Promoting discovery and use of repository content: An architectural perspective

John Mark Ockerbloom, University of Pennsylvania

Available at: https://works.bepress.com/john_mark_ockerbloom/7/
Thank you. We’ve been hearing a lot at this meeting about design of repository systems for trust. Today I want to talk briefly about designing repository systems for discovery, and show some of the things we’ve been building, at the Penn libraries to make our content more discoverable, and show the architectural principles behind what we’ve done.
If your content isn’t found, is it preserved?

- Archives are for use
- Rot spreads out of sight

(Photo by San Jose Library, 2007. CC license: BY-SA)

Now, you may be thinking: “I’m in charge of a *preservation* archive. Do I need to worry much about discovery?” Well, I’d respond that you can’t really say you’re effectively preserving content unless the people in the community you serve can find the content when they need to. Ultimately, archives are for use.

Also, we’ve found from experience that it’s a lot easier to detect archiving problems early when you have content that gets regular use. If you have an archive that’s dark either by design or by neglect, then problems with formats, consistency, metadata or context can fester out of sight. Automated checks can catch many problems, as we saw in some of yesterday’s talks, but the ultimate test of effective preservation is use. If the archive is regularly used, your users will notice when they have trouble working with your content, or if they see something unexpected, and they can warn you when it’s still relatively inexpensive to fix the problem.
Architectures for discovery
(or, my talk in one slide)

• **Information architectures**
  – Identify relevant “anchors” for search and browsing
  – Discover and exploit rich networks of relationships
  – Example design: Subject maps for catalogs

• **System architectures**
  – Open content and services to outside applications
  – Standardized interfaces make this practical
  – Example design: ILS Discovery Interfaces, VCat

• **Social architectures**
  – Engage the community that populates and uses content
  – Monitor usage, annotation to integrate into search and browse
  – Example design: PennTags

There are lots of tools and techniques that can aid in discovery. I'll talk about a few specific technologies, but I also want you to see three important architectural aspects that these technologies exploit.

I'll talk first about information architecture, and show how one can take advantage of the rich semantic structure that’s expressed or implied in repository metadata and content to guide users to the content they need.

Then I'll discuss system architecture, to show how standardized APIs allow a wide variety of discovery applications to work with a broad range of repositories and content.

And finally, I'll talk briefly about how the people that create and use the content of your repository can build up a kind of social architecture that you can use to enhance the discovery and use of your content.
Now, each of these kinds of architectures could easily be a talk of its own, but for the sake of time I want to focus on some techniques that I find particularly interesting. When we look at information architectures, for example, we see there’s been a lot of innovation and new tools for searching, but many systems have relatively primitive facilities for browsing.

And yet, it’s often very important when you’re accessing an archive to make sure you have a comprehensive picture of the parts of the archive that are relevant to your research. You need not just a few good items that might bubble up to the top of search results, but anything else that might be related to them for your purposes. Many legal requests for example, require all information you have related to a particular topic, not just a few representative hits.

For investigations like these, you really need to be able to browse. But the mechanisms we often provide for browsing can be very limited…. 
There’s the old-fashioned alphabetical browse that we used to do through cards in a catalog drawer…
In a lot of our online library catalogs, alphabetical browsing is still the only option given for browsing subjects. But it tends to give us worm’s eye views of a collection. If I’m trying to see what resources my library has on English dialects, for instance, I have to go through a lot of screens like this one, which lists 25 dialect regions in England starting with L, M, or N. To actually see what books we might have on these subjects, or similar subjects, I have to do a lot of clicking around.

In many ways, this display gives me less than the old card drawers did, where I could at least riffle through the cards quickly to see what’s there. It’s no wonder that our library logs tell us that most of our users don’t browse by subject very much.
In another common browsing architecture, the shelves in library stacks and the boxes of archival collections are often organized hierarchically. This is often combined with alphabetic or other linear structure…
Hierarchical discovery

Here, for instance, is a bit of an online EAD description showing how we’ve organized a collection of historical writings into a hierarchy of boxes and folders. This organization can work fairly well for browsing if your research problem naturally follows the collection’s hierarchy. But it often doesn’t. For instance, if I’m browsing for books on computer network security in my library, I need to know that some are filed under computer security, and some under communication networks, and those are in very different places in our library’s shelving hierarchy.
When we move to virtual space, we can take browsing in new directions unconstrained by physical limitations. Facet-based browsing, for instance, lets you slice and dice a collection along multiple dimensions. It’s commonly seen in E-commerce sites; for instance, an online fashion catalog that lets you focus on particular colors, styles, and price ranges for products you’re interested in. And it can be a very powerful way of zeroing in on a category…
Faceted discovery

...provided, again, that it neatly fits into the facets designed in the system. In this experimental faceted display of library resources related to Macbeth, many of the facets--like language and author name--are essentially discrete and independent of each other, so browsing them is pretty straightforward. But when it comes to subjects, things get trickier. Subjects are not independent of each other. Some are variants or specializations of other subjects. Some are related to other subjects but aren’t neatly nested within them. It can be easy to get caught in misleading cul-de-sacs when browsing subjects by facet. For instance, I’ve browsed some faceted catalogs for books about the civil war in a particular state, and have found them showing surprisingly few books. The catalogs only show books with metadata that specifically says “civil war” and the full name of that state. They won’t show other books about particular battles in that state, or a particular city in that state, or to secession and slavery issues in the state. All these subjects are related, but the relationship isn’t expressed in the catalog’s simple faceted organization.
We have another kind of browsing structure that’s very good at representing complex relationships, though: the map. This map of Sydney, for instance, displays a wide variety of districts and blocks, as well as individual buildings of interest inside the districts, and it shows how they all connect and nest together. It lets a reader see at a glance what there is in Sydney and how they can get to it, and is comprehensible at a glance while also supporting close scrutiny.
We can use the same general idea to display related books in a library. We don’t have to literally draw them as we would draw a geographic map, but if someone’s interested in a particular subject, we can display both the books on that subject and also show connections with related subjects. This two-column display here is visually dense, and is basically just text, but it shows you at a glance what books our library has on cooperative cataloging and related topics like intellectual cooperation. (If we scrolled down a little, we’d see not only the names of these related topics, but also many of the books that fall under those topics. And if we want to refocus on one of those related topics, all we need to do is click on it.)

I’ve been calling these displays “subject maps”, and they can represent quite complex ontologies and large collections. The catalog being mapped here contains over 3 million items. But it’s comprehensible to the user because it only focuses on a small portion of the collection at a time, and lets the user see and go to related parts of the map. And because users see both concepts and instances of those concepts at the same time, they learn their way around the ontology as they explore it.
Building effective subject maps

- Start with the best metadata, ontologies you can
  - Subject assignments in existing metadata records
  - Relationships from LCSH standards, crosswalks...
- Automate analyses for more connections, relevance
  - Subdivision, geographic, lexical, correlation analysis
  - Analysis can also auto-correct, localize subject headings
- Specialize and refine where it gives the greatest benefit
  - Logs can tell you what people are looking for, finding
  - You know what your special collections and communities are
- Customize where you have to
  - (but try to automate it, or share your customizations, wherever possible)
- Design your displays, navigation to give users bird's-eye view, and teach them the terrain

Our subject map is based on Library of Congress subject headings, which define an extensive controlled vocabulary of subject terms and relationships between them. But we adapt and enrich that network of relationships to fit the collection we're showing. Library of Congress Subject terms not used in our collection get dropped from our map, so the user won’t navigate into dead ends. And if our collection introduces new terms, we incorporate them into the map. We use a variety of automated techniques to do this, including lexical analysis, geographic reasoning, and correlations.

For example, we can automatically determine that a subject heading “Independence Hall (Philadelphia, Pa.)” is related to Philadelphia, which is related to Pennsylvania. Or, if we see books with two subjects, the first being Thomas Jefferson and the second being “Presidents -- Biography”, we can often safely infer that “Thomas Jefferson” is somehow related to Presidents.

We can also customize our subject maps manually, and knowing the nature of our collections and watching how users interact with them can give us guidance on how best to customize. But wherever we can, we try to automate the construction of our maps, to get the most benefit for our efforts.

[Add note about popularity of subject maps in Online Books Page]
Subject maps aren’t the only way to build networks of relationships. Depending on the kinds of collections you have, you may find it useful to analyze citations, like Web of Science does, so that people can find documents that cite or are cited by what they find initially. Google Book Search has done some interesting things with quotation analysis, relating documents that have text in common. There are also a variety of techniques for automatically organizing documents into clusters based on their text or metadata.

But you might be saying at this point, “That’s all well and good, but it’s hard enough just to create a standard trustworthy repository. Can we realistically require repositories to also implement a slew of high-powered discovery methods?” ….
System architectures

Well, if you architect the system right, the repositories don’t have to implement them themselves…
Make the repository part of an open, discovery-rich system

- Let the repository do what it does best
  - Ingesting, managing, providing access to content
- Let discovery apps do what they do best
  - Using functionally rich repository APIs
  - Using standard formats and interfaces wherever possible
    » (If you have to invent, be as simple as possible)
  - Giving as much access to data and metadata as feasible
- Tap into the community of users and tools
  - Gives you access to more resources and smarts than you can draw on by yourself

It’s enough for them to support a sufficiently powerful set of standard application interfaces (or APIs) that can be “used” by discovery applications. With the right APIs, you can let your repository do what “it’s” good at—manage your content—and open it up to external discovery applications that do what “they’re” good at—help your users find the right content.

You don’t have to settle for the user interfaces the repository itself provides, or that you develop in-house. If you use standard interfaces and formats for accessing your data, you can install any application that understands those standards. People can use that application to discover your content, no matter who develops that application or where.
Recently a Digital Library Federation group that I was involved in recommended interfaces and standards that will support discovery applications built on top of integrated library systems or ILS’s-- those are the systems that manage our library collections and catalogs. The group recommended standard functions for harvesting and aggregating records from a catalog, for querying the catalog in real time, for handling patron requests for information and services, and for linking to and from our library’s native catalog interface.

Now, there’s already been a lot of standards work in these areas. Rather than try to redo all that work, wherever possible we recommended adopting existing standards, preferably lightweight ones, to support the functions we recommended. In some cases, we did see a need to define a new standard or convention....
A simple call: GetAvailability

Request:
http://onlinebooks.library.upenn.edu/webbin/availability?id=olbp42044&id_type=bib

Response:
<dlf:collection xsi:schemaLocation="http://diglib.org/ilsdi/1.0/
http://diglib.org/architectures/ilsdi/schemas/dlfexpanded.xsd">
<dlf:record>
<dlf:bibliographic id="olbp42044"/>
<dlf:simpleavailability>
<dlf:identifier>olbp42044</dlf:identifier>
<dlf:availabilitystatus>available</dlf:availabilitystatus>
<dlf:availabilitymsg>HTML at loc.gov</dlf:availabilitymsg>
</dlf:simpleavailability>
</dlf:record>
</dlf:collection>

…but when we did, we tried to keep it as simple as possible. For example, this is a simple REST call and XML response for telling an application whether a particular resource is currently available in a library, and how it can be obtained.

This isn’t just a pie-in-the-sky recommendation, by the way. The Digital Library Federation has already managed to get public commitment from most ILS vendors to support an essential subset of our recommendations in future ILS products.
Once you have standard ILS interfaces, all kinds of new discovery applications can use them. For instance, a while back we noticed that our library’s video collection was being used by a growing number of patrons and classes, we created a special application for finding videos in the style of IMDB or Netflix. Underneath this application is the same catalog we use for our library in general. But we’ve automatically harvested the video records from our catalog, indexed them to bring out important aspects like director, genre, and content summary, and we also show instant availability status through calls to our underlying ILS.

(Now, to be perfectly frank, we built this application before the DLF recommendation work started, so we don’t actually use standard interfaces for it that I just described. But the DLF recommendation includes all the ILS functions this application uses, so the next version of this application could well be built on it.)
Interface categories work for more than an ILS

- Many work for repositories as well
- Categories
  - Data aggregation (OAI-PMH, OAI-ORE)
  - Real time queries (SRW, Z39.50...)
  - Patron Access info and services (OpenURL?)
  - OPAC Repository interaction (XSLT, templates, OpenURL...)

Worth working with colleagues and standardizing APIs?
(Some of this is already being done with DSpace, Fedora)

It’s not much of a stretch to go from defining standard interface suites for library catalogs, to defining standard interface suites to digital repositories. There are already a number of standards that repositories can use, like the OAI standards for aggregation, SRW for search queries, and OpenURL and XML stylesheet transformations for inter-application linking.

If enough repositories adopt a well-defined set of standards, then these standards can also support a rich and evolving marketplace of discovery applications, both old and new, for the communities that use the repositories.
And that brings me to the last aspect of architecture I want to focus on. Repositories serve a community of people that can form a kind of *social architecture* to support discovery and use.
Opening the system to its users, contributors

- Let your contributors, users act as expert guides to your resources
  - Let them describe and recommend resources
  - Tie their descriptions into your information and system architecture
- How to make this work?
  - Give them incentives to contribute
  - Monitor what they are doing
  - Accept and adapt messiness

The people that create or use the content of a repository often know more about that content, and its uses, than the people that curate it do. It's possible to collect information about how they use and describe this content to help other users find what they need.

In order for this to work, you need to get them to divulge this information. With the right incentives, many users will be happy to create and share descriptions of their findings with others. You can also monitor what they're doing, to gain insight into what they care about and how they describe it. User-contributed information tends to often be unstructured or otherwise messy, but it can still augment and improve the more structured information architecture that your own curation provides.
PennTags: Sharing our finds

At our university, for instance, we’ve developed a tool called PennTags that lets our users bookmark online information resources, describe and annotate them, and share their annotations with others. There’s an incentive for researchers to use the tool, because it helps them build up bibliographies and reading lists for their own work. And once they’ve done it for themselves, others can see what they’ve bookmarked. (It’s possible to make private bookmarks as well that can only be seen by you or your research group, but the default is public.) It’s a great current awareness tool to see what your colleagues in various areas are finding of interest. And it’s also a very useful way to find out how your users describe and group resources, which might differ from the ways that the creators or curators of the content describe and group them.

PennTags can be used to bookmark both our own content and external content. When a user tags a local resource, the application can pull in metadata about that resource. For instance, here I’ve tagged a book in our local catalog, and the author, title, edition and location information have automatically been included in an annotation.
Thanks to flexible system architectures, we can also embed PennTags data in our native online catalog, so that when a user finds the same book when searching our catalog, they can see that I've tagged it, and they can follow a link back into PennTags to see what related resources might have also been tagged.
Much of the value added by PennTags comes from seeing what the community as a whole is doing. PennTags can automatically build “tag clouds” showing the most commonly used descriptions in a collection, or in a set of search results. These clouds and similar tools let us and other users see what groupings and descriptions of resources are particularly important to our audience. We can use this information to help us better collect and describe resources in ways that make sense to our users. Others in the community can also use this information to help find and classify the resources *they* might find of interest.

Tagging can be messy. In this cloud, we see both a “statistics” tag and a “stats” tag. But we can take note of this to unify the terms in our discovery applications, so a search of one term finds resources cataloged or tagged using the other. We can also look to see what users have tagged as “statistics”, but that we might not have originally cataloged that way, so that users interested in statistics can find both the materials we originally identified and user-identified resources that also contain important statistical information.
Putting it all together: An open discovery architecture

- **Open information architecture:**
  - Create and exploit rich networks of relationships
  - Automated techniques can build and augment complex ontologies
  - Make displays that promote overviews, and teach the information topography

- **Open system architecture:**
  - Design for interoperation
  - Standardize APIs for discovery tools
  - Use the network to multiply your capacity

- **Open social architecture:**
  - Attract, coordinate communities to improve data and systems
  - Embed and analyze user actions and descriptions
  - Make the most of your own expertise and your communities’

So, to pull this all together, I’m advocating opening the architecture of repository systems to enhance discovery and use of their content. We can open up the information architecture, creating and exploiting rich networks of relationships between resources largely through automated techniques. Creating map-like displays of these networks makes it easier for users to get comprehensive overviews of relevant information.

We can make it easier to create enhanced discovery for our repositories by opening up the system architecture, using standard APIs that can support a wide range of discovery tools developed by a broad range of people.

And we can exploit social architecture by eliciting and linking information from the creators and users of our repository content, taking advantage of their expertise as well as our own.

I hope the example applications I’ve shown demonstrate how these architectures can be used to greatly improve the usefulness and the viability of archived content.
This ends my whirlwind tour of architectures for promoting discovery. If you’d like to know more, I’d be happy to answer questions here or later, and you can also find the slides and systems featured in this talk online at these locations.
Accept and adapt messiness
(extra slide)

• **Mess can be tolerated:**
  – Repositories already have a lot of messy metadata
  – New techniques, tools help (auto-correction, fuzzy matching…)

• **Mess can tell us something useful:**
  – Tagging tells us how “real people” classify, find things
  – We can adapt, augment our subject taxonomies accordingly

• **Mess lets us scale up:**
  – Wikipedia lets lots more people build an encyclopedia

• **Mess can be progressively improved:**
  – Online Books Page: From automated subject assignment to curated subjects
  – Penn videos: From hastily cataloged entries to detailed, high quality descriptions
  – Improvements targeted based on community needs

(use if there’s followup questions about this)