Some considerations on using P_META and Dublin Core

EBU Project Group P/Meta

Since the EBU P/Meta Project began in 1999, a number of standardized metadata schemes have become available, and there is considerable debate regarding their relative merits and appropriateness for different purposes. As well as P/Meta's output – EBU Tech doc 3295 (P_META v1.0) *(yet to be published)* – the EBU also offers Tech doc 3293 (Metadata for Radio Archives) [1], based on Dublin Core. However, the different schemes need not be mutually exclusive.

This article – written by the former Chair of P/FRA, Richard Wright from the BBC Information & Archives division – discusses the relationship between these two schemes and the scope for co-existence.

"Standard metadata"

In the archive world, **standard metadata** is shorthand for standard or core bibliographical data, or core records. It is the term for the set of information in the catalogue. Sometimes this concept is divided into **core** and **full** metadata, and both have their place. In the standardization world, however, core or minimum metadata is by far the easiest to standardize. In consequence, there is very general acceptance and implementation of **Dublin Core** metadata.

The important question when deciding about metadata standardization is: *"what do you want to do?"* Will your metadata stand alone, or will it be combined with similar metadata from other companies? If you are selling footage and want to sell it in some general electronic marketplace, then you will need to adopt the standard of that marketplace. Unfortunately we have, at present, more planned than actual e-markets, but the principle is clear.

If your metadata will stand alone, then it is not imperative to use anybody else's standard, although this approach (which has been commonplace in specialist archives, including broadcasting) is not only short-sighted but is very likely to become a real problem in a world where networking and interchange are dominant activities.

It is important to realize that simple, well-defined and well-managed metadata will usually be able to be *mapped* from a local system into a general system. At the time when your stand-alone archives enter some cooperative or commercial activity, data transfer or data interpretation will be required. As part of this exercise, your non-standard metadata can be mapped into a standard form. This can be done relatively easily, even for very large catalogues. The catalogues of the BBC, INA (the French audio-visual institute) and ORF (the Austrian national broadcaster) were exported, mapped to Dublin Core elements, and formed into a **union catalogue** as a relatively small part of the PRESTO project (D7.3: *Common access to broadcast archives*).

Expression and exchange of metadata

What does metadata look like? It (sometimes) comes up in the form of words on a computer screen. But what else do we need to know about it?

The problem is not how people see metadata, but how computers see it. In order to exchange data between companies, or to form union catalogues or to sell to a general e-market ... the data has to move from computer to computer.

Computers are very rigid and only communicate when there are no uncertainties. Either two computers will hold metadata elements in exactly the same way (unlikely), or there will be a special transfer operation involving labelling the metadata at the export from one computer, to guide the interpretation at the receiving end.

Such labelled data is easily handled with a mark-up language such as SGML, HTML and now XML. A mark-up language adds information to identify the data being passed. Thus if one computer holds **Title** as a database field of a certain length and in a certain character set, and the other computer has a database field called **Designator** which is the equivalent, but differs in name, length and character set – a transfer can still be made using XML. A computer programme needs to be written to look for items marked **Title**, convert them to the storage requirements of the receiving computer, and store them in the field **Designator**.

In XML there are further structures, such as schemas, to allow the computers to know that the data coming in is correct and complete, and guide the conversion into the receiving database.

The important issue is that metadata will (and should) be stored in whatever way a particular database prefers. This in no way invalidates the ability of that metadata to be in agreement with a standard, because standards are about the interpretation of the data, not the storage. When metadata is brought out of the database, it has to be in some physical format (ie expressed), and XML provides a very attractive method to express the metadata.

Storage of metadata

In the previous section, it was assumed that metadata was in a database. However, we have also discussed metadata that accompanies data in an electronic signal or file. This file can be permanently (we hope) held on physical media such as data tape, CD or DVD. So why not put metadata with the media, rather than in a database?

The short answer is that metadata held with the media is of **no use at all** for helping to find material in a collection. The whole point is to have a catalogue – index, inventory, finding aid or whatever you call it – available in some way that is separate from the collection, in order to find material in the collection. Therefore the metadata must have a separate home.

It may reside with the media as well, as a sort of distributed security copy, but that raises the problem of data update. Databases are easy to update: that's what they do. Metadata written to a CD is impossible to update.

The general conclusion is: only one item of metadata should **always** be with the media if held in a file format: the **universal identifier**. That is the key to the database. Full metadata should always and only be held in the database. Other core metadata, providing it will not change, can also be held in the file, and this is the approach used in the Broadcast Wave Format [2].

It is to be hoped that the file formats under development for video (e.g. MXF [3]) and film will adhere to this principle.

P_META and Dublin Core

Dublin Core [**http://dublincore.org/**] is clearly NOT a full set of metadata, capable of fulfilling all purposes of a large and complex business such as broadcasting. Dublin Core is not even capable of satisfying all the purposes of the corner of broadcasting where Dublin Core is most accepted and acceptable: the archive. In an archive, first of all there are inevitably legacy systems which predate Dublin Core. Secondly, there are various types of broadcasting metadata that are not within the scope of Dublin Core

- technical metadata (e.g. timecode);
- "preservation transfer" metadata (e.g. "number of dropouts in this recording");
- documentary metadata (e.g. transcript, running order);
- transactional metadata (e.g. loan period; charges for overdue material).

The archive is just one small corner of broadcasting. Personnel information about salary, grade and seniority; schedule information; rights payments and, finally, the literally hundreds if not thousands of "gory detail" elements (the minutia of various computer databases) – is simply not a part of Dublin Core.

The "gory detail" isn't a part of P_META either. P_META may look complicated, but just in the BBC archive alone there are about 10 major computer systems, between 50 and 100 minor databases and a variety of other information systems ranging from spreadsheets to simple lists. Across the BBC you can multiply this complexity at least 20-fold. Across the EBU, the complexity must add up to something like millions of separate data elements. Nobody is trying to standardize all of that!

P_META does intend to capture – and standardize the naming and semantic definition of – the major information exchanged at the major *interfaces*: the places where information is transferred from one organization's database to another organization's database. The goal of P_META has always been to standardize data to be exchanged between interfaces, because without standardization it falls back on people to read the data coming out of one system, and reorganize and re-type it – to get it into the next system. The fundamental principle of P_META has always been: **type it once.** **"**Type it once" requires standardization across the interfaces.

P_META versus Dublin Core

Why versus? Because P_META is bigger than Dublin Core? There is nothing in either P_META or Dublin Core that is compulsory. Dublin Core has 15 fields, but none are compulsory. Metadata with just titles and names can be compliant with Dublin Core. The important issue is to use a standard name and definition for exactly the metadata that is significant for achieving some purpose – typically for sending information from A to B.

Exactly the same is true of P_META: it is an inventory that is capable of providing *named*, *defined* and, in many cases, *controlled elements* – with a fixed vocabulary or other authority mechanism – for doing what you are likely to need to do.

P_META and Dublin Core, again

So, how to proceed? At this point guidance becomes complex because it depends upon your starting position, and your goal.

Starting from scratch

Not many people are really starting from scratch, with no legacy systems of useful databases. However if you are, then the first issue regarding metadata is the purpose to be achieved. When that purpose involves carrying information from one system to another (internally or externally), then you need either a metadata standard or the systems won't communicate.

You should use as few metadata elements as possible. This is where system design and metadata design are important. You could create ten (or more) fields for ten (or more) different types of people associated with

Abbreviations

HTML	HyperText Markup Language	**SGML**	Standard Generalized Markup Language
MXF	Material eXchange Format	**XML**	Extensible Markup Language

broadcast programmes – or you could have one name field and one role field, following the P_META principles.

It is important to realize that most databases don't correspond to any standardized metadata set, and most database systems (from Access to Oracle) are much more general (inherently) than any single metadata set, however large. It is when data goes in and out that the standardization comes into play and, if you're starting from scratch, then it isn't actually the database that needs to be P_META compliant – it's the route in and the route out.

If you're starting now, then these routes should be via XML interpreters/generators. The XML probably has to be written or trimmed to your purposes anyway, and during that operation, the compliance with P_META can be implemented – **just for those elements that are needed for your purposes**.

Legacy systems

For legacy systems, again the issue is import/export: the modern approach is XML. It won't cost any more to implement XML import/export which is P_META compliant, than it will be to implement something that has no compliance to anything – so there is every reason to use the P_META semantics and naming. That way, for the same price, you have XML routines that not only solve your local system-to-system communication, but also form a solid basis for moving easily and cheaply towards business-to-business communication.

Best of both

If your business-to-business requirements also involve the world outside of broadcasting, there is an advantage in being both Dublin Core and P_META compliant. This is not difficult. The only thing that is difficult is to make **every** data element agree both with P_META and Dublin Core. This is in general impossible, and in any realistic case there is no reason to pursue this unattainable goal. You simply identify the metadata that needs to be exchanged in a Dublin Core context, and ensure that import/export for those elements is in accord with the Dublin Core standard. You "name" those elements using Dublin Core. **And at the same time**, if you want to have the "best of both", you also name those elements using P_META. This is a trivial extra amount of XML.

To make it totally easy, the P_META committee has prepared a document to identify the most likely associations or dual-naming possibilities. During the working out of the XML routines for getting data into and out of your system(s), where needed you pick a Dublin Core name and a P_META name, making your XML routine bilingual.

Why bilingual? Why isn't there a strict subset of P_META that exactly agrees with Dublin Core? The answer is that there are differences of approach that make this route very difficult, perhaps impossible. It's much easier, certainly far more flexible and less restrictive on the implementers, to advocate the dual-naming during import/export, rather than trying to equate two systems which have some basic differences of approach.

For instance, Dublin Core has Creator, Publisher and Contributor. These are very general roles, and can be fulfilled by people or organizations. P_META has name attributes and role attributes, and a name+role pair would serve the purpose [1] of Creator or Publisher or Contributor. So rather than get bogged down in mapping Dublin Core to P_META and vice versa, the essential task is to map *your* data elements to BOTH – and it is almost no harder to do both than to do just one.

Disclaimer

This article gives my personal opinion. It is not the official view of EBU Project Group P/FRA (Future Radio Archives), because that group finished its business a year ago and closed. However it is my further view that all the members of P/FRA were (ardent) supporters of the importance of standardization of metadata, and

1. And a lot more purposes, as the list of roles is very extensive.

would therefore support this attempt to remove any unnecessary obstacles toward standardization caused by the present situation of multiple metadata standards and approaches.

Richard Wright, BBC Information & Archives, 31 October 2002

Bibliography

[1] EBU Tech doc 3293: **EBU core metadata set for Radio archives**

[2] Richard Chalmers: **The Broadcast Wave Format – an introduction**
EBU Technical Review No. 274, Winter 1997.

[3] Bruce Devlin: **MXF – the Material eXchange Format**
EBU Technical Review No. 291, July 2002.