provide diagnostic routines. We are in the process of identifying software tools and practices that were adopted by the Microsoft product teams while preparing the specifications of the Office Open XML Formats for the Office products.

2.1.7. SB-DK

The State and University Library of Denmark (SB) currently have 10-20 ongoing digitalization projects. While these projects are somewhat similar in terms of the formats of the produced data, very few projects share the same technical implementations. A more general solution is underway, with the implementation of a Digital Object Management System (DOMS). This harmonization effort involves analysis and conversion of existing digital objects.

The primary data categories, both in terms of projects and in terms of data volume, are sound, images and moving images. The de facto standards are TIFF, WAV and MPEG 1 & 2. For the DOMS, the same standards have been selected, although specified to Baseline TIFF, Broadcast Wave Format and SVCD & DVD compliant MPEG 1 & 2. Migration of data has been performed for some projects, but the big analysis and migration effort has yet to come.

Most of the data at SB has been digitized for or by SB itself, providing very consistent data objects. Unfortunately this consistency only applies inside each project, due to the different workflows between projects.

Migration challenges or lack of same

The type of scanned images at SB does not require high quality colour reproduction, as the vast majority is either greyscale or scanned pages with text. The practical implication for this is that

precise transfer of colour spaces isn't a hard requirement for a successful image migration at SB. Meta-data for images are normally stored beside the images, so the transfer of EXIF, IPTC or similar meta-data blocks has also not been an issue so far.

The quality of a migrated image is thus measured by a quantitative comparison of pixel values. As the target format for conversion is Baseline TIFF, such a comparison can be done mechanically with a perfect/faulty reply. Note: This does not guarantee a standard-compliant result with our current tools.

Practical migration problems with images have so far been limited to a batch of archaic TIFF files, where the challenge was to find a reader capable of reading the format.

Almost all sound at SB is stored in either Broadcast Wave Format (BWF) or Pulse-code modulation (PCM), uncompressed in either case. Downsampling of 16KHz BWF has been considered for economic reasons, but tests have so far revealed that the quality suffered too much. Some old projects have sound stored as MP3, Eventide and Real Audio, the latter two being proprietary formats. Migration to BWF is currently taking place for Eventide, using proprietary software.

Danish television channels are currently being real-time digitized by SB, with 100 TB of MPEG 1 & 2 files on disk, growing with a current rate of 70 TB/year. Some TV-stations deliver recordings as files in various proprietary formats. Migration from the proprietary formats to MPEG 1 or 2 is expected, but not performed yet.

Tools in use

The migration tools used at SB has been selected ad hoc, with no formal procedure for quality assurance of the programs. Whenever possible, stable open source programs have been selected.

For image migration, the open source program suite ImageMagick has been used extensively. In a single case with a collection of a few thousand TIFF files, the closed source freeware Windows program IrfanView was used to extract a specific layer from the files. The TIFF files might be a worthy package for the test bed.

The migration of sound from Eventide tapes to BWF is an example of a worst-case scenario. The Eventide tapes store sound in a proprietary digital format with no accessible specification. The software for extracting the sound had to be created by the company behind Eventide, which abandoned the format years ago.

For future migration, the tools Sox, MEncoder and FFmpeg are candidates, as they are currently used with success for generating sound and moving images for presentational purposes. SB has only performed ad hoc testing of these tools. The tools are all open source and are very widely used.

Name Tool/Service	**Creator**	**Purpose**
ImageMagick	ImageMagick Studio LLC http://www.imagemagick.org/	Conversion and manipulation of many different raster image formats. ImageMagick is the de facto tool for image conversion at Statsbiblioteket. ImageMagick is under a GPL-compatible license and is multi-platform and command-line driven.
SoX	Open Source (Chris Bagwell is current maintainer) http://sox.sourceforge.net/	Conversion and manipulation of many different sound formats. SoX is used for simple WAV downsampling at Statsbiblioteket.

Name Tool/Service	Creator	Purpose
		SoX is under GPL and is multi-platform and command-line driven.
MEncoder	MPlayer Team http://www.mplayerhq.hu	Conversion and manipulation of many different movie formats. Statsbiblioteket uses MEncoder for making presentational copies of high-quality movies. MEncoder is under GPL and is multi-platform and command-line driven.
Ghostscript	Open Source http://www.ghostscript.com	Reads and writes both Postscript and PDF, so can act as a converter. GhostScript is under GPL and is multi-platform and command-line driven, but has a number of GUI front-ends.
Dia	Open Source (Lars Clausen is the current maintainer) http://www.gnome.org/projects/dia	Reads and writes a number of vector and pixmap images formats, including SVG, DXF, XSD and WMF. Dia is under GPL and is multi-platform. It is primarily a GUI tool, but can also do batch processing from the command line.
Zamzar	Web site http://www.zamzar.com	Reads and writes a large number of formats, documents, images, audio and visual
ps2pdf	Open source tools Installed as standard in many Unix systems	Specialized conversion from PostScript to PDF on the command line
Gimp	Open source application GPL, common on Linux, works on Windows, too.	PhotoShop-clone that reads and writes many image formats. 8-bit channels only. Scriptable in Perl and Scheme, can be run without a UI. Weak on preserving metadata.
GraphicCon-verter	Shareware application Runs on Mac.	Image converter and viewer, GUI only
NetPBM	Open Source application Unix tool, GPL.	Command-line suite that uses several intermediate formats (generic but space inefficient) that allows Unix shell pipelines for migrating and transforming.
dvips	Open Source application Part of TeX installations.	Converts .dvi (Device Independent) files output from TeX into PostScript.

2.2. Summary current experiences and problems encountered

2.2.1. Experiences

Basic principles and level of experience

The institutions described in this chapter need to preserve a great variety of digital content. The preservation policies and use of conversion tools differ accordingly. Some digital archives are based on only preserving high-quality masters in stable file formats (mainly KB-DK). And some institutions require that objects submitted for ingest have been converted to one or few formats before they are considered for inclusion (ONB). Others prefer to archive a standardized version of specific objects in non-standard formats next to the original objects. Finally, only one archive has faced the problem of performing migration of objects in the archive (SB-DK).

The experience with migration tools of the participants in the PA/4 workpackage varies from no experience yet to the use of migration tools in operational workflows. Some institutions have tested specific migration tools or consider using specific migration tools for near future use.

Objects and tools

Institution	*Type of digital objects*	*Type of tool*
SB	*Moving images, images and sound*	*Open source or shareware (moving) image conversion tools, sound conversion tools*
KB-DK	*Images*	*Open source and proprietary image conversion tools.*
ONB	*Text-based documents*	*Proprietary software for conversion to PDF/A.*
KB-NL	*Text-based documents*	*Proprietary software for conversion to PDF/A.*

Microsoft, as the vendor of the MS Office suite, provides licensed conversion programs that offer conversion of older Office versions to newer versions (Office 97 documents and later versions to Office 2003 documents and Office 2000 documents and later to Office 2007 documents).

Additionally, an Office 2007 suite add-in offers the option to save Office documents to PDF and XPS.

For now, all the Microsoft conversion programs focus on single file conversion. However, Microsoft stresses the importance of conversion of large numbers of documents and will explore the possibilities for batch conversion programs.

2.2.2. Problems encountered

Some general conclusions can be drawn from the survey among the PA/4 participants

Most image conversion tools work well, with the exception of programs that target TIFF input or output. The TIFF format is somewhat more complex than most other image formats, which probably is the reason why there is a higher incidence of unsuitable tools in this area. Much the same problems are reported with tools for the JPEG2000 standard from other institutions, and most likely the cause here is also format complexity, but these reports are preliminary and currently unreported in a scientific forum.

Quite a few problems arise when converting proprietary moving image formats. The problems appear to be caused by format obsolescence and DRM issues mainly.

Conversion of text-based documents to a standard archiving format such as PDF/A is not as simple as it seems. Not many tools exist that are able to create a PDF according to the PDF/A specifications.

Two main points of interest can be distilled from the general conclusion from the tools survey within PA/4.

1. The first point of interest is the fact that not many of the currently existing conversion tools on offer have the purpose of being used for digital preservation. Conversion tools that will be used for digital preservation purposes need to meet certain requirements. Important characteristics of conversion tools are stability, accuracy, ubiquity and their options for conversion. Conversion to (open) standard formats is preferred by most cultural heritage institutions. The experiences with existing migration tools are hence somewhat mixed – some good, some bad. The general trend seems to be that institutions are rather conservative in their choice of operational tools, probably due to the fact that it is often difficult to find good, “all-round” tools. As conversion will often be performed on large, and very diverse collections, the demands on the tools are (as noted) significantly higher than for other, more common usage scenarios that most tools are supposedly intended for, particularly with regards to the ability to accept somewhat malformed or non-standard input and still generate well-formed output.
2. The other main point of interest is the testing of conversion tools for digital preservation purposes. A common problem appears to be a dearth of tools for automated QA. In the cases where automated QA is employed, is must be described as shallow and spotty, and not comparative. In effect, only some types of objects are subject to automated QA, and they are only tested for validity, not for whether they are a satisfactory conversion of the source object. The only known cases of comparative QA have been performed entirely as manual operations, and appear not to have been exhaustive.

While the experiences described in this chapter are not representative of the whole of the digital preservation community, it will be important for PLANETS as a project to collect and analyze both positive and negative experiences, as both will help to develop a better understanding of the (general) characteristics of a migration tool useful to the digital preservation community.

3. State of the Art and challenges

In this section, we present how object transformations are currently done, and highlight some of the issues in the current approach.

3.1. State of the art

The transformations that have taken place so far in the institutions interviewed have been characterized by 1-to-1 conversions with heavy dependence on file formats and existing tools, with only spotty QA. In this section, we describe this approach in more detail.

3.1.1. File formats

A *file format* is a standard for encoding some (related) items of information as a sequence of bytes. A file format may not be publicly or even explicitly documented (e.g., some Word file formats are just dumps of internal state), nor does it necessarily specify a meaning for its elements (e.g., XML merely describes how elements are arranged and tagged in a verifiable way, not how they make sense). The expectation is that information written in a given file format can be read later to retrieve the same information (where "information" is widely defined to include metadata, data, interrelations between data, presentation of data, and more).

Frequently, we talk about a family of file formats as one, such as Word or PDF. This is a dangerous but convenient way to handle the fact that requirements and available resources change over time, and so do the resulting file formats. Some formats attempt to provide backwards or even forwards compatibility, but not necessarily flawlessly. Other formats may change their structure entirely from version to version, especially formats that are native to a particular application.

In practice, even if a precise description of the format is available, differences will emerge in the details of the file format. This may be because the description doesn't describe some unforeseen situations, because implementers don't have the resources to fully and correctly implement a format, because the format description is faulty or misunderstood, because there are bugs in the implementation, because the format is used for purposes beyond the original intention, because competing companies want to exclude each other from the market, or for many other reasons. Thus, an important part of understanding a file format is knowing what variations to expect, how to notice them, and automatically validate the compliance with the standard.

3.1.2. 1-to-1 conversions

A *1-to-1 conversion* takes a single input file at a time and outputs a single file that (hopefully) contains the same content as the input file. It is usually performed in order to "upgrade" to a more capable format, whatever that new format may be. "Capable" here depends on the objective of the conversion. For objects of low complexity, this conversion can be fairly risk free if the target format is at least as capable as the source format (e.g. metadata is preserved). Common cases of objects that are likely candidates for 1 to 1 conversion are bitmapped images with simple colour models (GIF, BMP to TIFF), uncompressed sampled audio to another sampled audio format (WAV to FLAC) and raw text to more modern text encodings (7 bit ASCII to UTF-8).

These conversions usually present few challenges to the operation of the rest of the archive in terms of indexes etc., except when internal dependencies exist between the objects in the archive (e.g. archived web pages), but that is the case for any conversion of objects.

3.1.3. Superformats

In some areas, a viable approach to 1-to-1 conversions is using *superformats*, one or few formats that are sufficiently capable to represent all digital objects of a given class (i.e. bitmapped images,

sound, sampled video...). One existing example is the NetPBM suite, which uses an internal, ASCII-based format to convert between a number of different formats. Typically, the superformat is not used as a preservation format, but as an intermediate step to reduce the number of conversions that need to be made. The superformat is thus free of the normal constraints of having to be efficient in terms of both space and time, but can be designed to contain all the features found in related formats in an easy-to-program-against way. While this would have all the benefits of the 1-to-1 strategy and reduce the number of active formats, this strategy is unfeasible as a general solution, as some important types of digital objects are so complex as to make the creation of appropriate superformats impossible. The typical example is text documents, where even a single format can be of alarming complexity, and a format that combines the features of all known text document formats would be impossibly complex.

3.1.4. Manual QA and format validation

Quality assurance (QA or “validation”) is currently done in one of three ways: By carefully examining the results of a few files and assume that the tool works as well for the remaining files, by manually inspecting as many results as possible for obvious errors, and by using another tool to verify the validity of the output files. Automatic verification that a transformation actually transformed the original object faithfully is lacking, and significant resources have been spent on manual quality assurance.

One of the only examples of a current QA tool in current use is JHOVE. JHOVE is capable of determining if a certain file is a valid example of some file format for a rather limited list of formats, and can also perform some characterization tasks. As an example, it is currently used at KB-DK as a general tool for validating TIFF files as valid, and will in the near future be included as an automated QA step in KB's DOMS (Digital Object Management System), both for validation and characterization, specifically for checking that image masters are stored as valid, pyramidized TIFF files.

The use of JHOVE seems typical for the cases where automated QA is possible at all. A case where a tool for the same usage scenario is desired, but not currently available is validation of PDF/A files in archives where PDF is the preferred master format for documents.

In conclusion, currently automated QA coverage is fragmented, i.e. covers few formats, and not capable of comparative QA.

Format validation is a well-known technique, and can help avoid some systematic errors in transformations. Many common formats have tools for validation, and some, like XML, have systems where formal specifications can be written in a way that can be used for validation, such as DTD, XML Schema and DSD. This can be a valuable resource when programming a tool or when accepting input from other sources, but for validating transformations, it gives only one small part of the story.

3.1.5. Files are central

The common transformation approach currently used is based on taking individual files and converting them from one file format to another, possibly using a larger but more generic intermediate format. Informal sampling, format validation and manual inspection are used to check the quality of the transformations. In the next section, we shall examine some of the problems in this approach.

3.2. Challenges

This paragraph presents several points of interest in the context of transformation of digital objects as a digital preservation activity.

3.2.1. Digital object complexity

A common problem is that we lack a measure of the complexity of a digital object, as this often has a major influence on the effort or other expenditure required for migration, validation, identification or characterization. As an example, the "Life" project [4] attempted to predict the cost of archiving various digital objects, and assigns a "complexity" rating ranging from 0.0 to 1.0 to an object based on its file type. This value is apparently assigned ad hoc (but not arbitrarily).

Complex features

Complexity of a file format comes in many forms. We list here a selection of these to give a taste of what kinds of problems one can expect to encounter in digital objects:

- Containers: Many objects can be considered to contain objects, either explicitly (zip archives, TIFF files) or implicitly (Word files, PDF files with embedded fonts). The contained objects may be simply contained or part of a reference structure that could also include external objects. This issue is explored in more detail below, as it has an influence on how to handle multiple files.
- References: Akin to the problem of containers, references can be internally to parts of a file or externally to related files, system resources, remote files or even dynamic content.
- Structure: Some formats are only loosely structured, like plain text, some attempt to follow a formal structure but fail, and some are well-formed according to some format specification. The structure itself may be simple, like key-value pairs or strictly nested forms, or may allow overlapping areas, open-ended definitions or even recursive definitions.
- Documentation: Quality and accessibility of specifications vary significantly, and errors can have far-reaching consequences, but can also be systematic enough to handle automatically. We explore this issue further below.
- Identification: Does the document start with a unique identifier that specifies what format to use for reading the file? Typical approaches are magic numbers or XML namespace declarations. While an identifier can help finding the right tools, ambiguous or misleading identifiers can cause confusing errors.
- Human-readable structure: Having a structure that can be read without complex tools aids debugging and error checking, but typically increases file size, and can cause wrong associations.
- Human-readable contents: Some formats contain data in a form that can easily be checked by humans, e.g. the NetPPM formats. Again, this is typically more verbose, but aids manual checking. Binary, compressed or encoded data is dependant on similarly complex tools for investigation.
- Metadata complexity: Metadata, when embedded in an object, can be in more or less complex forms, just like the data. It is however frequently hidden from normal inspection with common tools.
- Invisible data: Version information, edit tracking, reviewer comments and other internal information can be invisible to casual inspection yet be an important part of the document in the long term.
- Encrypted or hidden components and protection information: Formats like PDF that contain information on what can be done to it cause additional complexity, both because they can affect the behaviour of viewers, and because it is an extra meta-layer of behaviour that can be difficult to transform.

Sampled vs. interpreted data: Sampled data are usually easier to understand and validate than data that relies on interpretation and execution of complex code.These are all somewhat orthogonal complexities, in that a format may allow one form of complexity but not another. When discussing the complexity of a file or a format, it is preferable to complexity as a number of measures rather than one “average complexity” number – some forms of complexity may be easier to handle in given systems than others.

Measures of object complexity

One can imagine measuring object complexity from several perspectives, such as:

1. **Ad-hoc and informally.** Here objects could be assigned a score of "simple", "medium" and "complex" based on simple heuristics such as file format. Such a measure is neither quantitative, precise, absolute, stable or task-neutral, but is commonly used informally ("XML is relatively simple to process, PDF is hard", "We can process Word 95 documents, but we can't process Word 2003 documents").
2. **Cost/benefit.** This is the approach used in the Austrian utility analysis study [5] and the Life project [4]. Conceptually a score of 0 to 1 is assigned based on the desired operations to perform on the object. The score is only directly applicable to the task that is considered, but can provide an indication of the difficulty of processing the object for other tasks. It is also not stable, as new tools (or the disappearance of a previously existing tool) may affect the score.
3. **Scientifically.** One could imagine assigning a score of 0.0 - 1.0 for an objects complexity bounded by a value of 0.0 for a file that presents no complexity at all, such as a "semaphore" file that only has one bit of semantic content (present/not present), and 1.0 for a (hypothetical) "incomputable" file. The problem with this approach is not to determine the end points of the scale, but how to score objects reliably and accurately in the range.

Clearly, the most desirable measure is quantitative, precise, stable, absolute and task-neutral, but that seems to be impossible to achieve in the absence of a scientific (SI-like) measure.

Presumably a cost/benefit measure could be developed in the context of a given preservation plan for a file format, or a specific object, in the context of a PLANETS architecture. Such a measure could be useful for preservation planning, but will largely depend on the task (or "goal") for the preservation plan. Hence, it is not something that can be developed further in the context of the PA/4 work package.

Therefore we can conclude that we are currently unable to measure object complexity in an absolute, scientific manner, and only in special cases in a cost/benefit manner, which may not be applicable in a general sense. We are therefore often reduced to use ad-hoc measures.

3.2.2. Containers and relationships

A common problem is that some digital objects are containers, that is they contain several other digital objects, or should be approached as such. Common container formats are e.g. ZIP archives and PDF documents, but any object that contains both data and metadata can also be considered as a container object. A few common cases are:

1. **Typed containers.** These are restricted in what they can contain, and are usually not viewed as containers at all. An example is a TIFF file that can be viewed as a container for a metadata stream, an image stream and a colour space stream, or even several units of related metadata, image, and colour streams. Hence, when processing TIFF files one must always assume that operations performed must be applicable to N embedded objects rather than just one, or create an artificial sub-typing of TIFF files as "single image" or "multiple image".
2. **Pure, untyped containers.** These objects are developed specifically for collecting bundles of objects in one object. Some have little or no associated metadata beyond the necessary information to extract the objects (e.g. ZIP files), while others contain additional metadata on a per-bundle or per-object basis (ARC/WARC files). These containers present few challenges, as they rarely add complexity and their "containerness" is usually known and accounted for.
3. **Incidental containers.** Some objects can contain other objects, with or without restrictions on the type of the objects they contain. Examples of this are Word files that can contain just about any object, including whole, executable files, or PDF files. These are usually not considered container files per se, but this aspect of their functionality should be addressed in a complete preservation architecture. Hence, they present the largest problem—they are

complex objects in their own right, their "containerness" is easily (and frequently) overlooked, and complex interdependencies usually exist between the embedded objects (e.g. converting the embedded images in a Word document to TIFF will "break" the document if internal references to the images are not updated accordingly).

These categories can overlap; For example ODF and Open Office XML documents are both packaged within ZIP archives, but contain significant internal structure and referencing.

A crosscutting concern for all these containers is that the individual container may not preserve information about the objects it contains, be that their type (which it may or may not be possible to infer from the contained object), or the boundaries between the contained objects. If no metadata for this is present in the container, and no outside source is available to provide it, the contents are essentially lost until such time as the situation is remedied, which may be never. Note that most containers of type 2 do contain this information embedded along with the objects, as it is essential to fulfilling their primary purpose. E.g. storing the directory structure of a ZIP file in a separate file would be dangerous, and hence the Zip format (and all other known formats designed as containers) stores this information in the container itself.

While it's a good general rule to assume that if an object is a container, it is more complex than a non-container object, it can be hard to draw a clear distinction, especially in the absence of a clear, universal definition of "container". E.g. one could consider the TIFF format a container format, albeit a very limited one, as it can contain several images and metadata in one file.

Hence one is left with three top-level choices when trying to (in part) gauge the complexity of a digital object based on it's "containerness": Whether to consider it in the case of a specific object (easier, but less useful), whether to estimate it from a sample of such objects (also easy, and possibly more useful), or "in potential", where one e.g. assumes the worst case in terms of complexity.

Container formats tend to present several challenges other formats do not. They do so both in the sense that they are often more complex to process, but perhaps more pertinent that they always require that the preservation planner takes a clear stance on whether the object should be considered as a single object, or as a collection of objects.

3.2.3. Sampled objects vs. interpreted objects

There are two fundamental models for encoding information in digital objects:

1. **Sampled.** A sampled object is essentially either a sampling of some real object (WAV from a violin concerto, TIFF scan of a book), the end product of some rendering of a more abstract object or recipe in a given resolution (44KHz WAV file produced from "play tone at 440Hz for 1 second", 600DPI TIFF image of a page of a Word document in A4 format), or born sampled (a bitmapped icon file). These formats are usually rather simple to present for an end-user (e.g. paint the bitmapped image on the bitmapped screen or printer), and can be roughly described as "largely similar to the final rendering". Also often referred to as “digitized”.
2. **Interpreted.** An interpreted object contains an idealized description of some object or process, and must be interpreted in some fashion to produce a presentation rendering. Examples are PostScript files, Word documents, SVG files, MIDI etc. These formats usually are rather abstract, and are usually not particularly similar to the final rendering. Sometimes referred to as “Born digital” objects, but as some sampled objects can be born digital (e.g. bitmap artwork), the term “interpreted” is more precise and useful to accentuate the fundamental differences from sampled objects.

Sampled formats present fewer challenges in terms of presentation, as they have fewer external dependencies, and are inherently quantified. Hence, transforming such objects to other sampled formats is often easy, and the fidelity of the conversion is also relatively easy to evaluate. Further, these objects tend to contain no or few explicit external dependencies, and few inherent dependencies.

This is not so for interpreted objects. They are inherently dependent on an, often complex, interpreter. Hence, to preserve such objects with high fidelity, the interpreter must also either be

preserved or an equivalent interpreter substituted. Sampling such objects is not a solution, as loss of fidelity will usually result (and is essentially unquantifiable), and some of these (e.g. interactive objects such as games) are essentially unsamplable.

Interpreted objects also have the unusual feature that they need not be of limited size. PostScript, for instance, is a Turing-complete programming language specialized for printing, and it is a simple task to make a PostScript file that does not terminate. The Halting Problem tells us that it is theoretically impossible to automatically determine if an arbitrary program terminates. Fortunately, such problems are in practice fairly rare, mostly the result of errors in, say, Word macros.

Essentially the problem of preserving interpretable objects is a special case of preserving live objects (here the required interpreter): To preserve an interactive (or “executable”) object faithfully, one must either preserve the entire runtime environment faithfully (emulation, virtualization), or substitute a (preferably provably) equivalent one, as in the case of running a Java program on a different JVM from the one it was originally developed, tested and targeted for.

Interpreted objects are uniquely useful; they allow for tasks that sampled formats do not, such as specifying the end result of a rendering in an abstract way, allowing for rendering with arbitrary precision (rendering device independence), interactive behaviour etc., and avoiding them entirely is clearly unfeasible except for very specific cases—and here a price of loss of fidelity is usually unavoidable.

We have no current general solutions to the problem presented by existing objects of the interpreted type. This is in contrast to the case of sampled objects, where re-sampling, possibly in another format, is a general solution, with well-known properties, both in a practical and theoretical sense. Hence, we are left to treat every kind of interpreted object as a special case, with a special solution.

The problem is both explicitly and implicitly acknowledged by the current trend towards using open standards engineered specifically for stability, flexibility and a minimum of unquantified, external dependencies. Examples are the use of virtual machines and well-defined runtime environments for modern programming languages such as Java and C#, and the transition from complex, binary formats such as the original Word data format to XML-based, open standards.

3.2.4. Encapsulation

The current consensus is that tools should be accessible as web services. The Planets test bed uses JBOSS as the main application server, which implies Java as the development platform. This is not a hard requirement, but for the discussion below it is assumed that Java will be used.

It is envisioned by the test bed people that access to data objects will be though standard file access. The mapping underneath to repositories will be transparent to the user.

Unfortunately, we are not aware of any existing projects that wrap tools for preservation (identification, characterization, conversion, migration and QA) in a service oriented architecture (SOA) framework. Hence, we can't learn from experience (ours or others), and must start from first principles.

Note that we only address wrapping to produce a single transformation. The architectural challenge of combining several transformations is not addressed here, but it can likely be achieved by using a combination of the design patterns Command and Composite [7].

Basic interaction

One way of dividing tools is interactive and non-interactive. Interactive programs are typically GUI-oriented, with rich graphical interfaces. Non-interactive programs are almost always command-line oriented.

The interactive programs present by far the greatest challenge for encapsulation. Some programs supports scripting, but the capability can’t be assumed. Java itself has no native way of sending simulated key presses and mouse movement to an interactive program. Third-party meta-programs for controlling other programs by scripting exist. A double encapsulation with Java controlling such

programs, which will again control a tool, seams feasible. However, scripting an interactive program is non-trivial, the handling of exceptions being one of the big hurdles.

Command-line driven non-interactive programs are, by comparison, easy to encapsulate. The first prototype encapsulation web service will focus on command-line driven tools. The main interfaces to such tools are the command-line parameters and standard input, output, and error streams. While Java has support for these, its support is minimal, and we may have to consider finding or developing a better Process interface.

Note that some interactive programs are scriptable, e.g. from the command line, or through an API. In the experience of the PA/4 participants, this is not always a reliable way to access these programs, as e.g. error conditions may result in dialogs requesting human interaction. This is a known problem when aggregating Open Office and various programs in the Microsoft Office suite.

UNIX-like signals are used in some programs to force restart, suspension, shutdown, or other control features. While such signals are available in most system in some form or another, Java does not support signals very well, being unable to send anything but a shutdown signal. Some native code implementation may be required to handle this.

Crashing and halting

Crashing of an encapsulated native tool is normally not a big problem. The obvious handling is to log any errors, mark the input data as problematic, and proceed with other executions. A more solid implementation could stop the overall processing if a given number of executions resulted in crashing.

The halting problem occurs when an encapsulated tool keeps on running. It is impossible to make a general case for whether the tool just needs more time or whether it is stuck, so that it will never complete its task. Monitoring the output of the tool cannot generally be used as an indicator of activity. One example would be the transcoding from above; if such a transcoding utilized two-pass encoding, the tool would take a long time before starting to produce output.

The standard way of handling such a situation is to provide a timeout, but it can be difficult to find fitting values for this. Consider a transcoder for moving images. If the input is a 30-second clip, a timeout might be 2 minutes, but is the input is 24 hours of footage, a much larger value for the timeout must be used.

The problem of long timeouts etc. relating to long running commands is well-known in the SOA community. Here this type of commands is referred to as Long Running Transactions (LRT), or "sagas".

Usable values for timeouts could possibly be estimated from the size of input data, but this method is not likely to be generally applicable.

Platform differences

Widespread systems with Java support, such as Linux, Windows, OSX and Solaris, differ in their execution of command-line driven tools. Linux, OS X and Solaris adhere (to various degrees) to the POSIX standards, and Windows XP and 2003 can be made core POSIX compliant with the use of an optional package from Microsoft.

While some differences are handled by Java's runtime, other differences need to be handled explicitly. Legal characters for filenames differ and arguments differ between both systems and program versions. Further, there are several more subtle differences between platforms that become evident when developing and debugging complex Java programs, such as locking policy for files (on Windows, any file access results in a write lock on the file, whereas this is not the case on UNIX).

IO handling

Basic IO with a single input and a single output is typically handled through files or streams. Error messages are typically streamed to stderr. Java encapsulation with streams is part of the standard Java API. For streams, some way of detecting and handling excessive amounts of data must be present.

Log files and temporary files are problematic, with regard to encapsulation. In case of program errors, some programs dump debug information as a log file. If this program error has not been encountered during tool testing, the location of the file might not be known. Temporary files might not be cleaned up if a tool crashes or is just not well behaved. Error-feedback and file cleanup can thus be difficult.

Upon termination, programs are expected to provide a return-code. The meaning of this return value is not well defined, and several ad-hoc standards exist. Hence, if an error occurs, e.g. a tool performs a migration that was partly successful, there is no generally applicable way to infer this form the return code—it depends on the tool in question.

Resources

Controlling resources such as disk space, RAM, CPU and network is hard, if not impossible, when using Java to execute native programs. A viable solution might be to run the whole web-service in a virtualized environment, such as VMWare. Such a solution is outside of the scope of this document.

3.3. File format difficulties

A number of difficulties arise from the fact that all digital data must be encoded in some way or another, and the exact encoding is often left up to programmers without sufficient time or knowledge to make a consistent, well-documented and properly implemented file format. At the same time, file formats are being used as competitive tools: if your program can read files that your opponent's program cannot, your program has a competitive advantage.

The below issues of file formats should be taken into consideration not just when picking a file format to transform to, but also when considering different transformation tools and when building new tools.

- The formats involved may not have any specifications. Many formats have evolved over time as programs needed to store more information, and the specifications may not have been updated, or even created in the first place. This may leave reverse engineering as the only viable way to figure out a format. Having the source of programs that use the format helps, or even having a binary-only runnable program that can read or write the format.
- The specifications may be proprietary. The original file format used by Microsoft Word is one of the most widespread document formats, but the specification is a secret, if it even exists.
- Specifications are often written in prose rather than with formal definitions. This leads to ambiguity and differences in interpretation, and disallows automatic generation of transformation programs. This is often the case even for file formats for programs, and is even more common for data formats.
- Specifications may be so large as to be nearly impossible to implement. For instance, the Microsoft Office Open XML specification is over 6000 pages long.
- The specifications may have omissions, implicit assumptions or references to proprietary information. The Office Open XML format, for instance, defines some fields by referring to how Microsoft Word behaves, and many XML specifications describe only the XML structure but omit the description of what the data means, focusing on syntax rather than semantics.
- The specifications may be internally inconsistent or contradictory. This is particularly likely if the file format has evolved over time to meet new needs with insufficient care taken to keep the parts consistent when they change.

- The programs involved (original writer or transformation program) may have bugs in the way file are read or written, or the programmers may have misunderstood the specifications. This can show up as systematic errors in the original files that could be recognized and handled as a special case.
- The original file may rely on (unpublished) extensions of the specifications. Whether or not this is an intentional strategy, it leaves us with files that do not follow the published standard.
- The transformation program may have implemented only a subset of the specifications. This is typical for programs where reading file formats is a secondary purpose, where some importable formats may have parts that are irrelevant or meaningless for the program, or resources may not have been available to implement all parts.
- The original format and/or target format may not match the internal model of the transformation program. For instance, there are several different models for Bezier curves that don't quite map to each other, or bitmapped images may use different, partially overlapping color spaces. This is something that a good transformation program can detect and warn about, possibly even give an estimate of how much quality was lost.
- External requirements such as fonts, style sheets etc. may be missing or different. In some cases, substitutions may take place without warning.
- System functions like font hinting may differ between the system that made the original file and the system that the transformation takes place on. Even more so than substitution of fonts, this may cause changes that the transformation program does not warn about.
- Internal limits or limits of the target format may be reached. Such limits are not always mentioned in the specifications or tested by the programs involved, and can cause subtle errors as well as crashes. Typical problems are integer overflows and truncated strings.

These issues should not be taken to say that file format problems make all transformations impossible. Rather, they should be seen as a warning list for picking formats and tools, and particularly for programmers implementing new tools, who both need to beware of falling into similar traps and should consider the implications of the issues on the selection of test data and handling of unexpected errors. It also goes to show that merely validating a file against a formal syntax specification, even when possible, does not necessarily determine whether or not the file can be transformed.

3.3.1. Conclusion

In this section we identify and analyze several issues related to object complexity and the challenges that is presents to a digital archive. Unfortunately we can no identify known or potential solutions to these problems, and are reduced to referring them in order to make sure they are not overlooked in current and future archives.

Encapsulation - and to some extend file format difficulties – propose challenges concerned with the transformation tools that will be used for digital preservation purposes. These challenges will need further research before solutions can be specified.

4. Novel approaches

This section describes our approach to transformations as preservation actions, both theoretical and practical.

4.1. Many interpretations model

4.1.1. The multiple-stream object model

A useful view of a digital object from this point of view is the "multiple streams" view. From this point of view a digital object contains one or more "streams" of information. Depending on the intended use of the object, some or all of these streams may be considered.

As a concrete example, consider a simple HTML document:

```
<html>

  <head>
        <meta content="text/html; charset=ISO-8859-1" http-equiv="content-type"/>
        <meta name="description" content="A page about hellos, and more hellos"/>
        <meta name="keywords" content="hello, reetings, test, html, helloworld”/>
        <title>My hello world page</title>
  </head>
  <body>
        <h1>Hello world!</h1>
        I greet you, my newly found friend!!!!!!
        <a href="http://www.hello.org/more_hellos.html">Click here for more.</a>
  </body>
</html>
```

This object contains several streams of information, including (but not limited to):

- A formatted text stream (title, headline, paragraph text)
- An unformatted text stream (the above, with formatting stripped)
- Several metadata streams (HTML type, content metadata, keywords metadata...)
- A document structure of title and text with embedded link.

When rendering the document in a web browser, the metadata streams are usually largely ignored, and are invisible to the recipient. In this case, the formatted text stream is usually the only important stream. When a search engine robot (say, the Google web crawler) processes the document, the metadata are usually at least as important as the textual content.

4.1.2. The multiple-interpretation model for object conversion and migration

The analysis on the previous section analysis suggests two strategies for migrating or converting an object:

1. 1-1: map objects 1-1 to new formats that are preferably at least as capable as prior format. Dangerous and not always possible.
2. 1-n: map objects 1-n to some or all currently known useful transformation including original object.

The 1-1 strategy is theoretically the preferable one, and represents the traditional approach to migration: Moving the object to a more capable and/or durable format. The danger is that there may not be any one format that is suitable for the migration, and that the migration to a (theoretically) more capable format may induce loss of fidelity or inclusiveness that is not immediately obvious. This risk may be lessened by also preserving the original object, but if migration is performed to "escape" a dying format this added security is largely theoretical. The advantages of the 1-1 strategy are simpler processing (fewer formats are used), and lower storage and processing requirements (less data is stored in fewer formats).

The 1-n strategy implicitly acknowledges the risks of the 1-1 strategy, and can be seen as a superset of it. The 1-n strategy does not rule out including a conversion to a presumably (at least) equivalent format as one of the n representations preserved, but may also for strategic reasons include conversions to less capable, but more stable, formats as an "insurance" against future issues of format brittleness.

4.2. Aspects

The multiple interpretations model of digital objects requires new terminology and tools. In this section we describe a tentative proposal for a new nomenclature for multiple interpretations of objects, and our evaluation of the XCL technology in this context.

4.2.1. Definition of aspect

An aspect is an abstract view of (a subset of the) information in one or more digital objects. Example aspects could be ICC profiles in JPEG images, metadata in MP3 files, or page breaks in Word documents.

4.2.2. Why aspects?

The problem that aspects attempts to address is this: Unlike physical objects, digital objects may contain layers of information that are not immediately obvious when the object is rendered in a way that emphasizes other layers of information in the object. Some common examples are TIFF files (embedded metadata is not visible when viewing the TIFF file as an image) and Word documents (editing history and author information is not visible in a normal print or editing mode). Hence, it is possible to consider the digital object as the sum of these “aspects”, or to consider the aspects separately. So the concept of aspects provides a more precise and flexible view of a digital object than "file format". An aspect can loosely be described as "a way to interpret an object". An aspect should be quite narrowly defined, e.g. "Editing history of a Word document" or "Metadata for a TIFF file". Hence, the aspect concept maps very well to the multiple streams object model and the multiple interpretation model for object conversion.

To sum up, the "aspect" concept is introduced to deal with several fundamental problems, primarily:

- The "file format" concept is imprecise and diluted. E.g. "is a pyramid TIFF file a TIFF file"?
- Objects can't be classified naturally by using a tree structure. A text file is a byte stream, but a byte stream can also be interpreted as text file (even if it does not always make sense to do so, e.g. interpreting an executable file as a text file).
- A description of a "file format" or some properties of a "file format" quickly becomes extremely complex. E.g. a Word document can contain text, metadata, objects (that in turn can contain metadata, objects...). Aspects can help reducing the need for elaborate or recursive descriptions.

Hence, using a concept of crosscutting, narrowly defined aspects is a much more workable definition, both from a theoretical and a practical perspective.

The concept of aspects does not exclude the description of a digital object by its five main characteristics: content, structure, context, appearance and behaviour. These five characteristics can be defined as overarching main aspects of a digital object.

Note that the concept of aspects neatly separates metadata from data—both are accessed through their own aspects.

4.2.3. Aspect content

If an aspect is "the interface" to some information in an object, the aspect content is then the data. If your aspect is e.g. "TIFF version 3 header", the aspect content is the information in the TIFF V3 header of the object.

Usually the aspect content ("data") is the input to some operation.

4.2.4. Fidelity and inclusiveness

With an aspect-oriented view, the “quality” of a transformation expands to become a two-dimensional measure: one dimension is which of the existing aspects are included in the transformed objects, and the other is how well they are transferred. These two dimensions are called *inclusiveness* and *fidelity*, respectively, and are described in more detail below.

Inclusiveness

Inclusiveness is not presently a well-defined concept in digital preservation, but we found a need for a complementary concept to fidelity.

In this deliverable, inclusiveness refers to the degree to which a conversion from one format to another (or several) format(s) preserves the information streams and capabilities of the original object, or in terms of aspects how many aspects of the source object are included in the transformation. Hence, a conversion of a Word document to PDF will usually have a high degree of fidelity with regards to printed output, but will often have a low degree of inclusiveness, as many capabilities and information sources in the original Word document are lost in the transformation, e.g. author notes, editing history, editability, live embedded objects such as Excel spreadsheets...

Fidelity

While the word "fidelity" is generally used in the digital preservation community, we use it in a very specific sense here, namely in conjunction with the “aspect” concept.

Briefly, when we discuss "fidelity", we mean "the degree to which a given aspect of a digital object in one format is alike the equivalent aspect of the same object in another format". Hence, we can discuss the fidelity of an A4 print aspect of a word document vs. the A4 print aspect of the same document converted to PDF version 1.6.

Hence, when discussing "fidelity" in the Word/PDF example above we are not concerned with information streams or capabilities that are lost in the conversion to PDF (e.g. editing history, author notes, editability...), as they are not relevant for the comparison of fidelity—here whether the printed output from the two formats is sufficiently alike to make the conversion acceptable for its intended purpose.

Note that fidelity can usually not be assessed without using some viewer or renderer for the object. For the vast majority of object conversions, comparing the “naked” bitstreams provide no information on the fidelity of a conversion. Hence, a discussion or evaluation of fidelity of a conversion will almost always be in the context of one or several viewers for the objects in question, although this assumption is often implicit.

For a detailed discussion of how to assess the required and actual fidelity of a conversion, see the Austrian utility analysis study [1]

4.2.5. n-to-m conversions

This type of conversion is in the experience of the PA/4 participants relatively rare. The only example we know of is compiling (La)TeX files to several output formats (dvi, ps, pdf…). They should easily be accommodated by an archive that already supports 1 to n and n to 1 conversion, but a “one object per record” archive would require significant architectural modification to support n to m conversions.

4.3. Semi-automatic QA and XCL

4.3.1. Computer-aided inspection (semi-automated QA)

A case of semi-automated QA is manual inspection and comparison of the results of a conversion. A specific, realistic case would be conversion of Word documents to PDF.

While some aspects of the conversion (page breaks, contents...) can be verified automatically with relatively simple custom programs given the required knowledge of interfacing with Acrobat Reader and Word, some elements of the conversion can only be verified manually, e.g. quality of the embedded images, minor typographical details such as markers for footnotes and references etc.

Another example could be conversion of images from TIFF to JPEG2000.

For these tasks, even a simple tool that allowed easy side-by-side comparison of the original and converted objects would streamline the process of validation immensely, and possibly allow the task to be offloaded to less technically qualified personnel.

We are currently unaware of any tools specifically developed for this purpose. However, single-object browsers are a well-known technology used in many archives for general revision and QA, and extending these to side-by-side comparison would appear to be a relatively simple task.

We know of at least one case where just such a tool would have resulted in large time savings (comparison of Word original with PDF conversion at KB-NL).

While this area is not explicitly a part of the PLANETS field of interest, we advocate that more work is done here.

4.3.2. Reduction to simpler formats for "fuzzy QA"

As mentioned, in cases where some loss of fidelity or inclusiveness is acceptable, QA becomes problematic—especially automated QA.

If both the original and converted format can be converted to some simpler format, these may be easier to compare. Essentially, we might be able to "downgrade" both the source and destination object to a representation that can be compared directly, but still retains sufficient detail that the desired fidelity and inclusiveness can be considered.

For documents, one possibility is printing -- this removes functionality (e.g., links) but retains the visual aspect. Printing to a file typically uses a format that is common to the platform (PostScript for Linux, WMF for Windows, PDF for Mac OS X) using fairly standard routines (GDI printing, for instance). These files might then be comparable, though we currently have no known tools applicable to the mentioned formats.

Rather than comparing the printed formats directly, we could further reduce them to bitmaps, which are easy to compare. Making a low-resolution bitmap would give information about high-level differences, such as page wrapping having changed. See the genre classification paper from ECDL 2006 [6] for inspiration. The higher the resolution of the bitmap, the subtler a difference can be discerned. Things like font substitutions, kerning differences and details of vector graphics rendering can be seen at high resolutions. Some experimentation may give an idea of what levels of resolution are useful for the aspects that are intended to be preserved by the migration.

Note that it's not given that we will compare only one derived format—a feasible approach for complex document formats would be extracting and comparing e.g. bitmaps of printed pages at several resolutions, raw, unformatted text in a common encoding (say, UTF-8) and metadata streams for comparison on several levels of inclusiveness and fidelity.

Note that the XCL working group have independently arrived at the conclusion that comparing derived formats is a potential approach to "fuzzy QA" when no direct comparison can be performed.

4.3.3. Extraction of characteristic data for "fuzzy QA"

Many formats have tools that allow extraction of information such as word count or links. These can also be used as indicators of problems; they provide a rough indication of success (or failure) of a transformation, and indicate if there are significant differences. For example, in the migration of a word processor document, the word count, character count or character distribution (in case of words being split, e.g. for kerning purposes) would be expected to be preserved

This approach is a subset of the approach described in the previous section, only it focuses on high fidelity comparisons of very narrowly defined derived information streams.

The XCL technology, which is part of the PLANETS project, is expected to provide a general way to extract these characteristic data for some, but not all, file formats.

4.3.4. Extraction of metadata for "fuzzy QA"

A special case of the approach described in the previous section would be comparison of various metadata streams extracted from the objects.

This approach is actually more practical, as there are tools for extraction of specific metadata streams in well-defined formats for many classes of objects. An example is EXIF data extracted from bitmapped images (TIFF, JPEG 2000). Hence, the foundation for the approach exists in terms of well-known and operational tools for some formats, and it is feasible to validate and implement this approach using current tools and procedures.

As with characteristic data, the XCL technology, which is part of the PLANETS project, is expected to provide a general way to extract these metadata for some, but not all, file formats.

4.3.5. What is XCL?

XCL is a complex of technologies that is concerned with extracting and storing various interpretations of a digital object. XCL consists of two components:

1. **XCDL or "results":** Preserve as many kinds of information as can be derived from a digital object in a well-formed format
2. **XCEL or "recipe":** Describe how the various kinds of information stored in 1 can be extracted from a digital object in a well-formed format

XCL is XML based, and aims to be a "one stop shop" for handling these tasks.

We assume that XCDL can be used independently of XCEL, as it is conceptually only a container for digital objects, albeit one that is designed specifically for the purpose of storing several renderings of one object. If that is not the case, an alternative format with the same purpose could be used instead to achieve the same goals.

4.3.6. How do they relate to each other?

XCL relates quite well to the "aspect" concept. The "recipe" part of an XCL description would correspond to an "implementation" of how to provide an aspect, and the "results" part corresponds to the data that can be extracted from an object through a given aspect of it, i.e. the aspect content.

XCL does not concern itself with the actual processing of the extracted streams in the general sense, nor does it implicitly incorporate a QA process. In effect, we can view XCL as an implementation of Aspects with one operation (store aspect data) and no inherent QA requirement.

4.4. The next steps

The next steps in the PA/4 work package will consist of experimenting partly with extending/creating transformation tools, partly with how to use transformation tools in an SOA architecture and integrating transformation tools with the general PLANETS frameworks being developed in other work packages. The integration part depends on the developments in the other work packages, and so can only happen when material becomes available from them, but we can still do some more isolated experimentation till then.

4.4.1. Workflow

Potential workflows for migration have been discussed at Planets PA/4 meetings in emails between Stephen Carr and Toke Eskildsen. In this report, three probable workflows are presented.

Simple web-services

One or more tools are encapsulated as a web-service with a request-response interface that handles one migration at a time. The caller is responsible for iterating through the material that is to be migrated.

Such a workflow calls for a generic batch processor, which acts as a middle layer between the caller and the web-service with the encapsulated tool. Performing calls between web-services for each single migration might be a performance bottleneck. On the positive side, creating a new encapsulating service is relatively simple.

Batch processors

One or more tools are encapsulated in a stateful web-service. The web-service is responsible for batch-running migrations of many files and has methods for controlling the process as well as inspection.

A batch-processor is obviously more complex than the simple web-service outlined above, but the potential for fast processing is higher.

Visitor framework

In PC/4 a prototype GUI-driven characterization tool framework has been created. The frameworks makes use of the Visitor Pattern [7] and could be used as inspiration for a similar Visitor-based migration framework.

Such a framework would have several components:

- A source-repository with files to migrate
- A destination-repository which can accept migrated files
- A pool of migration tools (Java objects, potentially encapsulating native tools)
- A server responsible for controlling the process
- A front-end for the server, which would probably be a webpage

In the most basic form, the server requests files from the source-repository, migrates them with one of the available tools and stores the result in the destination-repository. This calls for a lot of file transfers for each data object.

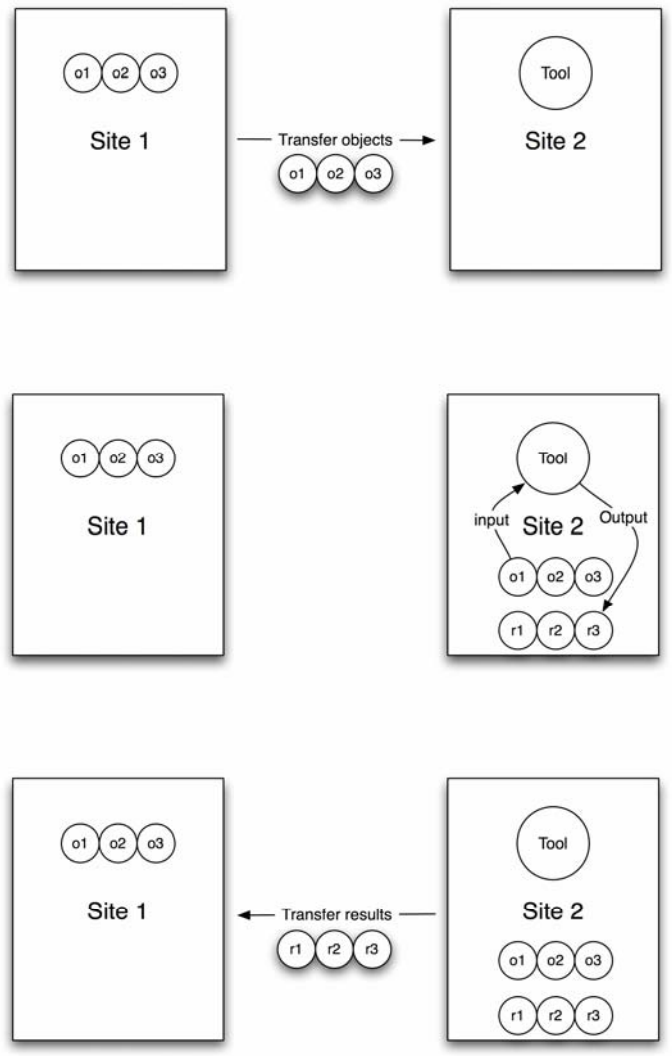
If a migration tool can be executed by the source-repository, then the server can request that the source-repository do the migration, before delivering the result to the destination-repository. This

will lower the amount of file transfers. If the destination-repository is identical to the source-repository, the performance will be optimal. The requirements for a migration tool, in order to be executed by the source-repository, will typically be that a certain native tool in installed or that the Java code is serializable.

The strength of the Visitor framework lies in the decoupling of the repositories from the server and the migration tools. If it is only to be used within a controlled environment with local access to the data objects, it offers little benefit. One downside to such a framework is the tie-in to a certain platform, which will probably be Java. Especially for wrapping "native" (compiled to machine code) tools, the Process interface in Java offers little control over the handling of signals, crashes and the like, due to Java's "lowest common denominator" nature.

Data object transfer

The potential transfer of data objects from a repository to a migration service presents performance as well as security challenges. The scenario is illustrated here:



This is the classic “web service” (or more general: “remote invocation”) scenario.

The advantage of the scenario is that tools are located and maintained off-site. This allows several sites (e.g. Site 1a, 1b, 1c…) to use the tools, and share the expense of installing and maintaining the tools.

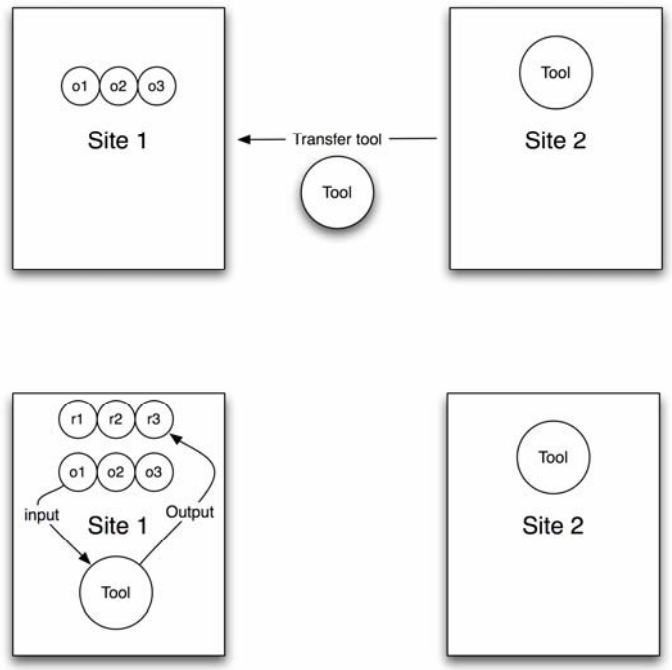
The drawback is that the data objects o1, o2… likely constitute a significant amount of data in preservation plan enactment scenarios, as do the results r1, r2…

Additionally, sending data objects over SOAP or a similar text-based protocol has a substantial CPU as well as bandwidth overhead. These protocols also often impose a maximum transaction size that is significantly lower than the size of the typical digital object stored in an archive. This can be remedied by transferring files separate from instructions through another protocol, such as FTP.

Letting the migration service request the objects means less startup-delay. Several solutions exist here, where transparent mapping of data objects to the local file system seems to present the simplest interface to the migration service.

Transformation transfer

Instead of moving the data to the program, one might consider the opposite solution. The scenario is illustrated here:



This has many advantages, as the data in most transformations will be several orders of magnitude larger than the size of the tool.

While this is feasible in a homogenous environment, such as Java, it is less so in the case of wrappings of existing tools, such as a UNIX command line program. The reason is that while it is feasible to provide the necessary environment for the wrapped tool in particular cases, especially if they (like JHOVE) are implemented in Java, it is almost impossible in the general case. The challenge is analogous to the problems presented by preserving logical access to other live objects, and the same problems arise.

In brief, unless the environmental dependencies of the tool in question can be replicated exactly at Site 1, or Site 1 already offers an identical environment to Site 2, moving the tool from Site 2 to Site 1 requires that the entirety of Site 2 is replicated at Site 1 in a transparent way, which is often a formidable challenge.

Hence, this is not attempted in the current prototype, and not a part of the PLANETS proposal at this time.

4.4.2. Prototype

A native tool encapsulation web service prototype has been developed by PA/4 . The aim has been to implement a simple web-service, as described above. The prototype at bullet points is

- JAX-WS on JBOSS, which means Java underneath
- Input from local file system
- Output to local file system
- No security handling
- Single input, single output, stateless
- Dia as native tool (multiplatform, capable of migrating between different vector graphics formats), but generic native tool interface

Interface

The interface for the prototype is based on Java's native execution capabilities. Neither Java, not the interface, holds any big surprises with regard to the parameters.

- **Program:** The native program to execute.
- **Input:** An input file for the program.
- **Output:** An output file for the program.
- **Stdout to:** Pipe stdout to this file.
- **Stderr to:** Pipe stderr to this file.
- **Args:** The arguments for the program. Stating %i and %o substitutes with input and output from above.
- **Switches**
  - **Pipe infile:** Pipe the file from input above to stdin.
  - **Stdout is output:** Pipe the output from stdout to output from above.

5. Conclusion

We conclude that transformations should be non-destructive, e.g. only be allowed to add to the current data associated with a logical object, be that metadata or digital objects. Removing old, unused formats rightly belongs under administration or data administration, nor preservation.

The transformations should be “slim”, e.g. very narrowly defined in terms of input and output formats and action performed. This is desirable for several reasons, primarily that this simplifies wrapping for SOA, and because it is easier to provide one or several simple interface(s) for a very capable tool than providing a very capable interface for a simple tool. Presumably, more capable “fat” transformations can be constructed by combining several “slim” transformations in a detailed preservation plan.

The primary current operational challenge is automated QA. Currently, automated QA is spotty (e.g. that it is only possible for a few formats), and often not thorough (e.g. often only cursory validation is performed). Worse, even in the cases where automated QA is in use, it only provides “isolated QA”, i.e. validation of whether the objects produced can be processed. What we really need is automated “comparative QA”, that actually compares the source object with the destination object after conversion.

The secondary operational challenge is, as always, to find or develop tools for providing the required transformations from existing formats to new ones with acceptable fidelity and inclusiveness, but reports of current practice from the institutions indicate that while this is a common concern, all institutions are currently able to fulfil their mission adequately. That being said, all institutions indicate that they would be able to perform better if a wider variety of high-quality transformation tools was available.

The primary architectural challenge we have identified is n to m mapping, i.e. the fact that some conversions do not map 1 to 1 in terms of source and destination files. The OAIS model does not address this, as it is usually interpreted as “one digital object per record”. This appears to be reflected in several of the systems in use at the participating institutions, in that they are not currently able to accept several digital objects as a representation of the logical object they preserve in a record.

We propose that this should be addressed, both by modifying current archives to be able to accept an n-object record, and by embracing the multiple interpretation model for digital objects.

The primary strategic challenge is (also as always) maintaining access, i.e. logical preservation. We propose the multiple interpretations model as a way to both provide flexibility (e.g., only store many/expensive interpretations for objects of known, high value) and lower risk of loss of fidelity or inclusiveness. The alternative is to be able to provide flawless conversion from every format X at risk of extinction to some new format Y that is a superformat of X, something that in our experience is not always possible, and likely increases the average object complexity in the archive over time or to accept loss of fidelity or inclusiveness at conversion.

Object complexity is mainly a practical problem of interest to cost/benefit projections for preservation, and not our area of responsibility when considered as a static problem (“the current state of an archive”), but we must point out that relying on a superformat migration strategy will likely increase the average complexity of objects, and hence the price of maintaining logical access (“the implications for future archive complexity of a superformat strategy”).

As for SOA wrapping of existing tools, we conclude that it is possible, but requires that several fundamental decisions with regards to implementation and policy are made and agreed on. Examples are error handling and file access. We point out that while migration of live preservation actions will often be desirable in a distributed SOA, as the data to process will usually be significantly larger than the live code needed to process them, it presents almost insurmountable problems when the preservation actions consist of a wrapped tool. The problem is, simply stated, that these tools in the general case rely on a runtime environment that consists of the entire computer they are installed on, which makes migration of live programs extremely hard.

We propose that XCL is a promising technology to enable comparative QA in a general way. Unfortunately there are limits to the level of support that can realistically be expected for complex formats, and therefore XCL can not be expected to solve this problem entirely.

6. References

1 The OAIS model

Overview of the OAIS ISO standard.

Available online at http://nost.gsfc.nasa.gov/isoas/

2 PREMIS preservation metadata model

For information on the PREMIS working group, results and publications see

http://www.loc.gov/standards/premis/

3 Design patterns Command, Composite and Visitor

Wikipedia on design pattern Command http://en.wikipedia.org/wiki/Command_pattern

Wikipedia on design pattern Composite http://en.wikipedia.org/wiki/Composite_pattern

Wikipedia on design pattern Visitor http://en.wikipedia.org/wiki/Visitor_pattern

4 LIFE

“LIFE: Life Cycle Information for E-Literature”, home page containing information and final report form the project.

http://www.ucl.ac.uk/ls/lifeproject/

5 Utility analysis

Rauch Carl and Rauber Andreas: Preserving digital media: Towards a preservation solution evaluation metric. In: Proceedings of ICADL 2004 in Shanghai, December 2004. pp. 203-212

Available online at http://www.ifs.tuwien.ac.at/~rauch/Rau04_Utility_Analysis.pdf

6 Genre classification paper

Kim, Dr Yunhyong and Ross, Prof Seamus (2006) Genre Classification in Automated Ingest and Appraisal Metadata. In Gonzalo, Julio, Eds. *Proceedings EUROPEAN CONFERENCE ON RESEARCH AND ADVANCED TECHNOLOGY FOR DIGITAL LIBRARIES (ECDL)* LNCS 4172, pages pp. 63-74, Alicante, Spain.

Available online at http://eprints.erpanet.org/110/01/genre_extraction_KIM_ROSS_2006_ECDL.pdf