DLXS

Update on Plans

About Today

- Our current thinking.
- A chance to discuss.

- Not an unveiling.

We want you to share your opinions and your expertise.

Goals

Create a new and improved DLXS that serves **MLibrary's** needs for preservation and access in areas not served by other repository systems or where DLXS functionality provides a clear advantage, especially to end-users.

What about HathiTrust?

- Original functional objectives met.
- The effort is shared by all members.
- **Everything** will go into HathiTrust... eventually.

What about MLibrary?

- 97 image collections
- 126 text collections
- Finding aids are hot
- MPublishing is expanding

Ten Meetings in May

1. Strengths and Weaknesses of DLXS
2. Brief Survey of Digital Library Systems
3. Collections
4. Repository Relationships
5. Metadata and Content Structure Mapping
6. Indexing
7. APIs
8. User Interface
9. Content Loading and Indexing Workflow
10. Shared Services

Wishes...

- Virtual collections of mixed content
- Collection building tools
- Repository wide search
- Other search engines
- Modern UI
- Publishing tools
- Interoperability
- Simplified and automated loading
- Better code management and distribution
- Faster performance

Strengths and Weaknesses of DLXS

DLXS is an access system more than it is a repository system, but it is both.

Brief Survey of Digital Library Systems

- The most robust repositories are weak on access.
- The most robust access systems are weak repositories.
- Trend may be toward choosing a repository first.
- Access systems are typically built for a specific repository.

Collections

- The ability to build and share collections, at every scale, is very practical functionality essential to the work of libraries and individuals.
- Collecting, ideally, spans all content types, and sources.
- Once a collection exists, possibilities emerge for specialized access to the materials.

Repository Relationships

Repository Agnostic

It would be very useful to have an access system that could work with a mix of common data sources/repositories.

Metadata & Content Structure Mapping

A Mapping Tool

Much of what we do to prepare data and deploy a collection can be considered mapping. We map elements for ingest, transformation, indexing, searching, display, and distribution. This work needs to be easier.

Indexing 1/3

Search Engine Options

- The code, and our content deployment decisions, are too dependent on the capabilities of XPat.
- Solid, complete, search engine integration is good.
- Broader, architectural, dependency of a system on a particular search engine is constraining, and that’s where we are now.

Indexing 2/3

Versatility and Creativity

Functionality is often bounded by the capabilities of a search engine or chosen indexing method. Collections are like sub-climates of content, with potential for deeper understanding and exploration.

Indexing 3/3

Full Repository Searching

The ability to search across the full body of content in a system is important too, and lacking in the current DLXS.

APIs

Separate, Interoperable, Integrable
All uses of the content cannot be anticipated and separation of content from style and functionality simplifies and encourages use.
Versioning of an API stabilizes its use by allowing UIs (and other clients) to lock-in on the version, and be protected from changes as long as that version is maintained.

User Interface

Currently, customizations often require code-diving, resulting in difficult maintenance and migrations in the future.

We need to spend less time doing routine configuration and customization of interfaces.
Accessibility issues must be addressed.

Content Loading and Indexing Workflow

We can reduce the amount of time we spend on repetitive, mundane tasks by establishing requirements, sharing recommendations, automating processes, building better tools, and involving content providers, and content selectors, directly in the process.

The payoff is more, higher quality content, faster turn-around, fewer problems, and more time to do other, possibly exciting, things.

Shared Services

Cloud Potential

If DLXS becomes a system interoperable with multiple repositories, we, or anyone, could provide hosted services for remote repositories. For example, institutions using DSpace could contract with us to provide indexing and access services.

HathiTrust: A Growing Experience

- Ingest quantity, quality, diversity and interoperability
- Repository standards, reliability
- Catalog integration
- Indexing automation, completeness, and performance
- Search scale and performance
- Rights management and access control
- Page image delivery quality and speed
- Collections big, small and dynamic
- Search repository wide, within collection, within item
- Development in a secure and collaborative environment
- Complete system replication

But do we need DLXS to scale to HathiTrust proportions?

Storage, hits per second, search response time, etc.

Automation, workflow efficiency, increased productivity.

Lowering the bar on technical expertise and sharing the effort more widely, involving more people.

Capability to work across institutional boundaries to collectively solve problems.

More Goals for DLXS

Give precedence to scale by building core functionality that broadly accommodates content with minimal intervention.

Provide a path for evolutionary progression of functionality so that advanced features can be deployed for special content and/or advanced users.
Build a system that accelerates content deployment by providing powerful tools that simplify workflow and allow content providers to be involved in a more direct and collaborative manner.

Availability of Resources

Michigan's intense focus on HathiTrust development can give way to DLXS being the top priority.

HathiTrust development effort will become increasingly shared across institutions.

Likely Features of the Next DLXS

- Basic Architecture
- Repository
- Search
- Content Ingest/Deployment
- Collections
- User Interface
- Access APIs
- Security
- Distribution, Installation and Migration
- Usage Statistics
- Broadly Discoverable
- DLXS on the Cloud

Basic Architecture

Separate, interoperable, parts

- Repository
- Search engine
- API
- UI framework

Virtual Collections

- Robust support for virtual collections, including alternative indexing methods, and template themes.

Repository

- Repository agnostic
- Access layer
- Pair it with one or more repositories
- Includes simple, useful repository of its own

Search

Across All Items

- Catalog
- Full-text
- Browse
- Alternative
    - used for collections or specific content types
    - e.g. structured text

Collections

Powerful collection building tools for collections large and small.

User Interface

- Customizable
- Accessible
- Multilingual
- Mobile
- Shareable

- Citations
- Creative Commons / Rights and reproductions

Access APIs

- Strictly versioned for reliable integration with applications
- Supports content sharing standards (e.g,. OAI, Atom)

Distribution, Installation and Migration

- Entirely Open Source
- Distributed through popular code sharing sites
- Designed for ease of migration from one version of DLXS to the next, in part by separation of content from functionality and style.

Usage Statistics

Easy integration of analytics tools.

Broadly Discoverable

Optimized for search engines.