Disclaimer: This white paper represents our research and thinking at a specific point in time. The landscape continues to evolve rapidly, and the authors recognize that the recommendations put forth would reflect more current trends and developments if written at the present time.

Digital Library Systems and UCLA: An Environmental Scan with Suggestions for Near- and Medium-Term Technology Strategies and User Stories / Functional Requirements

UCLA Digital Library Program

Pete Broadwell, Academic Projects Developer
Dawn Childress, Librarian for Digital Collections and Scholarship

August 2017
Updated: February 2018

Executive summary

This document presents the motivations, methods, and results of an environmental scan of digital library systems conducted by the UCLA Digital Library Program (DLP) from November 2016 through July 2017. Potential courses of action for the UCLA DLP are recommended in the concluding section. The authors arrived at these recommendations through a careful balancing of considerations such as current and anticipated staffing models and resources, access to a community of practice and support for new system development, and the needs of the UCLA Library and its partners. In brief, the report recommends short-term use of external systems like CDL's Nuxeo/Calisphere to manage urgent collection publishing needs while the DLP simultaneously evaluates the Samvera (formerly Hydra) environment and eventually adopts and adapts the portions of the ecosystem that are crucial to our future needs.

Background

Software migration is a fact of life for institutions that manage digital information, and this is doubly the case for digital library systems that seek to support an evolving range of digital assets and user access modes. It is certainly also true that the process of migration can be made much more streamlined -- and ideally even managed via upgrades of existing systems -- with the judicious selection of a well-architected set of technologies, which (through planning, gradual implementation, and a probably a bit of luck) turns out to be a good fit for the near- and medium-term goals of a memory institution.

This report presents the findings of an environmental scan of the current state of technologies in digital libraries and related fields, with the goal of helping to guide the UCLA Digital Library's near- and medium-term choices of software systems for digital asset management and related tasks. This work was conducted via focus group sessions with stakeholders at UCLA, interviews with technicians and users at peer institutions, attendance at relevant conferences, consultations with vendors and their product documentation, and participation in online forums.

Current status

The UCLA Digital Library currently uses a variety of software systems to support management and access to digital assets in its numerous collections. These all provide some degree of functionality, but they are not in whole or in part considered entirely acceptable solutions for the future of the digital library "stack" beyond the immediate near term:

- **DLCS:** the oldest software component and arguably the one that is still most effective at managing the entire pipeline from ingest and description through publication is a homegrown system known as the Digital Library Collection System, which is built upon an Oracle SQL database and a Java-based server infrastructure. Nevertheless, its server environment and user interface are in dire need of an overhaul, and its lack of modularity and open source community support means that adding new features (data access APIs, new storage systems, advanced search indices) would be prohibitively difficult.
- **Islandora 1.0 (Fedora + Drupal)**: in contrast to DLCS, this technology is open source, community supported, and built on fairly advanced technology for its data repository (Fedora), index (Solr), and user interface (the PHP-based Drupal content management system). Islandora also has a modular design that is intended to facilitate the addition of new data types and API features

as needed. Yet Islandora 1.0 has proved to be an unsatisfactory “full-stack” solution for the UCLA DLP. Many of these reasons are outlined in the “UCLA Position Paper on Islandora” section of the November 2016 UC Libraries DAMS Technology Report; they include the unsatisfactory level of integration between Islandora and the mainstream Drupal 7 codebase and other factors which made it very difficult for the library to apply its pre-existing expertise in hosting Drupal sites. Some UCLA Digital Library collections are now hosted on the Islandora “stack,” or at least portions of it (most commonly Drupal and Solr), but there is no plan to consider either the now-legacy Islandora 1.0 or future iterations of the stack as viable options for the future of the UCLA DLP.

- **Islandora CLAW (Fedora 4 + Drupal 8)**:[1] The next version of Islandora, which like Hyrax and Hyku from the Samvera community is at an “almost ready” stage of development, represents a substantial overhaul and update: it will be compatible with Fedora 4, use Drupal 8, and promises a variety of other desirable features, such as JSON-LD integration and support for microservices. Such promises, however, are not likely to provide sufficient incentive for UCLA to consider the platform seriously. Adopting Drupal 8 would involve learning and supporting an entirely new PHP codebase. More significantly, the Islandora community is mostly limited to Northeastern US and Canadian universities and one vendor, and our experiences from working with this community in the past have not given us confidence that they will be able to support our diverse and evolving set of use cases into the future.
- **Nuxeo/Calisphere**: Recently, UCLA Digital Library collections have been published in the California Digital Library’s Calisphere access system (which is also a data source for the Digital Public Library of America) via CDL’s central Nuxeo digital asset management system. The viability of this arrangement for future collections is discussed below.
- **DSpace:** The UCLA Digital Library is currently in the process of implementing a new digital collection via the DSpace 6 open-source repository (which is most commonly built on a “stack” of a PostgreSQL database, Solr index, and Java/Javascript user interface), and is also developing a pilot research data repository based on the same software. Like Nuxeo, we consider DSpace to constitute a potential member of a suite of technologies that, although each individual component is not fully sufficient on its own, may in aggregate serve the needs of the UCLA DLP in the short to medium term (see below).

[1] http://islandora.ca/CLAW

Challenge

This environmental scan is intended to aid in the selection of a digital library system (or systems) that the UCLA Digital Library Program can begin to focus upon in the second half of 2017. It is hoped that the software environment developed through this effort will fulfill the unique needs of the UCLA DLP and its constituents in 2018 and later.

The work done for this scan involves a consideration of several digital library systems, with a focus on workflows and other functional needs related to describing and serving digital library assets.

Methods

In exploring the current digital library landscape, the goals of this environmental scan are to: 1) gain a better understanding of our local digital library needs and contexts; 2) explore the systems in use or in development at other institutions and how these meet the needs of their local contexts. The first goal was addressed by conducting internal focus group discussions within the UCLA library and conversations with select end users, the results of which were distilled into functional requirements and user stories. Efforts towards the second goal progressed via active and passive observation of online digital library-related sites and discussion forums, participation at disciplinary conferences, and remote interviews with key staff at peer institutions.

Internal focus groups and discussions

Each internal focus group consisted of 3-7 participants, grouped by their relationship to digital library workflows:

- **Group 1:** Stakeholders who build, manage, and maintain digital library systems at UCLA (Digital Library staff)
- **Group 2:** Stakeholders who interact directly with digital library systems and workflows (metadata, special collections, CCDT)
- **Group 3:** Stakeholders who manage systems or workflows that connect to those of the digital library (preservation, data services, digitization)

A standard set of questions was developed for the focus groups and the external interviews mentioned below, which were then tailored for each session, giving priority to the questions that participants were most keen to address.

Additional informal interviews and discussions were conducted with a variety of end users (students, scholars, and instructors) during the course of this study; some of whom were selected and interviewed, others were encountered during the course of our daily work.

External interviews

The 2016 DLF Forum in Milwaukee coincided with the start of this study and served as an opportunity to explore the current digital library environment beyond UCLA through informal conversations with attendees from other institutions. These findings were supplemented by information gathered via the Code4Lib forum archives. Both venues helped to identify new or developing solutions that UCLA might evaluate for the next iteration of its digital library system. After surveying the landscape, interviews were conducted with individuals and small groups regarding existing or in-development systems and workflows. The 9 interviews representing 8 institutions included participants with positions ranging from developers and data curators to metadata specialists and managers.

Developing a common terminology

During the first focus group, a brief discussion of terms helped to uncover some of the more loaded terminology and establish a common set of terms and their usage in this study. For example, the term “repository” can be interpreted differently in different contexts and might best be avoided except when used to describe a type of service (i.e., research data repository or institutional repository). Instead, using terms that more clearly indicate their function within the stack, such as asset management system or user interface, is preferred. See Appendix 3 for the annotated list of terms.

Scope of the systems under consideration

It should also be noted that the scope of this study is primarily focused on the digital asset management layer (DAMS) and to some extent any primary publishing layers (i.e., the UCLA Digital Library Collection System’s end- user public interface). This is, in part, based on the argument that the activities around managing, publishing, and preserving assets are not optimally served by the same systems and that these activities should be decoupled, yet interoperable. The study’s focus on the DAMS layer reflects the Digital Library’s view that the DAMS is the system’s “core” and should be the starting point for building out publishing, preservation, and other services. The final recommendations, while focused on the DAMS, take into consideration and are informed by preservation and publishing needs as identified in the focus groups and interviews.

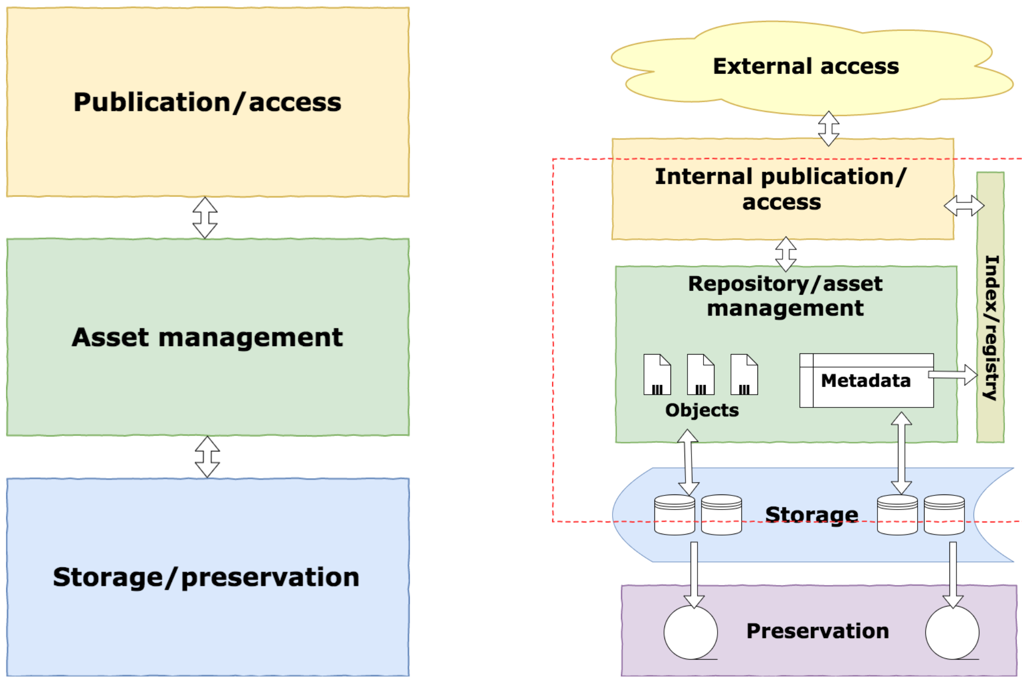


Figure 1: Schematic representations of the primary functional components of a digital library software "stack" and the main flows of information exchange. The version on the right divides the largely abstract sections from the high-level visualization on the left into more discrete components. Note that this study is primarily concerned with the components within the dashed red box on the right: the software modules most closely associated with digital asset management, rather than preservation or publishing of those assets.

Systems not considered in this study

A few prominent software systems for digital asset management and related activities were not included in this environmental scan as viable options for the future of the UCLA Digital Library program. Chief among these is the Islandora platform, which was excluded for consideration for the reasons outlined in the Background section.

We also have not actively pursued OCLC's CONTENTdm system, despite its considerable level of adoption in the digital library field. We did so largely for the same reasons that we are skeptical of the suitability of the other proprietary, vendor-sourced systems considered in the Discussion section: CONTENTdm's "black box" software design and hosted "software-as-a-service" model make customization largely impossible, and although CONTENTdm provides some modularity via APIs that allow the use of a range of publishing front-ends, its large feature "footprint" in the DAMS portion of the digital library stack still severely limits the amount of modularization and customization that would be possible. These factors make

CONTENTdm a poor fit for the extensive range of collection types and paradigms the UCLA Digital Library must support.

Results

Findings from DLF and Code4Lib

Information gathered from attendance at the 2016 DLF Forum and through the Code4Lib forum archives revealed that UCLA is one among many institutions who are less than satisfied with the options that are currently available for digital library solutions. Most solutions fit into one of two categories: 1) closed systems that are convenient and relatively easy to adopt, but that fall short when it comes to desired flexibility, functionality, and extensibility -- especially related to emerging technologies and features; and 2) modular, extensible systems with the potential for implementing the latest advances in digital libraries, but that require extensive infrastructure and adoption of new programming languages and environments. With the exception of a few leaders in the field, all of the individuals interviewed at DLF stated that they are eager to review this study's findings in the hopes that it will illuminate a path forward at their institutions, as well. Likewise, while a few digital library solutions were discovered via the Code4Lib discussion list archives, most conversations were unanswered queries or discussions around the same few solutions. In reality, there are few "ready" options, rather "almost-there" solutions that some are investing in and that many hope will be viable in the near future.

Here is the short list of specifics that were found:

- The distributed development model of the Samvera (formerly, Hydra) ecosystem seems to have the most momentum in adoption and development. There is a lot of anticipation around the two current Samvera stacks: Hyrax and Hyku.
- Islandora adoption is slower and some are dissatisfied with their implementation; in most cases the pain point seems to be Drupal.
- At least 1 institution has adopted MongoDB for a digital library project (for early printed science books).
- Reed College is developing their own system using Rails + AngularJS.[2]
- Medusa, developed by UI Urbana-Champaign, is a Ruby on Rails + Postgres DB system based on the Archivematica workflow, with a focus on asset life cycles. Medusa is used for the Illinois Data Bank and is being adapted for digital collections.[3]

[2] https://rdc.reed.edu
[3] https://digital.library.illinois.edu

- Michigan is developing a preservation workflow for ArchivesSpace > Archivematica > DSpace.[4]

Findings from local focus groups

The local focus group discussions were quite productive and led to a long list of features and use cases, briefly summarized here. The complete list of features can be found in the **User Stories/Functional Requirements** section at the end of this report, with the DAMS-related features summarized in the DAMS Feature Matrix in the "Discussion" section below.

UCLA's immediate and near-future use cases and projects extend beyond the Library Digital Collections which are currently managed and published via DLCS. UCLA's Digital Collections include both public and UCLA-restricted content, primarily from Library Special Collections and other internal library locations. Other collections that the new DL system would support include the Center for Oral History Research collections and partner library collections from the Clark Library, Chicano Studies Research Center, and the Bunche Center for African-American Studies. Ideally, the new DL stack would also accommodate the Library's grant-funded projects, like the International Digital Ephemera Project, the Syriac and Arabic Manuscripts project, VSim, and PRRLA (the Pacific Rim Research Libraries Alliance meta-catalog). Finally, there is much interest in a "Digital Stacks" collection for materials like PDFs and media files that are acquired by curators, but are not hosted by vendors or publishers. These often do not fit into the digital library collections and have viewing and download restrictions that are not met by current digital library access policies and functions.

Ideally, one content-agnostic DAMS could be used as the core infrastructure for each of these use cases.

In addition to the use cases outlined above, a list of desired features emerged from the focus groups around asset management, user interfaces, and other layers. Here, the report outlines and summarizes these features.

Asset management features tended to be the focus of internal focus groups 2 and 3. These are divided into three categories: checks and preservation, asset and metadata workflows, and other features.

[4] http://journal.code4lib.org/articles/12105

- *Checks and preservation:* There was a lot of interest in more robust features around file integrity and audits of storage and changes. Any future DAMS should provide fixity, logging, and reports. Including metadata related to preservation and conservation in addition to descriptive metadata was also an important feature mentioned in 2 of the focus groups. Finally, automated or facilitated deposit to a preservation layer was identified as essential by all groups.
- *Workflows*: The ability to bulk ingest both assets and metadata without intervention from a programmer was at the top of the workflows list. Users of the DAMS also called for more options when editing and exporting metadata for internal use, such at batch editing and export format options. An easy and clear system for rights management tied to access as well as integration of authority reconciliation services were also popular features from DAMS users.
- *Other features:* It was noted by many that DLCS currently supports a rich data model, and that retaining this is essential. This seems in part to be a reaction to the rather generic data model used in CDL's instance of Nuxeo. Other features noted were more granular user roles, embargos and expiration dates, and restrictions by location, in addition to Shibboleth/single sign-on capabilities.

Front-end related features were introduced primarily by focus group 1, with some echoed in group 2. The ability of users to obtain collections metadata for research was high on everyone's list, as was user download of images in a variety of formats (à la Flicker) and embedding of DL content into other platforms (CCLE). Improved search/browse and viewing options (page-turning, gallery views, organization by box/folder) were discussed in group 2. Finally, user-created content in the form of self-deposit (IR, data repository) and
crowd-sourcing of annotations and metadata was identified as an important future direction by group 1.

Other layers proposed by focus groups include supporting harvest of collections via OAI-PMH and/or ResourceSync; integration with federated collections such as DPLA, layer-agnostic APIs for public consumption of collections; and GIS and data visualization layers for exploring collections. There was also a bit of discussion around relating, linking, and connecting to other services, but few concrete examples or methods were offered. This suggests that we need more research into the use cases and methods at other institutions and perhaps a better understanding of what services would be applicable in the local context.

Findings from discussions with end users

Many of the front-end features mentioned by the focus groups were echoed by the students and faculty, if not always in the same terms. The need for more sophisticated search/browse options was at the top of the list, followed by more viewing options for various formats (book readers, gallery views for images), For scholars and instructors, sharing options such as downloading or embedding content (images, metadata, text files, etc.) are at the top of the list; making DL content usable in instructional materials and for research and various scholarly projects. Essentially, the ability to reuse Digital Library content in new contexts would be a boon for many types of users. That being said, most scholars interested in using collections data for their research prefer simple CSV files and bulk downloads over less-accessible APIs and institution- or project-specific programming interfaces.

It is important to note here that the UCLA Library and the Digital Library are themselves end users of this service. Many of the features desired are for our own ends, so that we might do better work more efficiently, and to develop richer digital collections for our users. For example, developing technologies like APIs, ResourceSync, IIIF, etc. to interface with our digital collections allows the DL to more easily build out new projects and platforms without creating new core infrastructures. Of course this has the added benefit of opening our collections to communities outside of the library, allowing them to construct their own projects and co-locate our collections with those of other institutions. Likewise, developing programming interfaces that allow users to explore our collections as data, annotate items, or transcribe texts serves our own needs as much as it might those of our users, if not more so.

For a more detailed look at these findings, see the **User Stories/Functional Requirements** section at the end of this report.

Findings from interviews with peer institutions

We conducted interviews with representatives from the peer institutions listed in Table 1, seeking whenever possible to speak to library administrators, developers, and users/stakeholders (e.g., digital library librarians and metadata specialists). See Appendix 4 for a list of the main topics discussed. Note that the majority of the institutions in UCLA's general geographical vicinity seem to be trending towards Samvera-based DAMS solutions. North Texas and the Getty are to some degree outliers on either side: UNT has a homegrown solution that is managed by a single developer, whereas the Getty use a dizzying array of commercial, open-source, and homegrown technologies, which requires a very substantial staff to manage.The rest of the institutions fall somewhere in the middle (as would UCLA, most likely).

Institution	**Current technologies**	**Anticipated technologies in <=5 years**	**Notable paradigms (org, development, content philosophy)**
University of California, San Diego	Hydra/Rails front-end, homegrown back-end repository (supports Hydra 3 APIs)	Samvera family (want to adopt Fedora 4 repository, push linked data/RDF capabilities)	Content-agnostic Use sitemaps, Rails micro-data for APIs Use Excel sheets to manage ingest
Getty Museum/Getty Research Institute	ExLibris suite: Alma (catalog) Primo (search) Rosetta (pres-ervation). Also, lots of other software!	Plan to build new IIIF-based front end, move EADs out of ArchivesSpace, adopt linked-data systems like Arches	Incredibly complicated software environment, multiple vendors + in-house dev. Requires a lot of staff.

California Digital Library	Calisphere (pub)
	Nuxeo (DAMS)
	Merritt (storage)
		Samvera family, probably	Nuxeo=DAMS
			eScholarship=repo
			Single main back-end developer

University of North Texas	Aubrey (access) Coda (preservation) Both homegrown Python/Django apps; Aubrey uses Solr, Coda ResourceSync!	Likely the same homegrown Python/Django system, barring a major shift	Content-agnostic Single developer Mediated upload only Support harvesting APIs but LOD skeptic
University of	None	Samvera family	LOD proponent

California, Davis		(emphasis on "in-a-box" solutions, e.g., Hyku)	
Stanford University	Hydra family: Blacklight, SearchWorks, Hydrus (IR), Fedora (library website runs on Drupal though)	Samvera family: replace Hydrus with Hyrax -- supports self-deposit, one-step approval	Blacklight Search-Works provides the "union catalog" of collections via Solr
University of Illinois at Urbana-Champaign	Ruby on Rails + Postgres (Medusa) for coll registry and DL front-end, storage/preservation NOT Hydra, file system-based, used RabbitMQ for coord.	Same. They have an IR in DSpace that they want to migrate to Medusa, and aso plan to migrate their special collections from CONTENTdm to Medusa	Devs revolted when they tried to adopt Hydra (esp. Fedora); Medusa was built for Illinois Data Repository but also is used (with a different front-end) for DL collections
Duke University	Fedora/Hydra all the way for DL colls! Still using Fedora 3, though -- ran into problems with Fedora 4 migration.	Planning to adopt Hyrax/Samvera and move to Fedora 4 eventually.	Previously used in-house Python/Django platform; prefer Samvera's integration with preservation layer

Table 1

Explanations of paradigms/models shorthand

Hydra vs. Samvera: To disambiguate the two, we are using "Hydra" to refer to Hydra 3 and earlier versions; we are using "Samvera" to refer to the current version, previously known as Hydra 4.

Content-agnostic: Everything in the main DL/DAMS infrastructure is just a "thing," meaning that publications in the institutional repository, audio and video collections, user data in the research data repository, etc. all can be managed by the same system.

Single developer: DAMS development and sometimes the production ingest process has a “truck number” of 1 -- only one person really knows how it all works. This seems to be an evolution of the “database admin” model for back-end storage management, where a single, valuable DB admin keeps the system running, and shares the same benefits and drawbacks.

Discussion/analysis

We evaluated the options discussed below through focus groups, interviews, and research. Here we attempt to outline relevant features, level of modularity, relative ease of adoption, community support, and the perceived sustainability for each. We do not include scalability as a major dimension of our analysis because it has too many aspects (e.g., storage scalability vs. collection processing scalability) and is difficult to generalize, as one institution's "scalable" may be another's "hopelessly limited." The User Stories/Functional Requirements appendix therefore contains the most pertinent considerations of the "scalability" of any given system for the UCLA DLP.

Samvera (formerly Hydra) family (Fedora 4 repository + Samvera head on top). The two current Samvera stacks are **Hyrax** and **Hyku**. Hyrax is the standard Samvera stack and is modular, customizable, and extensible. Hyku is built on Hyrax and represents the "in-a-box" option that is meant to be used as-is. As we anticipate that our use cases will require significant customization, we are primarily considering Hyrax.

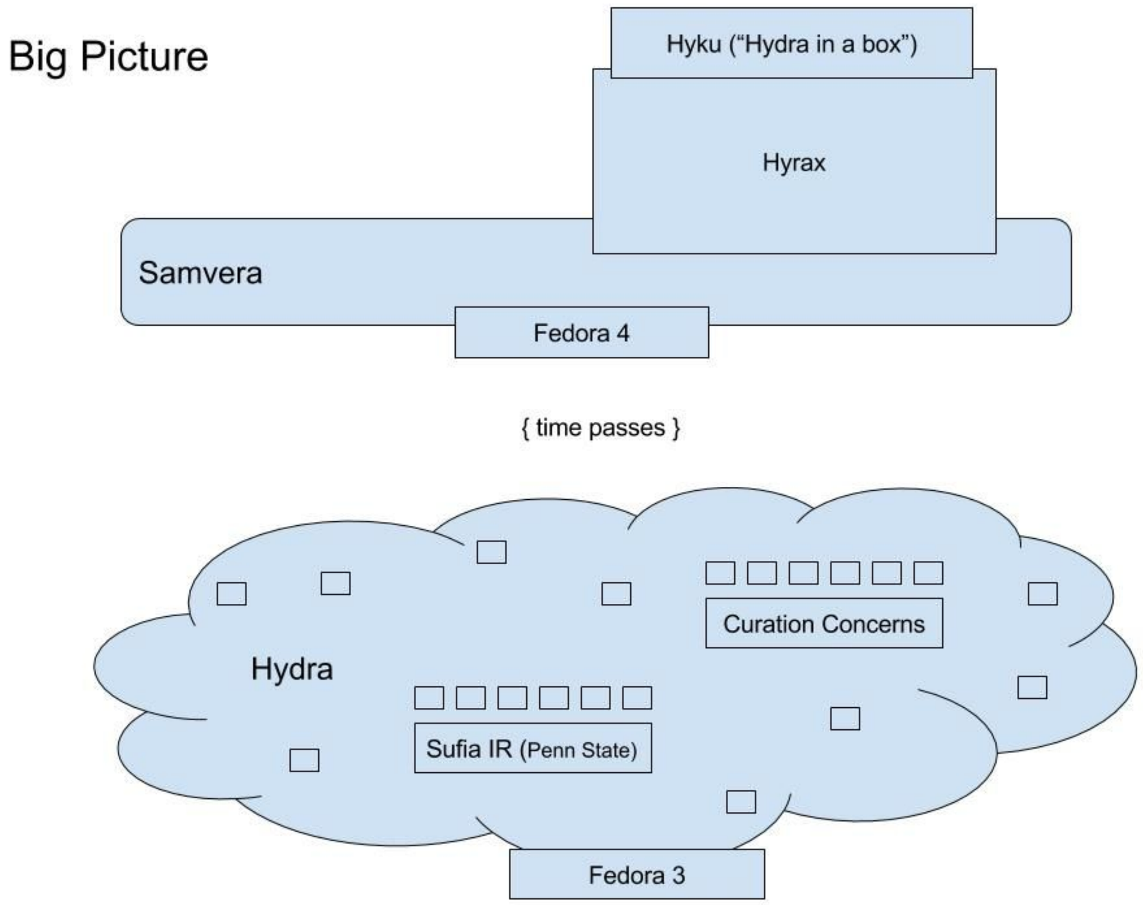


Figure 2: A "big picture" view of the relationship between "legacy" Hydra (based on Fedora 3) and the Samvera implementations (Hyku and Hyrax) being built on Fedora 4.

- *Features*: Hyrax would give us most of the asset management features that we want, such as fixity, audits, and event logging; asset and metadata organization; bulk asset and metadata ingest; and granular user roles.
- *Modularity*: We could potentially use Hyrax as our all-in-one DAMS. The Hyrax system is built to be quite modular and to support multiple ingest and publishing layers and APIs ("heads").
- *Ease of adoption*: Programmers would need to learn Ruby/Rails, need to administer Fedora, etc. Most of our interviewees strongly asserted that learning the new languages/frameworks was well worth the effort in terms of the benefits they offer. A greater source of potential "adoption fatigue" was the tendency for large multi-stakeholder projects like this to churn and drift, as arguably has happened with the development of the Fedora 4 spec, PCDM adoption, and the debate between whether to adopt the Curation Concerns or Sufia codebases for the DAMS layer.

- *Community support*: Lots of peer institutions are now building their digital library systems on parts of the Samvera stack, and more are planning to do so in the future, especially among nearby institutions on the west coast.[5] Open-source practices and philosophies are foundational to the ecosystem.
- *Sustainability*: Given the number of institutions focusing on the Samvera family for the immediate future as well as the ~5-year horizon, the outlook for long-term support seems the strongest for Samvera among all open-source solutions.

Nuxeo/Calisphere through the California Digital Library

Nuxeo is an open-source (largely Java-based), commercially supported digital asset management system that serves as the repository and asset management layer for Calisphere, the UC's central digital library platform hosted by CDL. Nuxeo is the staff interface for creating collections and preparing them for publication while Calisphere is the web application that is the public interface to UC's digital collections.

- *Features*: Nuxeo provides basic asset management, but CDL's instance has a limited data model and bulk ingest only supports simple objects at the moment. Nuxeo would likely not be a good candidate for an all-in-one DAMS, but could potentially meet the needs of digital library collections and special projects with some effort. Many of the asset management/preservation features that we want may be available, but are only available by request to CDL at this time. Restricted access to the database and other services within the Nuxeo/Calisphere ecosystem will be a pain-point.
- *Modularity*: Nuxeo/Calisphere embraces the modular approach and there are several ways that UCLA could make use of Nuxeo: 1) use CDL's instance of Nuxeo, but build own front-end(s) and push to own Solr; 2) host own instance of Nuxeo and push to Calisphere and own front-end(s); 3) no Nuxeo, but push to Calisphere from whatever new system is developed. However, the modularity of Nuxeo does not compare to Hyrax/Hyku.
- *Ease of adoption*: The DL has already adopted Nuxeo/Calisphere for publishing publicly-accessible collections. Long-term adoption would require: boosting CDL's storage and server capacity to support the increased system load; developing custom front-ends for harvesting UCLA-only collections and assets for special projects (IDEP, Syriac/Arabic); negotiating with CDL for

[5] Consider the list of Samvera "Partners and Implementations" from the wiki: https://wiki.duraspace.org/display/samvera/Partners+and+Implementations compared to the more generic list of "Samvera Partners" on the official site: https://samvera.org/.

more access to the underlying database and systems; working with CDL to expand the data model to meet UCLA needs.

- *Community support*: Although the Nuxeo software is nominally open-source (https://github.com/nuxeo), the codebase is quite complex, and support would come primarily from the Nuxeo corporation and from CDL. CDL would likely have limited capacity to support active development in areas they are not pursuing.
- *Sustainability*: It is not clear now long CDL will stay with Nuxeo. It is very possible that they will opt to move to a Samvera solution in a few years. There are no other digital libraries using Nuxeo as far as we know.

DSpace

Originally a joint venture between HP Labs and MIT, DSpace is a widely used open-source repository system that is considered especially well suited to providing access to institutional publications. Modern installations are usually based upon a PostgreSQL database, Solr index, and Java server applications, including interfaces based on XSLT or JSP. Future iterations of DSpace will be based around a UI built on the Angular2 Javascript library, which is a significant change to the codebase.

- *Features*: As described above, DSpace's features are tailored to support institutional repositories of publications, providing a fairly flexible data model and exposure of metadata via OAI-PMH. Many of the other features we will want in the next several years, however, would have to be provided by customized modules adopted from open-source projects or written from scratch (e.g., custom metadata ingest workflows, support for complex objects and multimedia).
- *Modularity*: DSpace has a fairly monolithic structure -- certain components (e.g., the underlying database and the front-end UI) can be swapped out, but it is generally considered an "all-in-one" solution for the problems it is meant to address. Nevertheless, its open-source nature means that many plugins and expansion modules are available for adoption, although using them is often not "plug-and-play."
- *Ease of adoption*: Medium -- it's not a "turnkey" system, but also fairly easy to stand up. The codebase and underlying technologies: Java, SQL, Javascript, XSLT, OAI-PMH -- are fairly mainstream if a bit old-school.
- *Community support*: There is a large and still quite vital community of institutional users of DSpace who contribute modules, documentation, and are generally responsive to requests for troubleshooting suggestions.
- *Sustainability*: As mentioned above, the upgrade path from DSpace 6 to 7, and particularly the rebasing of the interface technologies on Angular2 may

prove problematic and a source of user attrition in the long term (though it might play into the DL's development of greater Angular2 expertise). Yet the installed base of DSpace 6 and below remains quite large, and the underlying technologies are likely to be quite long-lived.

Medusa

Medusa is a home-grown digital library system developed at the University of Illinois at Urbana-Champaign libraries, using Ruby on Rails, Postgres, and a file system-based preservation layer as its main components. See documentation (a bit outdated) at https://wiki.duraspace.org/display/hydra/Medusa, also Github activity at https://github.com/medusa-project. They briefly considered basing the repository on Samvera and especially Fedora, but the developers "revolted" due to the inflexible data model and general bugginess of Fedora 4. The Illinois DL team found it quite liberating to ditch Samvera and build their own system, which they said they were able to stand up in just a few weeks.

- *Features*: Medusa is based on a home-grown preservation repository layer that uses the file system as a main storage layer (with backup via NCSA services and Amazon Glacier) and stores metadata in a Postgres database, supporting complex, hierarchical data models. Additional services are built on top of this layer, using a Ruby-on-Rails-based workflow management system loosely based on the Archivematica workflows, a Solr index, selective RDF triplestore, etc. They also use a homegrown IIIF server, Cantaloupe.
- *Modularity*: Medusa's components are connected into a truly distributed system, coordinated via RabbitMQ messages.
- *Ease of adoption*: Given that Medusa's design is to some degree tailored to the hardware and software environment at UIUC, it likely would not be easy for us to adopt the entire system. But Medusa's basic model -- file system + backup services for object storage, Postgres/Solr for metadata and searching, Rails for the front-end, distributed design -- is a promising one for a home-grown system.
- *Community support*: Although Medusa benefits from being built on open-source, community-supported components, UIUC is the only institution that uses the full stack. If UCLA were to join them, an institutional community of two would be an awkward size.
- *Sustainability*: Lately they have begun to experience some age-related creakiness in the system and are dealing with the consequences of decisions they would have made differently now (especially regarding access/authentication), but it's not clear this is any more of a challenge to

sustainability than if they were using a vendor- or wider community-supported system.

Quartett
Quartett is a commercial digital asset publishing solution from vendor Adam Matthew, currently in development. The Library purchases many databases from Adam Matthew and therefore the relationship with the vendor led to our being asked to consider this product. There is at present no release or documentation that could be used to evaluate the product. The vendor shared a few wireframes and outlined a few specifics: Quartett will be a hosted solution with a hefty price tag on a per-project basis. Pricing will be tiered, based on total size of assets and chosen publishing features (maps, visualizations, etc.). There are currently no plans to support APIs, Fedora connectors, or other solutions that would allow integration with other systems or layers. Development plans include some very user-friendly features at the asset management layer and for creating exhibits; however, the proprietary, block-box approach does not meet the modular and extensible system requirements of UCLA's Digital Library. This report does not recommend further evaluation of Quartett as a digital library solution. Perhaps, down the road, this could be a viable front-end for a grant-funded boutique project that sits atop our digital library stack, but that remains to be seen.

Libnova
Libnova is run by a company out of Spain. One of their representatives approached the DL Head at CNI to review their products and services. Libnova provides a suite of tools and services to support digitization workflows, focusing on management of digitization projects and image processing. Libnova does have a modest publishing solution, LIMB Gallery, with search and browse, hi-res image viewing, page-turning, and OAI-PMH capabilities, but like Quartett, it is a proprietary, black-box solution that will not meet the requirements of the UCLA Digital Library. Libnova's tools seem to be more suited to the digitization activities that happen in SRLF.
http://www.libnova.com/en/systems-for-digitization-and-diffusion/

DLCS 2.0 (roll-your-own)
A final option is for UCLA to build its own digital library system. The current system has served the program well for a decade and institutions like the University of North Texas and smaller schools like Reed College have had success with this approach. More discussion of the potential for a roll-your-own version is discussed below in the "Conclusions" section below.

The Contenders

Of these systems, only three seem to be potentially viable solutions in the current landscape: Samvera’s Hyrax, Nuxeo (the DAMS serving Calisphere), and a from-scratch “DLCS 2.0”. Below is a Feature Matrix outlining the **DAMS-related features** that came out of the focus groups and the potential for each of these systems to meet local needs.[6] A possible DLCS 2.0 (roll-our-own) has been left out since the potential is infinite. We have also noted features that we would likely develop in-house in addition to adoption of other systems; in the case of Hyrax these could be contributed back to the community as modules or “heads.”

DAMS Feature Matrix			
Asset Management *(checks/preservation)*	**Hyrax**	**Nux/Cal**	**Notes**
Fixity	✓*	--	*via Fedora
Storage audits/reports	✓*	**	*PREMIS event service in Fedora: https://wiki.duraspace.org/display/FF/Design+-+PREMIS+Event+Service **Request from CDL?
Status (workflow tracking)	✓+	✓-	Hyrax more granular than Nuxeo
Change log/audit trail	✓	?**	**Request from CDL?
Deposit to preservation layer	✓	✓	Both options are ready for this; we might potentially develop our own feature in-house
Asset Management *(workflows)*	**Hyrax**	**Nux/Cal**	**Notes**
Bulk ingest of assets (recursive) and metadata	✓+ *	✓- **	*Recursive folder ingest enabled **No recursive or complex batch ingest With either, we will likely need to develop scripts or tools for our needs, but this might be easier with Hyrax
Form for manual entry of metadata	✓	✓ **	**Nuxeo has spreadsheet mode

[6] For more information on features available in the Samvera (Hydra) ecosystem, see the Sufia Feature Matrix (https://github.com/projecthydra/sufia/wiki/Feature-matrix) and the LDCX lightning talks (https://drive.google.com/drive/folders/0B20rJmC1X-AmLWNaXzZMU0cxb2s)

Export metadata as csv or other structured formats for internal use		--	Need more research on Hyrax
Rights management tied to publication/access	✓	-- **	**Calisphere = public items only. CDL's Nux has access field that we could use if developing our own front-end
Authority auto-fill/reconciliation service built in / LOD (Questioning Authority)	✓ *	**	*See Cornell presentation: https://docs.google.com/presentation/d/1L3rhAtHUqyxS2wo_cBo5Ny1iAx9BiBZe5zKAr6KYXKA/edit#slide=id.g1daf2d4a0f_2_19 **CDL is interested, but no plans
Unique identifier/handle creation and management	✓	--	We could probably set this up in Nuxeo, but would have to develop it ourselves. Baked in to Hyrax
OCR on ingest... this is not a high priority, but we mention it since there is a possible Samvera (Hydra) solution (Plum) and it came up in the focus groups		--	Possibly with Plum (Samvera): https://github.com/pulibrary/plum. Also, the libnova suite has Tesseract baked into its digitization workflows solution, so adoption of a similar solution might be a more logical place for OCR.[7]
Asset Management *(other features)*	**Hyrax**	**Nux/Cal**	**Notes**
Maintain robust data model	✓	--	DLCS has a robust data model and we don't want to lose that, although it could probably use some streamlining.
User roles well-defined and somewhat granular	✓+	✓-	Hyrax has much more well-defined and granular user roles than Nuxeo
File download from DAMS (for internal)	✓	✓	
Set embargos and time-restrictions (copyright expiry dates)	✓	--	
Restricting user access - not just Shib, but IP	✓	--	This is essential for our collections and workflows. We cannot do this with CDL's instance of Nuxeo unless we build frontend
Asset management for non-published assets	✓	✓	

[7] OCR as a service would likely be tied to Fedora's message queue, not coupled to a UI, even one as nice as Plum's. That doesn't mean there shouldn't be a UI for OCR cleanup, but the DL is in favor of a disintegrated / modular as possible system so that best of breed apps can be swapped in as needed.

(with metadata and for retrieval)			
Front-end	**Hyrax**	**Nux/Cal**	**Notes**
Self-deposit (user contributed content for IR/data repo)	✓		This would be useful for Digital Stacks
Crowd-sourcing of assets a possible longer-term goal	✓ *		* Hyrax self-deposit feature could make this possible
User-friendly asset export formats - also restricted in some cases to low-res			This is mainly front-end, but might have implications for the DAMS - does the DAMS need to auto create certain derivatives in certain scenarios?
Multiple ways of viewing collections - by structure (box, folder), gallery (all of it), by series...			Again, front-end, but a DAMS that supports display structure from metadata would be great. This is something that we would potentially develop in-house to meet local needs.
Export metadata in various structured formats from public interface			Maybe this would happen from a query interface in the “Collections Lab”? Need to be able to query Solr prolly
More robust searching/browsing/faceting across collections	✓+	✓-	Front-end, but dependent on good DAMS structure and good metadata!
Other Layers	**Hyrax**	**Nux/Cal**	**Notes**
Support harvesting (OAI-PMH, ResourceSync)	✓	API only	Hyrax has ResourceSync enabled
API that is agnostic of other layers for access to assets	✓ *	✓ **	*Hyrax API: see https://drive.google.com/drive/folders/0B20rJmC1X-AmLWNaXzZMU0cxb2s; also API-X (Fedora): https://wiki.duraspace.org/display/FF/Design+-+API+Extension+Architecture and Fedora API: see http://fedora.info/spec/ **Calisphere -> DPLA We would also want to develop multiple APIs for access in different contexts - Hyrax enviro would give us more possibilities
GIS tools/data vis layers	--	--	GeoBlacklight and other services would be layered, developed as appropriate. Likely used via APIs and Solr

Support for large packages of things like VMs, containers, code		--	This doesn't seem to be an immediate need, but there were some "what-ifs" mentioned during focus groups.

Conclusions

The following section outlines a few of the most likely scenarios for near- and medium-term technological adoption in the UCLA Digital Library. These discussions are not meant as a full endorsement of any single approach and technology stack; in fact one of our primary recommendations is that any strategy of adoption should first involve pilot studies and limited deployment to determine the suitability of the new technology and to avoid committing too many resources to a path that may ultimately be a false start or at best a short-term solution.

To help avoid the worst of these dead ends, we also speculate on the amount of staff resources, retraining, and other investments that would be necessary to produce an acceptable timeline to viability, and we include a section on additional considerations related to how the central DAMS/repository software this scan focuses on would interact with some of the publishing/UI and preservation layers that are most likely to accompany them.

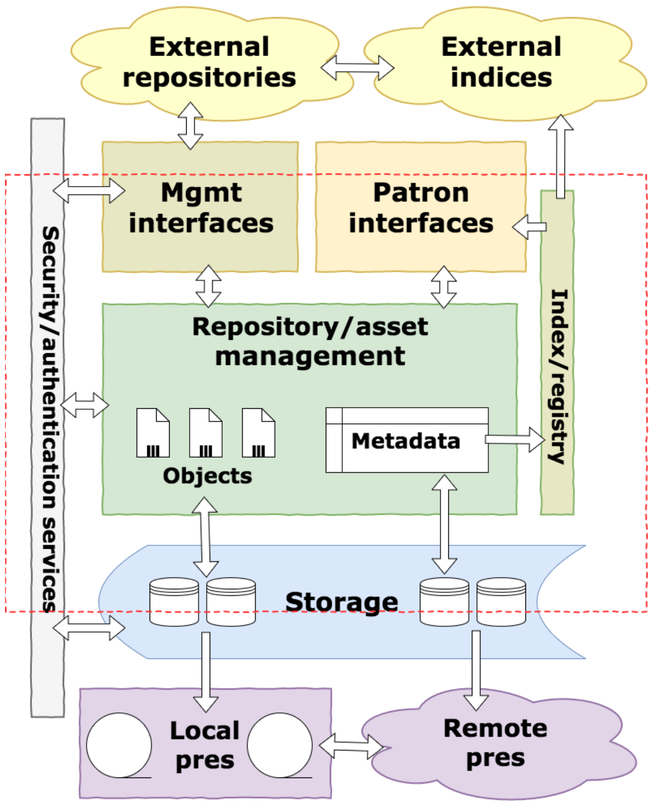


Figure 3: A detailed view of the components and modules we expect that a current digital library software "stack" would need to incorporate or interact with.

Potential scenarios

Outlined below are a few viable scenarios of technological adoptions in the short and medium-long term. This section attempts to evaluate/discuss the implications of the scenarios, including staff resources and re-skilling, community and support, and integration with front-end layers, preservation systems, and other layers.

Option 1: CDL's Nuxeo/Calisphere

Short term: Use CDL's Nuxeo to publish publicly-accessible collections to Calisphere. This will help to relieve the ingest portion of the digital collections backlog and allow CCDT to resume accepting digital project proposals in the interim. Nuxeo could also serve as a solution for UCLA-restricted assets, the Syriac and Arabic manuscripts., and other projects as needed if programmer time is allocated to spin up additional front-ends, connecting to Nuxeo via its API. Blacklight, Angular2, and Django are a few options; the Calisphere code (Django) could also potentially be repurposed. Some upfront front-end development could buy a year or two to develop a more robust DL system.

Long term: Long-term adoption of CDL's Nuxeo would require further development on a number of fronts. Custom front-ends for harvesting UCLA-only content and for special projects (for example, IDEP and Syriac/Arabic Manuscripts) would need to be developed. UCLA would also need to work with CDL to boost CDL's storage and server capacity to support the increased system load from existing and future UCLA assets and traffic. It would also be wise to negotiate with CDL for more access to the underlying database and systems as well as to expand the data model to meet UCLA collection needs. This being said, it is unlikely that there will be a CDL Nuxeo option in the next few years. CDL will eventually migrate to Samvera, so UCLA would need to be willing to take over the Nuxeo instance or migrate to a new system at that time. Another option would be for UCLA to host and pay for its own instance of Nuxeo; although this falls under the "Roll our own" option listed below, it should be noted that If UCLA were to consider adopting its own instance of Nuxeo, it might be prudent to explore other commercial DAMS, like Canto Cumulus, which may have more robust features and integrations than Nuxeo.

Option 2: Fedora/Samvera (formerly Hydra) family: Hyrax & Hyku

Short term: This path would begin by conducting pilot experiments of Hyrax, setting up a local instance of Hyrax, testing its built-in features and also investigating how other modules could be added to it. At the same time, we would want to gain familiarity with the languages and frameworks it uses, perhaps holding Ruby bootcamps for developers, and also experimenting with a Blacklight front-end to gain experience with the interaction between Rails and Solr. It would be helpful to identify a target collection or two to focus on during these experiments, or perhaps adopt it for one of our grant-funded projects. A short-term hire or two to take over day-to-day activities could help to facilitate these experiments by allowing the permanent DL developers to get up to speed on the Hyrax system. We would also want to take advantage of the Hydra/Samvera community to help bootstrap the DL developers' knowledge.

UCLA would need to determine where we stand relative to the points of divergence within the Samvera ecosystem and community (assuming the entire community hasn't coalesced around a single stack design in the meantime, which seems highly unlikely). In particular, we'd need to evaluate whether Hyrax offers a sufficiently comprehensive feature set out of the box for the immediate future, if Fedora 4 is sufficiently mature and whether or not the Hyrax bridge to Fedora 4 is sufficiently mature, or whether we'd need to contribute development in these areas. Hyrax does offer potential for modular development of the "missing" features; however, learning a new programming language and framework (Ruby/Rails) takes time and it would likely be some time before UCLA could make significant contributions to the Samvera ecosystem.

Long term: If the Hyrax pilot is successful, full-scale adoption would likely require 2 temporary hires (1-2 years) to keep the DL program afloat while the DL developers get the Hyrax system and its integrations up-and-running. Since the infrastructure required for this option would be significant, it would be wise to do an up-front inventory of the technical resources needed (servers, storage, AWS instances, etc.) and establish these early on so time is not spent negotiating these requirements at the point of need. Long-term adoption of Hyrax will require at minimum a Linux environment with Ruby, Redis, Rails, and Javascript; installation and maintenance of a Fedora 4 repository, Solr, ImageMagick, FITS, LibreOffice, and Hyrax itself; and re-skilling of digital library staff, including Rails and PCDM. For information on what the management Hyrax likely will involve, see the Sufia Management Guide.[8] It would also be beneficial to coordinate with peer institutions to integrate more

[8] https://github.com/projecthydra/sufia/wiki/Sufia-Management-Guide

closely into the Samvera/Hyrax development community -- this would give UCLA the structure and support needed to adopt the new technology.

Option 3: Roll our own!
Short term: We would need to go back to first principles to decide on the base technologies on which to build the homegrown DAMS. This could take a while to determine, and likely would be influenced by the pre-existing specialties (or technological aspirations) of our developers. A new position paper would be required to evaluate core technologies and approaches.

Long term: Implementation of the system also would take quite a while; we would need to continue to use legacy and/or stopgap systems in the meantime. Ideally, building our own system would allow us to tailor it to the unique needs of the UCLA Digital Library, ultimately streamlining processes to be more efficient than they probably ever could be with an externally sourced solution. Yet it is also important to keep in mind that the “unique needs” of the DLP are more or less constantly changing in response to outside influences such as grant funding, the emergency of new partners from on campus and off, etc. In this environment, a community-based approach and particularly the ability to take advantage of contributed modules as well as others’ expertise with the system may ultimately prove to be a more desirable situation.

Conditions for success: A handful of institutions from our external interviews had successfully implemented some or all of their digital library system more or less from scratch, though in most cases even the “from scratch” components were based on open-source packages at some level. Two places -- North Texas and UIUC -- had built the majority of their DL stack from scratch and remained generally satisfied with it, while two others -- UC San Diego and Duke -- had designed home-grown components of their DL stack (Duke had a home-grown Python/Django-based front-end and DAMS, UCSD had a custom back-end) that they either had replaced or were planning to replace with components of the Samvera stack.

Though this sample size is admittedly very small, we can summarize some observations regarding the conditions that seemed necessary at the above institutions in order for them to roll their own DAMS layers and be at least generally satisfied with the results:

- Dedicated DAMS stack developers who have few if any other responsibilities other than developing and maintaining the system. The DAMS developers

also seemed to be fairly involved in the ingest of data into the systems once they were running.
- Few external time/funding pressures, i.e., no grant-funded projects that need a working site *immediately*, which might have prompted them to adopt a pre-built system that works "out of the box," and/or make hasty decisions regarding the design of their home-grown system that they would come to regret later.
- Collaborative, nimble decision-making between DAMS developers and architects with (apparently) little outside intervention, OR a single main developer who makes all the decisions.
- Developer familiarity with the technologies used (programming languages, DBs, APIs) was *not* cited by any of these institutions as a crucial requirement, though they still often gravitated towards technologies they already knew.

If at least some of these conditions are not met, our attempts at developing our own DAMS, in whole or in part, are unlikely to succeed.

Recommended course of action

Follow through with **Option 1** short-term scenario and continue to use Nuxeo to relieve pressure on the DL, possibly developing simple interfaces if needed for high-value projects that cannot be published via Calisphere. The report does not recommend adopting Nuxeo as a long-term solution, whether CDL's instance or its own.

Meanwhile, focus on implementing **Option 2**: Option 2 outlines a testing and evaluation phase; however, given the short list of options and the strong case for Hyrax, this report recommends that the DL begin planning for long-term adoption of Hyrax. This would start with short-term hires: a technical project manager and additional developers to take over day-to-day activities of the DL so that the permanent DL developers can get up to speed on the Hyrax system and integrate with the Hydra/Samvera community. Spin up an instance of Hyrax for testing and exploration in order to make informed decisions and recommendations on how to proceed with full-scale adoption of Hyrax. The initial focus of UCLA developers should be aimed at learning the Samvera system, learning the idiosyncrasies of the language and underlying platform (RAILS), and developing strong devops practices that will help them adapt and succeed in this new environment. The DL can also begin experimenting with Blacklight as the front-end for at least one project.

These recommendations are based on the following rationales:

- The concentration of adoption of Samvera/Hyrax on the West Coast and in the UC system should provide the supportive, collaborative environment needed and will increase UCLA's connections to peer institutions.
- Of the systems available at the time of the report, Hyrax appears to offer the most functionality out of the gate than the other available options. This solution will likely best meet the UCLA DL needs going forward.
- The modularity and extensibility of Hyrax will allow the DL to develop a more robust DAMS and DL stack over the years, especially give the community and collaborative environment. We may be able to build a custom system that meets UCLA-specific needs in a short period of time, but over time the lack of community engagement and shared development will leave us struggling to implement new technologies and features with our isolated system.
- There will certainly be a learning curve, but ultimately, making a decision and following through, despite difficulties, should pay off in the form of a robust digital library system and new knowledge.

Some further considerations

Functional and scalability evaluation

The recommended course of action involves setting up a test instance of Hyrax. In addition to test-driving its features and workflows, it will be essential to do a thorough evaluation of its scalability within the context of our anticipated use cases and functional requirements as outlined in Appendix 1 (this advice really applies to any system we might adopt or build on our own). Scalability tests of the Fedora repository and Hyrax "heads" may include the following:

- **Ingest/management:** Can Hyrax ingest collections on the scale of Frontera or all of the IDEP materials (i.e., hundreds of thousands of records) in a reasonable amount of time, and subsequently manage them efficiently?
- **Multimedia:** If large multimedia objects (music and video) cannot be stored in Fedora effectively, are the alternatives (e.g., file system storage with links from metadata) workable?
- **Search/browse:** Can several large collections all be stored in a single Solr index in a way that makes searching, browsing, and management of the index itself tractable? If not, how workable are the alternatives (e.g., maintaining multiple Solr "cores")?

Publishing/UI options

It is important to keep in mind the potential publishing, access and search layers that the DAMS systems described above would interact with. Some of the technologies involved:

- Solr (powers the search/browse features of Blacklight, among others)
- Python and Python frameworks (Django)
- AngularJS (and Angular2), other Javascript-based interfaces, e.g., React
- Blacklight (discovery layer framework built on Rails, with plugins for exhibits, archival collections, and geospatial data discovery, etc.)
- IIIF server and viewer (such as Mirador, Loris, Universal Viewer)
- Media servers
- Other?

Linked-data/data API strategy and technologies

- Fedora RDF triples -- Should we expose these? If so, how to expose these? (no one has really worked this out yet; most Samvera adopters don't see full semantic web integration as feasible, though it can be done for specific collections)
- Rails microdata -- gems can automatically expose a great deal of collection data via a built-in URL-based API
- Interaction with automated authority sources
- Metadata- and resource-syncing technologies: OAI-PMH, ResourceSync/sitemaps
- IIIF (standards and APIs for sharing images)

Storage/preservation integration

We assume a workflow in which the digital library DAMS is the main point of entry and management for new digital items, and these items eventually make their way to the preservation layer after ingest into the DAMS. There are, however, alternative workflows in which items could move from the preservation layer up into the DAMS via software ecosystems like the ArchivesSpace/Archivematica suite (though these tools also support top-down ingest procedures).[9] See for example the University of Michigan's Bentley Historical Library's ArchivesSpace->Archivematica->DSpace workflow (http://journal.code4lib.org/articles/12105).

In any case, it is worth noting the main DAMS/preservation paradigms we observed at peer institutions: although few of these systems were fully automated, several had instituted policies by which preservation items were created as soon as an item

[9] Allain, Sara. "Archivematica in the Middle", https://sallain.github.io/c4l17-archivematica/#/

was ingested into the DAMS (though the frequency of subsequent “syncs” could be quite variable). Later edits of these items (which on the whole is quite rare) would trigger the creation of a new, replacement item in the preservation layer.

User Stories / Functional Requirements

User Stories

Digital Library Collections (and Digital Collections Lab)

*Narrative: The **Digital Library Program** collects and maintains collections of digitized content on behalf of the University Library (the **UCLA Digital Collections**), the UCLA Community, and for special and extramurally-funded projects (such as NewsScape, IDEP, Sinai Palimpsests, and Frontera. These collections are often accessible to the world without authentication. In many cases, these collections are accessible only to the UCLA community. Some content may have download restrictions (downloads not permitted or downloads restricted to derivatives only). The assets are in various formats including but not limited to TIFF, JPG, WAV, MP4, MP3, TXT, PDF, XML) in addition to their metadata. The proposed **Digital Collections Lab** will provide access to the Digital Library Collections via downloadable CSV files, APIs, data endpoints, programming interfaces, and crowdsourcing environments to facilitate scholarly use of these collections and their data.*

Story: As a DLP Staff member, I want to…

- bulk upload assets and metadata in a various data formats (see above)
- automatically generate/assign unique identifiers/handles (ARKs) on upload of assets
- edit permissions on collections, items and files
- assign administrative permissions for read/write/delete access on collections, items, and files
- edit metadata on collections, items, and files
- search for collections, items, and files
- limit searching by tags
- attach tags to collections, items, and files
- limit searches by collections, items, and files
- Identify and view collections, items, and files based on their workflow status (no metadata, metadata in progress, metadata complete; pre-/post-quality control, etc.)
- publish and unpublish collections, items, and files individually
- publish and unpublish collections in bulk
- leverage professional looking end-user search interfaces to my DLP collections, items and files; including embedding search tools and results pages into other web publishing interfaces (e.g. Blacklight, Spotlight, Mirador, Django, etc.)

- capture and present the organizational structure of complex objects (books, photos with front/back, archival box/folder, serials, etc.) in our user-interfaces using metadata
- embed links to published collections, items and files inside of other web publishing interfaces (e.g. blogs, CMS pages, CCLE...)
- utilize technologies for sharing my collections widely and openly, such as IIIF manifests, ResourceSync, OAI-PMH, etc.
- make use of APIs, IIIF, ResourceSync, etc. for republishing Digital Library Collections materials in new interfaces and platforms that allow for transcription, annotation, use by third-party applications, etc.
- delete collections, items, and files for which we no longer have retention or other rights
- provide a mechanism for DLP partners (Special Collections, IDEP, CSRC, etc.) to enter assets and metadata into the DAMS
- mask (hide from view based on permissions) collections, items, and files as necessary, either manually or on a schedule (embargos)
- store and represent different versions of files which can change as a result of corrections or additional data
- have library data & metadata processors (i.e. metadata librarians, approved curators, assistants, and partners) process collections for use by end-users
- leverage internal and external controlled vocabularies when entering authoritative terms into the metadata record
- export metadata from the DAMS for internal use/requests in various formats including CSV, JSON, …
- deposit collection-level “bags” containing assets and metadata into preservation layer from DAMS or repository (as needed or triggered by actions or state)
- query across collections and projects and take advantage of inter-collection linking and indexing, either via semantic technologies or a unified search schema

Story: As a Subject Librarian or Curator, I want to:

- download files or groups of files from the DAMS to fulfil patron requests (as masters or derivatives would be nice if I can get it)
- batch edit metadata for collections, items, and files for which I have permissions
- bulk upload assets and metadata in various formats to collections for which I have permissions
- search for/browse digital assets in the DAMS by keyword or specific field search and by browsing collection hierarchies
- create exhibits or “spotlight collections” using items already in the Digital Library Collections

Story: As a DLP Partner, I want to:

- batch edit metadata for collections, items, and files for which I have permissions
- bulk upload assets and metadata in various formats to collections for which I have permissions
- search for/browse digital assets in the DAMS by keyword or specific field search and by browsing collection hierarchies
- create exhibits or “spotlight collections” using items already in the Digital Library Collections

Story: As an end user, I want to…

- access a search page that returns faceted results of the collections, items, and files
    - facet should limit by subject, tags, keywords, dates, collections, format, language, geographic, etc.
- browse the collections, items, and files via multiple views (list, gallery, map, network graph, archival arrangement, etc.)
- be prompted for UCLA Shibboleth permissions at the point of need
- download collections, items, and files for which I have permissions
- browse and search the metadata and descriptions for items for which I do not have download permissions (and be properly notified of my privileges before attempting to download files)
- embed Digital Library Collections content for which I have permissions into web pages and other media
- have the option to view book objects in a book reader format; have the option to view archival collections by box and folder

Story: As a scholar, I want to…

- access Digital Library Collections assets and data via various options including:
    - APIs and data endpoints for use in computational research methods and in scholar-led thematic research collections or projects
    - downloadable files (csv, json) files (most researchers prefer to download data as csv over using APIs and other data endpoints)
- access and/or download textual content (OCR and text files) from the Digital Library Collections for text mining and analysis
- create exhibits or “spotlight collections” using items already in the Digital Library Collections
- make use of IIIF manifests to gather together high resolution images of certain collections and items (ex. books and manuscripts) from the Digital Library Collections and other institutions into my own Mirador (of other viewer) instance

and publish my own manifests and annotations to share with others, use in my classes, etc.

Story: As an instructor, I want to…

- create exhibits or "spotlight collections" using items already in the Digital Library Collections
- embed Digital Library Collections content for which I have permissions into web pages and other media
- have access to training packages (datasets with learning modules for collections as data)
- make use of IIIF manifests to gather together high resolution images of certain collections and items (ex. books and manuscripts) from the Digital Library Collections and other institutions into my own Mirador (of other viewer) instance and publish my own manifests and annotations to share with others, use in my classes, etc.

Digital Stacks

Narrative: Subject librarians and Curators often acquire published digital content (ebooks, audio files, excel worksheets) that are restricted by copyright, but are not hosted by vendors. This content tends to not fit the scope of the Digital Library Collections and has specific access and restriction needs. The Digital Stacks will provide librarians and curators a place to upload and host this content and set access and download permissions as needed. Items would be discoverable via the Library OPAC.

Story: As a Subject Librarian or Curator, I want to:

- provide online access to curated digital content in various formats (PDF, MP3, MP4, Excel)
- upload content and metadata without remediation from the Digital Library Program
- restrict access to UCLA-only when appropriate
- restrict download of items for all users when appropriate (including PDFs)

Story: As an end user, I want to:

- find UCLA-hosted online content via the UCLA library catalog and access this content via a link in the catalog
- view / read items for which I have permissions
- download items for which I have permissions
- be prompted for UCLA Shibboleth permissions at the point of need

A/V Preservation (still in progress)

Narrative: The A/V Preservation unit digitizes audio and moving image materials from the UCLA collections. These collections are often accessible to the world without authentication. In many cases, these collections are accessible only to the UCLA community. When digitizing moving image, lesser quality derivatives are often created for streaming on public interfaces while the larger, high quality preservation master files are stored as an archival copy and not publicly accessible. These are currently stored on our NetApp shared storage.

Story: As an A/V Preservation Librarian, I want to:

- store, describe, and access master versions of audio and video files alongside their published derivatives
- record technical and digital provenance metadata for audio and video files (capture, processing and QC metadata, etc.)
- deposit collection-level archival "bags" containing masters versions and metadata into preservation layer from DAMS or repository (as needed or triggered by actions or state)
- Automatically generate intermediate and access derivative files from deposited preservation master according to our standard specifications.
- Automatically generate MD5 checksums for files upon deposit into DAMS or repository. These checksums will be used as a baseline for future fixity or "health checks" of the files.
- Automatically embed metadata into the files such as the Federal Agencies Digital Guidelines Initiative (FADGI) Specification of the Broadcast Wave Format Version 2 (2011).

Requirements

Architecture

- Must be highly scalable, open, and flexible. System must scale to accommodate:
    - 2 million (and growing) items
    - high usage levels from both inside and outside the University
    - high ingest volume within a reasonable timeframe (system load)
    - high volume of simultaneous users (do we have any stats on this?),
    - users harvesting our collections via APIs and other tools (need to moderate that or partition?)
- Support for a multitude of formats (or content agnostic) - need to be able to serve up different formats
- Expose as much data as possible via RESTful or the like
- Support our existing rich data model
- Infrastructure must not limit access to particular types of workstations, operating systems, or creative software
- Support the translation of digital assets into a wide variety of formats

Checks/preservation (these are just examples - I don't really know)

- Fixity:
    - Fixity checks are executed daily on a limited number of repository objects. For each object a "fixity check event" is recorded in the database (not the repository) with the results of checksum validations of all its current data streams (latest versions only).
    - Fixity should be checked on each object at 60-day intervals.
- Checksum guidelines
- Include PREMIS interoperability
- Audits (needs review with experts)
    - What questions do you want to be able to answer from the audit trail?
        - Object version history
            - Agent that created the version (user or system)
            - Optional (?) comment on change (could be stored in event_detail)
        - Actions performed on object
            - Ingest
            - Derivative generation
            - Fixity calculated
            - Fixity checked
            - object versioned
            - version restored
            - object withdrawn

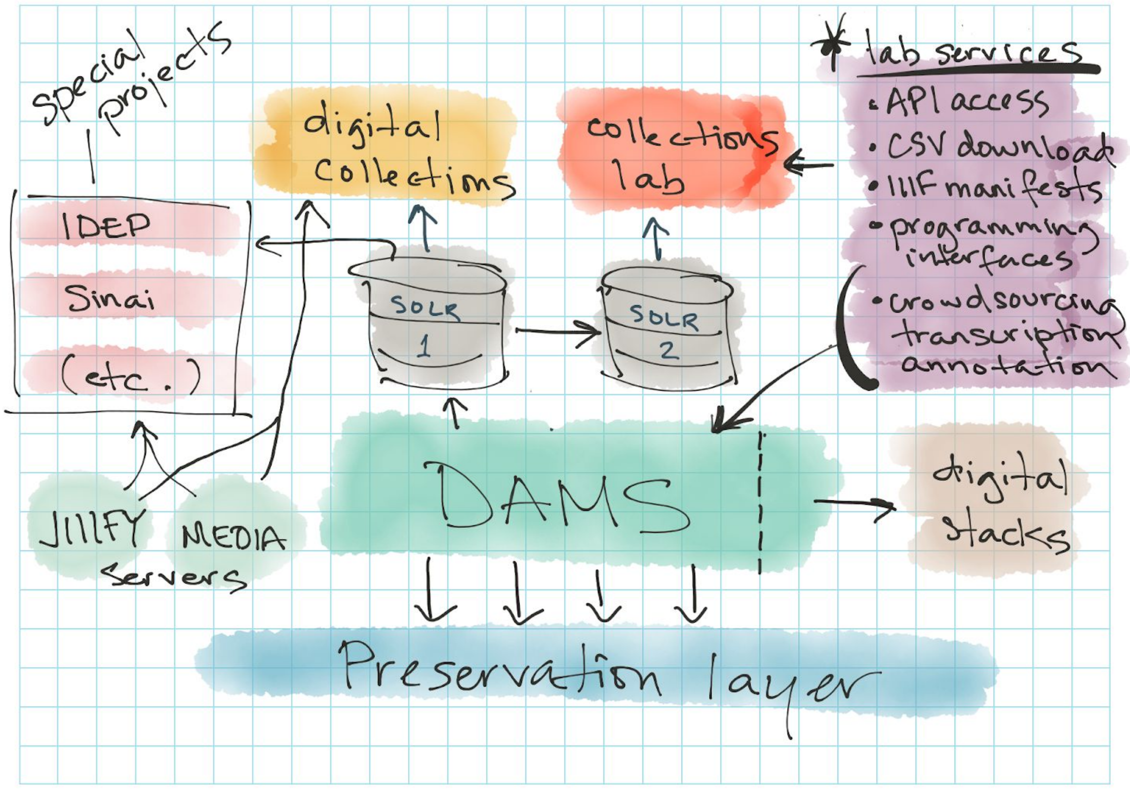
    - object purged
- Deposit to preservation layer: pointer to preservation logs, track format migrations, link to or record preservation/conservation documentation

Appendices

Appendix 1: User Stories / Functional Requirements

Available here:
https://docs.google.com/document/d/1YShZiTCMBViFhvT5aeig88UZuNn2ERdkewOLzBtJS_E/edit?usp=sharing

Appendix 2: DL Services View Diagram (work in progress)



Appendix 3: Glossary of terms

Note: some were shamelessly plagiarized from the Nov. 2016 UC Libraries DAMS Technology Report and the LDCX 2017 terms, abbreviations and acronyms glossary

ArcLight - an in-development implementation of Blacklight serving the discovery and delivery of archival description

ArchivesSpace - an open source archives management platform and community affiliated with Lyrasis. The successor tool to Archivists' Toolkit (AT) and ArchOn, which were competitor systems to serve the archives community in processing and description workflows.

Avalon - a Hydra-based application for delivering and managing time-based media (audiovisual materials). (A Hyrax proof-of-concept version appears to be in development)

Blacklight - a Ruby-on-Rails engine for retrieving and searching Solr content (in addition to Rails's RDBMS data)

Complex Objects: Common in digital library projects, complex objects consist of many files and may have a complex hierarchy of relationships between component parts; for example, a photo with front and back content or a book object with pages.

DAMS: Digital Asset Management System, for managing digital files and related metadata [could use further elaboration]

Duraspace: Umbrella 501(3)(c) organization founded in 2009 that works with Fedora, Samvera Project, and other open source software projects.

Export, Transform, and Load (ETL): Process for bulk loading metadata from a local system into a DAMS.

Fedora Repository: a backend component for digital object repository systems developed by DuraSpace affiliates, supporting repository services such as fixity, versioning, transactions, import, and export. Modular, open source, and now with native linked data support.

GeoBlacklight - a Blacklight engine for providing geodata interfaces for discovery

Samvera (Hydra) Project: Open source repository solution and community, built on top of Fedora using Ruby on Rails, as well as Solr/Blacklight. Focuses on front-ends and middleware. There are currently two viable Samvera solutions: Hyrax and Hyku. Hyrax could be considered an alternative to Islandora.

Hydra Head: An application built using Samvera technology that takes advantage of the underlying repository layer (Fedora) and other Samvera middleware.

Hydra-in-a-Box: See "Hyku"

Hyku (Hydra-in-a Box): A project aimed at making it easier to adopt a Samvera based solution for a repository. A Samvera app, based on Hyrax, meant to support multiple tenants (one for research data, one for ETDs, one for library collections). The plan is to offer a "Hyku Direct" hosted solution for smaller institutions, in addition to the standalone version. This is more of a "blackbox" solution.

Hyrax: A Samvera application or "head" that merges and enhances two earlier Hydra heads, Sufia and CurationConcerns. It is the foundation of Hyku. Hyrax is meant to be more modular and configurable, supporting features such as ResourceSync integration.

Islandora: An open source repository solution and community, built on top of Fedora using Drupal/PHP. The current "1.0" version of the stack uses Fedora 3 and Drupal 7, while the forthcoming Islandora "CLAW" will feature Fedora 4, Drupal 8, and a suite of new middleware services.

Linked Data Platform (LDP): Standardized architecture for creating linked data applications used in version 4 of the Fedora Repository.

Linked Data: An approach to data modeling that uses the Resource Description Framework (RDF) of the Semantic Web.

Lyrasis - non-profit library consortium, providing hosted software services for deployments of ArchivesSpace and Islandora

METS: Metadata Encoding and Transmission Standard, originally based on work by UCB; XML standard that can be used to manage complex objects.

Nuxeo: Enterprise content management platform, open source with commercial support.

Portland Common Data Model (PCDM): Based on work originally undertaken by UCSD, the PCDM is a linked data standard that can be used to manage complex objects. Samvera Project and Islandora are working together on PCDM, with the aim of interoperability between systems using Fedora.

Repository: For the purposes of this study, this term is generally used interchangeably with a DAMS.

ResourceSync - a syndication and synchronization specification supporting the discovery of web resources

Appendix 4: Focus group and interview questions

Technical leads and stakeholders at peer institutions

Understand current work and capacity

- Who at your campus is actively participating in digital content/collection creation? ex. data curators, digital librarians, tech services, metadata librarians, special collections, IT staff.
- How many assets are in your digital collections in total? Do you have a breakdown by type, topic (collecting area), etc.?

Workflow

- Describe your current workflow (selection, ingest, metadata, access, preservation, etc.)
- What kind of content do you primarily work with?
- What is the most complex digital asset type you work with? Describe your most complex collection.
- Do assets always belong to collections, or are there one-off assets?
- Describe the type of hierarchy used to organize digital assets.
- What's your workflow for creating metadata, and loading and staging files associated with the metadata?

DL Systems

- How would you define the terms we throw around to talk about various components of DL systems (see list on parent page)?
- What do **you** think the components of a DL system are and which are the most important?

- Can you tell us about where the DL stack fits into/overlaps with other systems? Is there shared infrastructure? Other services like data repo, IR, etc.?

Management - probably most important for us

- What kind of system are you using to create and manage your digital assets, and what features are you using?
- How are you utilizing and managing authority records and controlled vocabularies, within the context of your system?
- Are you managing end-user contributed content in your system?
- What kinds of export formats can you generate (or need to) for your collection?
- How are you preserving your digital content?

Extended features

- Are you able to provide metadata for harvesting (OAI-PMH), and, if so, what kind? What about LOD?
- Does your system support bulk editing and uploading of item records, and how? If so, who has the ability to do it?
- Can items belong to multiple collections within your system?
- Do you intend to expose/export some collection items for access via other systems (at other places?) if so, how?
- What is your total digital asset storage requirement now and in 5 years? Are there any plans for the future for your digital library resources that we should know about?

End user needs

- Does your system support multi-language collections and/or interfaces?
- Does your system support end users maintaining “personal collections”? such as bookbag or a shopping cart?
- Tell us about your search and browse functionality

DL developers, administrators, and stakeholders at UCLA

Understand current work and capacity

- Who do you know who is actively participating in digital content/collection creation on campus (in case we've missed anyone)? ex. data curators, digital librarians, tech services, metadata librarians, special collections, IT staff.

Experience using DL systems

- Which DL systems do you work with (DLCS, Islandora, etc.)? What do you wish we could do with the collections that the system doesn't provide?
- What types of materials/content do you wish we could do more with?
- Are the metadata/object description features we have now sufficient? If not, what would you like to add?
- How would you define the terms we throw around to talk about various components of DL systems (see list on parent page)?
- What do **you** think the components of a DL system are and which are the most important?

Management

- What digital asset management features do you think a DL system should provide? What component of the stack should provide this? Can you describe your desired architecture or an implemented example of it?
- Do you want to be able to manage end-user contributed content in our system?
- What kinds of export features and formats should we support?
- How should the DL system fit into the digital preservation question?

Extended features

- What kinds of metadata harvesting services (OAI-PMH) should we provide? How about LOD – and where should this go in the stack?
- Does should a DL system support bulk editing and uploading of item records, and how? If so, who has the ability to do it?
- Should items belong to multiple collections within your system?
- How should we expose/export collection items for access via other systems (at other places?) and how do you expect to see this practice expand in the future?
- What will be your total digital asset storage requirement now and in 5 years? Are there any plans for the future for your digital library resources that we should know about?

End user needs

- How should a DL system support multi-language collections/interfaces?
- Should we support end users maintaining “personal collections”? such as bookbag or a shopping cart? If so, which DL system lets us do this?
- Search and browse functionality – tell us what you think!