AssetStoreUseCases

To aid coming up with a data model for AIPs in DSpace, here's a brainstorm of use cases. These use cases are focussed on the sort of content that needs to be stored, and what sort of supporting information (representation information) needs to be stored with it in order to keep it usable in the future (by migration or emulation, or a mix).

Please add to this list! Also feel free to annotate any use cases with more detail, or give an indication of how important it is to you. Add any use case you can think of, doesn't matter if it's obscure, or if it overlaps or is sort of covered by another use case. Brainstorming a complete list is the aim here.

In no particular order:

Contents

Simple image

e.g. a GIF or JPEG. May have embedded metadata. Will often want to generate thumbnails.

(ScottYeadon) Will also want to generate a variety of derivative images (e.g. larger branded/watermarked versions, a variety of thumbnail size/formats if linked to external discovery/search/access services, etc).

.zip file

Could contain anything. Example where some sort of hierarchy/nesting in the model may be necessary.

(ScottYeadon) This is true for other aggregates such as .tar which are also further complicated when additionally compressed (e.g. .tar.gz)

PDF document

(ScottYeadon) Unfortunately there are many variations of PDF some of which can be interpreted as being corrupt if not understood by the PDF reader application. Should we need to be able to "type" PDFs since some will be stored as image PDFs without accompanying text.

(JörnNettingsmeier) If it's possible, DSpace should harvest information on the type of pdf from the file, i.e. whether it's searchable or not, and whether it has stupid misfeatures such as "No cut and paste" enabled.

Microsoft Word document

Powerpoint slides

Powerpoint slides with embedded video

To reliably preserve, you need to know about the embedded video

(JörnNettingsmeier) In my not so humble opinion, such proprietary formats are the antithesis of document preservation. Instead of wasting any more time with non-documented, non-standardized formats from non-cooperative companies, let's spend some efforts on user education instead. (I know we have to deal with them anyway... See my note on such formats at the bottom of the page)

Slides as a sequence of GIF files

Need to know the sequence as well as the format of the files

Video clip (.avi/.mov)

Need to know audio and video encoding algorithms. AVI is really just a wrapper. As is the MOV container which can also contain text, interactive elements, and QuickTimeVR elements.

Audio clip (.mp3)

May contain embedded metadata (artist, track name etc), which is part of the MP3 standard. However the embedded metadata may or may not conform to some standard (e.g. genre taxonomy/categorisation)

Audio clip (Ogg vorbis)

(JörnNettingsmeier) Ogg Vorbis provides equivalent or better than mp3 quality and is not as patent-ridden. The ogg container can include metadata. Other interesting formats are Ogg Speex (tailored towards very low bandwith speech encoding) and Theora (a video codec). See www.xiph.org for details.

Musical composition (might be .wav)

(PeterRaftos) Composition consists of 1 or more movements, each movement represented by an individual file. Each movement has some metadata attached to it (movement name, duration), but most metadata resides at the composition level (composer name, composition name). Each movement is part of an umodifiable and arbitrary sequence within the composition. They might be numbered, but might also be names. On retrieval, the composition must always be presented in such a way that the sequence of movements is correct and complete.

For example, think of a Beethoven symphony with four movements: the movements can't be retrieved alone or out of order.

(JörnNettingsmeier) Aside: there exists an excellent free c helper library called libsndfile (http://www.mega-nerd.com/libsndfile/). it comes with an example program sndfile-info that extracts information from a great number of different sound formats. You might want to look at it, it's GPL. As another aside, if the dspace developers have any questions concerning the handling of audio data, I'd like to recommend the Linux Audio Developer's Mailing list (http://linuxaudiodev.org), there is a lot of expertise there, which should also be applicable to cross-platform projects. If you don't want to subscribe, direct questions to me, i'll pass them on.

DRM'd Audio clip (some AAC? - designated as m4p "Apple's DRM in iTunes", some .wpa)

Encrypted file. Will we have to deal with this sort of content? Files with DRM characteristics are likely to become more prevalent over time, but will probably be difficult to deal with as encryption and the characteristics of most DRM technologies are at odds with ease of preservation. The question is should DRM'd anything, audio, video, PDFs, etc. be in an archive, unless it includes the encryption key, or the DRM times out.

(JörnNettingsmeier) No, it should not.

Web page

Potentially lots of related files of different formats (HTML, HTML with embedded Javascript, images, shockwave flash etc). Perhaps environment information (which browser/OS/software dependencies) is more realistic than capturing format of every single component of every single file (though less ideal)

(JörnNettingsmeier) side thought: when archiving web page structures, all server-side processing is already done. i.e. dynamic content is not represented. so we need a very prominent time-stamp of when the wget'ing or whatever was done.
Ideally, dspace could do the mirroring itself. The user just enters a URL and preferred link depth (and maybe other wget-like options), and dspace pulls the page. This way, more metadata can be harvested by dspace as needed (whois information of site owner etc.)

Source code package (e.g. DSpace!)

Need to know build environment(s), dependencies (DSpace relies on a lot of other packages). How do building/installation instructions fit in? Will we know format of every file? May be one-of-a-kind data files (e.g. dspace-source/registries/dublin-core-types.xml).

Executable file

Need to know execution environment, dependencies and so forth. Will probably have several possible permuations of environment. So we could describe one known working environment, specify general guidelines (e.g. x86 processor running Windows), or enumerate lots of environments.
(JörnNettingsmeier) I do see the need for this, but my gut feeling is it's a fight against windmills anyway, and better people realize their problems now than later.
Executable code dies with its platform, and since you can't preserve platforms in dspace, you can by definition not preserve executables, only delay their death a little while.

Installer (e.g. MyApplication-setup.exe)

Both an executable and a 'package' format that contains many pieces.

Configuration file

From relatively simple configuration files like application settings (dspace.cfg) to complex server setups

E-mail

Can contain MIME-encoded attachments. Will DSpace have to deal with e-mails? Is one e-mail per DSpace Item too much?

(ScottYeadon) Should the submitter/collection owner determine whether one e-mail per item is appropriate? Not sure the model should really deal with these issues?

Data set

Need to know semantics of data set as well as the format of the file ('XML' or 'SQL99 dump' isn't enough)

Medical image

Other data about the image may be essential – equipment specification and calibration parameters, patient details

XML documents

(ScottYeadon) Need DTD and DTD version info, associated stylesheets and version info. Should an attempt to preserve XML processor or processing environment be made? Probably generally not but there may be cases where this is determined to be necessary
(PeterRaftos) Or a W3C XML Schema; or a RELAX NG schema; or a Schematron file. Or combinations? This helps with machine processing, but not human comprehension: for well-known XML applications (Docbook, XHTML) there are external descriptions of the semantic (as well as syntactic "payload" of element sets in a given application; what about custom XML apps? For instance, I understand that Lonely Planet are XML-ising their materials: they've written their own XML application to suit their documentary needs. If LP stores their original XML documents in a repository, where should they store a human-friendly description/exegesis of the app and in what format?

Notes on Aggregate Items

(ScottYeadon) Some of the above examples are types of aggregate items, that is, they are comprised of multiple bitstreams of varying formats and these bitstreams are so tightly related such that item coherence is reduced when one or more of these bitstreams is not available.

In many instances long-term preservation will be best supported if the aggregate were decomposed into its raw components with each of the raw components archived in an open format. This also means the relationships between each of raw components must also be stored in some way (e.g. RDF, METS, some other XML map) and then interpreted to support access. The advantage gained here is not only is the aggregate more likely to survive long-term, but that its components may be used/accessed/discovered in the future in ways not currently considered.

This is not going to always be possible and so proprietary aggregate items are likely to be stored. At the very least we then need to store software, versions, OS metadata and the collection owners need to determine an ongoing migration strategy to keep this material alive.

Notes on data formats that only work with proprietary software

(JörnNettingsmeier) There should be a flag of some kind that indicates whether the item is in a format that is a) documented and standardized, but only proprietary viewer implementations currently exist, or b) is undocumented and only proprietary implementations will ever exist. A little tombstone icon to indicate the imminent death of the data comes to mind, or a cuneiform icon that tells the user that advanced deciphering work may be required in the future.

Notes on metadata-only stubs

JörnNettingsmeier) Now this is off-topic for asset store use cases, but it seems some people (me included) use dspace as a research bibliography database, and not all media are available in electronic form. in that case, the item description might contain a pointer to a paper copy, perhaps using a library signature or room number. Currently those items don't even touch the asset store, only the metadata db, but i wanted to point this out in case it becomes relevant by future design revisions. Of course, such items should also be clearly marked as second-class citizens, since they cannot be truly preseved within the system.

Item relationships

(JörnNettingsmeier) I'm not sure how this applies to the asset store, but have you considered that items might be related in a number of ways? For instance, an item might be a metadata stub for a monography, while the articles contained within are separate and somehow link to the monography item (same with periodicals). Or an item might be a keyword from a controlled vocabulary, together with a usage explanation, and other media could include a pointer to it.
I would also like to see such relationships reflected in the input UI, i.e. present the submitter with a drop-down list of periodicals when s/he enters an article from a magazine, or present a choice of keywords.
Update: some people suggested modeling item relations with sub-communities (see this thread from the mailing list archive). my gut feeling is mixed about this, but it might work for most people and does not require any changes to dspace's data structure...