A workflow for digitalization projects

Tom De Mulder — tdm27@cam.ac.uk

May, 2005

Abstract

More and more institutions want to convert their traditional content to digital formats. In such projects the digitalization and metadata stages often happen asynchronously. This paper identifies the importance of frequent cross-verification of both. We suggest a workflow to formalise this process, and a possible technical implementation to automate this workflow.

1 Introduction

In the entertainment industry, the importance of synchronizing the audio and video tracks of a movie is well understood. It is vital that both audio and video (and, if present, subtitles) run smoothly alongside one another. If this weren't the case the result would be a discordant unintelligible mess.

In a similar vein, we want to keep the metadata and data streams of our imaging acquisition to remain synchronized so that in our end product, a collection of data and metadata, makes sense.

In the past, this lack of synchronisation has caused some problems for digitalization projects at the Cambridge University Library. In these cases, the imaging process was completely separate from the project expert creating metadata for the items being photographed. Only at the very end would both streams of information be brought together, at which point inconsistencies would often be found.

It proved time-consuming and complicated to resolve these: human intervention was needed to go through the entire collection to spot and correct errors and omissions. More time was taken up by having to schedule more work for the photographic unit, and to recombine the final results.

In this paper, we try to address these problems and formulate a process to catch errors before they impact other aspects of the imaging workflow.

While this paper focuses on photographic imaging of manuscripts, its scope could easily be seen to include any project where metadata and data acquisition happen as separate tasks, such as images of objects, 3-dimensional object scans, digitising analogue audio or video, ...

2 Synchronizing: embedding common keys

Broadly speaking, metadata serves two purposes: identifying and describing data. It will be used to navigate to or locate the data (in our case, the manuscript images) when browsing or searching a repository as well as to gather more information on the data itself once it has been found.

In the context of digitising workflows, identification is the use we care most about—only after the data has been unequivocally identified can more useful work be performed on it, such as adding more descriptive metadata.

So, a way needs to be found to uniquely identify the object being photographed. For example we might use its library classmark. The most straightforward way of encoding this classmark in the aquired image with the current state of technology is to make it part of the filename.

A more "analogue" approach towards image identification would be to make sure the image identifier (the "classmark") is always displayed inside the image itself. This could be printed on a small piece of paper, or written on a small whiteboard and included in the field of view of the camera when the photograph is taken.

In addition, a variety of ways exist to embed metadata directly into data files (see also Appendix A). This effectively glues the data and metadata together, reducing the chance of them getting separated. The sooner in the workflow this happens, the higher the chance of the workflows remaining synchronized. It also makes it substantially easier to resolve discrepancies at a later date.

3 Workflow

3.1 Definitions

For the purpose of this workflow, we define the "Expert" to be the person controlling the metadata for the material to be imaged.

The "Photographer" is the person (or group of people) in charge of producing the image files of the material.

The "classmark" is a document's unique identifier.

4 A technical implementation

4.1 Infrastructure

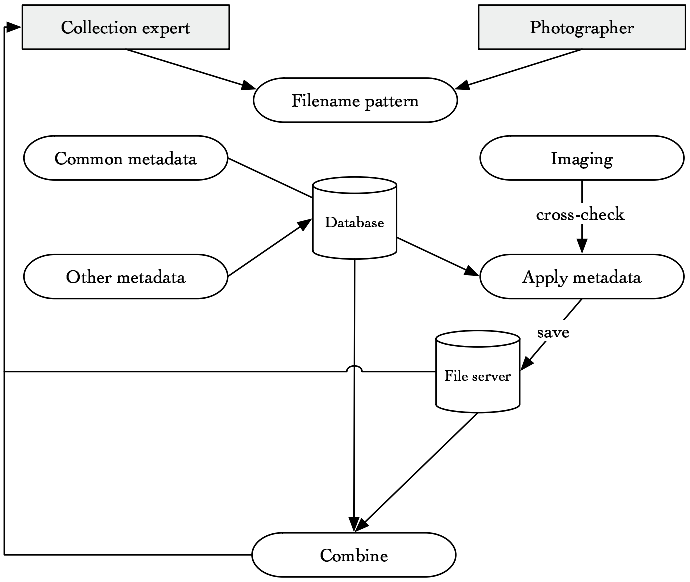
For this example, we assume that some level of technical influence is possible over all steps of the imaging and metadata workflows. Without it, workflow synchronisation becomes very difficult.

4.1.1 Central services

The core of the system is a set of central networked services.

One of these is a relational database service, which will handle the project metadata. Its schema (in reality probably a set of views[1]) is tailored to the

[1] A database view is an abstraction layer on top of the actual database schema, making it possible to represent information in a way that makes sense to the user while hiding the techni-



needs of each individual project. The metadata fields can be exposed through a web interface or via ODBC[2] clients.

Also centrally provided is a networked filesystem, accessible both by the central server and the photographer. This is where the photographs taken will be stored, and where they undergo a series of automated workflow operations.

The final central service can most generally be described as providing "remote procedures". It will be used by various other components of the system to retrieve or store information associated with various steps of the workflow.

4.1.2 Clients

We assume that the photographer is using an Apple Mac with some version of Mac OS X installed. This allows the use of "Folder actions": Applescript-controlled actions (typically small programs) that are executed whenever (for example) a file in a filder is saved/opened/modified.

4.2 First steps

Before the metadata or digitising workflows start, some standards need to be agreed on. These will serve as the key elements to synchronize workflows:

- The exact lexical format of the identifier. For example: nn.xxx-yyy:bbb,[r/v] where the possible ranges for nn, xxx, yyy and bbb are defined and r/v are agreed to mean recto/verso.
- The metadata scheme to be used. In most cases use of the Dublin Core[3] will be appropriate, possibly with custom extensions.

cal complexities of the implementation. See also http://philip.greenspun.com/sql/views.html

[2] A protocol to access databases remotely, often through visual clients such as OpenOffice BASE or Microsoft Access.

[3] http://dublincore.org/

- Common metadata: a set of metadata tags that will apply to the whole collection, such as “collection name”, …

4.3 Workflows

4.3.1 Metadata

The expert enters metadata in the central database. No particular order of data entry is assumed, this could even happen in batch if the client supports it, then transferred to the server. The record identifier field, agreed at the very start of the project, should be checked rigorously.

Whenever the server deems a metadata record to be completely filled in, it can check the networked file system for the corresponding files. If they are present, metadata can be added.

It is important that, should a metadata record that was previously flagged as “complete” be changed, the embedded metadata in the corresponding image is changed immediately.

4.3.2 Imaging

If the photographer uses Adobe Photoshop CS, then a metadata template can be defined holding the collection’s common metadata. This template should then be applied before the image is saved, making sure the image contains its metadata as soon as possible. This reduces the chance of the image becoming “orphaned”.

When the image is saved on the networked filesystem, a folder action can check the filename to make sure it agrees to the standard defined in the very first step of the project (in the example, nn.xxx-yyy:bbb,[r/v]). Any mistake here should be caught immediately and the workflow halted until it is resolved.

4.3.3 Server-side automation

It would be bad practice to rely on folder actions alone to synchronize the two workflows without an extra level of checks.

The server should make periodic checks over the database and the networked filesystem, checking filenames, testing images for embedded data and validating or adding metadata as appropriate.

4.3.4 Notification

Both the expert and the photographer can be automatically notified (if they so desire) of the progress of the other party. The photographer could be sent, at the end of each day, an email with an overview of what metadata records are complete. The expert could receive a list of images, outlining where corresponding metadata records exist or are missing.

At any point in time, a simple web interface can show the entire project status, highlighting if any discrepancies are found between the two workflows.

4.4 Final stage: the fully enriched image

Once both the metadata and imaging workflows are complete, a final merger of both can happen to produce the fully enriched data with embedded metadata and, preferably, a separate direct metadata dump from the database in XML format (which for most applications is easier to use than embedded metadata).

It may seem that this final step renders many of the previous steps redundant, but those steps are crucial to retain project cohesion should, for some unforeseen reason, the project be aborted or put on hold for a long time. In that case, no final "output dump" would have been produced, but at least the data produced would still be identifiable.

5 Conclusion

Embedded metadata seems the solution to many digitalization woes. However, because much software (and many digital formats) wasn't designed with embedded metadata in mind there are many potential pitfals that one should be careful to avoid.

One risk is that, due to the difficulty of parsing and indexing embedded metadata, another method might be used to store the actual metadata (for example, a relational database). Unless careful checks are made, the risk is substantial that the two metadata sets would start to diverge.

Any utility or repository that reads or manipulates embedded metadata should be aware of these caveats. It is important to define an *authoritative* source for metadata, and periodically verify any other set of stored metadata against it.

In most cases, embedding metadata will be the final alteration to the data before it is stored in a repository, and the data nor the metadata will ever change again. In this case it becomes a very valuable component of digital preservation, as it makes sure that even in the future data and metadata will never get separated.

However, as shown in this paper, embedded metadata can also serve as a useful tool for workflow management, and increase the reliability and value of digital materials.

A Embedded metadata: a short technical overview

A.1 History

Most graphics formats have some history of allowing metadata to be embedded. For example, the TIFF and JPEG standards allow sets of EXIF and IPTC metadata fields. These fields are narrow in scope, however, and tend to be aimed at describing the technical aspects of the image capture process rather than the more descriptive metadata we are interested in.

With the advent of semantic web technologies (most notably RDF[4]), however, it became possible to describe content without the constraint of a fixed set of fields. Rather, the schema could be flexible and tailor-made for the content, while still being machine-readable.

Using this technology, Adobe developed XMP[5]. It allows RDF/XML[6] metadata to be embedded into a variety of file formats. By now (2005), all current versions of Adobe's own products support this standard, as do many third-party tools.

One of the default schemas supported by the XMP standard is Simple Dublin Core. For our purposes this may seem disappointing, since Qualified Dublin Core is much better suited for exhaustive metadata (a complete discussion of Simple vs. Qualified Dublin Core is outside the scope of this paper). However, several ways exist of encoding Qualified Dublin Core in RDF/XML so that parsers expecting Simple Dublin Core can still read the SDC fields.

Of course, if this approach is used, care should be taken when editing files — if a file containing QDC XMP is edited and saved using a tool that expects SDC, the extra metadata fields will in all likelihood be lost.

[4] The Resource Description Framework. As its name implies, it is a framework for describing and interchanging metadata

[5] http://www.adobe.com/products/xmp/main.html

[6] A common way of encoding RDF in XML format.