ArchivesSpace Technical Design Meeting, June 3-4, 2010

Summary Report on Technical Architecture

Prepared June 2010

by Mark A. Matienzo, Technical Architecture Consultant

0 Executive summary

Overview of meeting and report

The ArchivesSpace project team hosted a technical planning meeting at New York University to seek advice on the technical architecture for the proposed development of a next-generation archival management system based on the logical merge of Archivists' Toolkit and Archon. Attendees included a group of invited systems architects and technical managers with expertise in creating, deploying, and maintaining software for libraries, archives, museums, and educational institutions. The meeting was organized into sets of breakout sessions to discuss specific architectural areas. Once the attendees reviewed the description of the architectural area for a given area, they nominated candidate technical solutions for that area. Following each set of sessions, the attendees gathered in a plenary session to discuss the candidate solutions. Finally, the attendees were given time to provide additional comments and advice to the project team for the planning of the development for the merged application. This report summarizes these discussions, lists identified candidate solutions, and specifies recommendations identified by the technical architecture consultant and the meeting attendees. The report also provides discussion of the attendees' remarks, and links to additional resources.

Fundamental architecture

The merged application should be a web application, built using principles of service-oriented architecture (SOA) and Representational State Transfer (REST). Service-oriented architecture is a set of design principles wherein an application's business logic or individual functions are modularized and loosely coupled as "services" that can be accessed or consumed by client components. Representational State Transfer is an architectural style for web applications that defines a set of constraints for a system's architecture based on principles underlying the design of the World Wide Web. These architectural recommendations were chosen over both building a "thick" desktop client application (comparable to the Archivists' Toolkit) or building a web application using SOA with a different web services layer, such as SOAP. A clearly defined service layer allows for better separation of concerns and greater flexibility in terms of other implementation details, such as the application's persistence layer or end-user interface.

Project management and development methodology

The development team should choose a development methodology as early as possible, and should consider using an agile software development methodology, such as Scrum. One attendee noted that if a non-agile methodology was chosen, the project team should have business reasons that can be articulated to stakeholders. Development of core application architecture may be difficult in a distributed setting, and attendees therefore urged the development team to consider doing in-person code sprints for architecture, if possible. Integration of the application's stack may become more complicated as development progresses, and accordingly the development team should perform this integration as early as possible. Finally, establishing a clear testing methodology for the application is very important.

Community development and involvement

Development of a community around the application and providing opportunities for participation should be prioritized. Suggestions included the creation of a regular schedule for the ArchivesSpace team to issue announcements about the project. Several attendees specified that the development process needed to be transparent; means to achieve this included an active and open e-mail discussion list for developers, use of wiki software such as Confluence, and issue tracking software such as JIRA. That the project team should create low-barrier and low-commitment opportunities for community members to participate in the development. Providing seemingly informal opportunities for participation in the application's development may allow those individuals to have a greater sense of ownership in the merged application.

Integration concerns

The development team should consider specifying the ways in which the application would need to integrate with other systems, such as digital repository software like Fedora and DSpace, as well as identifier services such as ARK and the Handle System. In addition, given the existing body of independent development by the Archivists' Toolkit community, the application should be extensible and provide a command-line interface wrapper to the application's service layer.

Multi-tenancy

Multi-tenancy software architecture is a method that allows a single instance of an application to serve multiple customers. A multi-tenant deployment can be desirable solution for organizations or service providers who would otherwise need to maintain several instances of a single-tenant application for a given base of clients. It provides better scalability for service providers since the application's architectural components can be scaled based on the needs of individual tenants. Given that multi-tenant deployment was identified as a potential goal in the AT/Archon Hi-Level Requirements, the development team should consider committing to accommodating multi-tenancy from the beginning of the development process. However, attendees also advised the ArchivesSpace team that multi-tenant development was complicated. In particular, the team should consider hiring a software architect that has done multi-tenant development before, preferably multiple times.

Licensing

Potential nominees for licensing the application include the Apache Software License (ASL), version 2.0, and the Educational Community License (ECL), version 2.0, which is a variant of the Apache Software License. Several attendees suggested avoiding viral licenses, such as the GNU Public License (GPL). Licensing will most likely have an impact on how the application is distributed, particularly in terms of the ability of the ArchivesSpace team to create a packaged, installer-based version of the merged application. The development team may want to consider explicitly excluding dependencies that are under viral licenses, which is the approach taken by the Sakai project.

1 Background on technical planning meeting and report

1.1 Overview

On June 3rd and 4th, 2010, the ArchivesSpace project hosted a technical planning meeting at New York University. The meeting's attendees included project staff and a group of systems architects and technical managers (see Addendum 3.1). The goal of this meeting was to create a technical development strategy and a set of architectural recommendations for a proposed application resulting from a logical merge of Archivists' Toolkit and Archon. Before the meeting, the attendees were asked to review a set of documentation, including the Archivists' Toolkit/ Archon Hi-Level Requirements compiled in January, 2010, a set of preliminary specifications for data models for the merged application, and a background document (see Addendum 3.2) prepared by Mark A. Matienzo, a consultant hired by the ArchivesSpace management team to assist with the creation of the project's technical development strategy. This document included background information on the archival domain and Encoded Archival Description; the data model and functional "modules" of both Archivists' Toolkit and Archon; an overview of the potential deployment requirements for the merged application; and background information and analysis on areas of the application's architecture. The identified architectural areas were as follows:

- Authentication
- Authorization
- Storage/persistence layer
- Import and export
- User interface
- Discovery
- Reporting
- Workflow management

These identified areas served as the primary areas for discussion during the meeting (see "Meeting structure" below).

1.2 Meeting structure

The structure of the first day of the technical planning meeting was mostly focused around two sets of breakout sessions. Within each set of sessions, there were three individual sessions. For each session, the attendees signed up for one of four architectural areas (see above list) scheduled for that set. The attendees within an individual session selected a member to take notes and represent the group during the plenary discussions. Each session's attendees also had the option to review the notes of previous sessions on that topic or to start over with a "clean slate." Once the attendees reviewed the description of the architectural area for that session, they were asked to nominate candidate solutions.

The attendees were asked to consider the framing principles and functional requirements for the merged application, especially in terms of the interoperability needs and deployment scenarios identified by the ArchivesSpace team. In addition, attendees were asked to identify pros, cons, and "show stoppers" for each of the candidate solutions whenever possible. Following each set of sessions, the attendees gathered in a plenary session to present their notes from the sessions and to identify particular candidate solutions. The Technical Architecture Consultant prepared an exhaustive list of all the candidate solutions derived from notes taken during the individual breakouts and plenary sessions.

The second day of the meeting consisted of reviewing the list of candidate solutions prepared from the first day's notes and discussion and allowing attendees to make further nominations for candidate solutions that had not been specified previously. Additionally, attendees had the opportunity to reiterate any strengths or weaknesses for these individual candidate technologies. Finally, the attendees were given time to provide any additional comments or advice to the project team for the planning of the development for the merged application.

1.3 Report structure

The remainder of this report presents results from the meeting split into sections based upon topic or architectural area. Some sections, such as "Import and export" and "Discovery," were subdivided into smaller units based upon subsets of discussions that concerned an aspect of these areas. Most sections are broken down into four subsections: potential candidates, recommendation, discussion, and resources. Most of the potential candidates were identified during the meeting (see "Meeting structure" above). The recommendation subsections list the best option or set thereof that were identified by the attendees and the Technical Architecture Consultant. However, in some cases, no recommendation could be identified; these cases are noted as such. The discussion subsections summarize the discussions between attendees and the ArchivesSpace team during the meeting and provide additional background or context to assist the report's readers. The resources subsections provide references to additional material for readers. Unless the link is directly related to a specific recommendation, readers should not necessarily interpret this as an endorsement. If the link is not directly related to a recommendation, it should be recognized only as supporting material or a reference model for the particular architectural area.

2 Results

2.1 Fundamental architecture

Potential candidates

- Web application, built using principles of service-oriented architecture (SOA) and Representational State Transfer (REST)
- Web application, built using principles of service-oriented architecture and a different web services layer (e.g. SOAP)
- "Thick" desktop client that runs natively on end-user machines (e.g. the existing Archivists' Toolkit application)

Recommendation

- **Web application, built using principles of service-oriented architecture (SOA) and Representational State Transfer (REST)**

Discussion

The attendees were in general agreement that the merged application should be web-based and built using principles of service-oriented architecture (SOA), a set of design principles that allow for a loose coupling of services combined to provide an application's functionality. Some attendees also emphasized that a clear definition of the application's service layer would allow for greater flexibility in terms of other implementation details, such as the application's persistence layer or end-user interface. In addition, the attendees also expressed support for the application's services to be defined using constraints expressed within Representational State Transfer (REST). REST's constraints include client-server separation, server-side statelessness, providing cacheable responses, allowing for layering of systems to ensure scalability, and providing a uniform client-server interface using a set of additional constraints. With a "RESTful" architecture under consideration, a few attendees also suggested that the project's technical team should consider investigating the use of resource-oriented architecture (ROA) to allow for full create, read, update, and delete access to any "resource" defined within the application.

Resources

- Wikipedia: Service-oriented architecture
- Talis: *Opening the Walls of the Library: SOA and Web Services at Talis*
- Roy Fielding: "Representational State Transfer" (chapter from dissertation)
- Roberto Lucchi, Michel Millot, and Christian Elfers: *Resource-Oriented Architecture and REST: Assessment of Impact and Advantages on INSPIRE*
- Brian Sletten: *Resource-Oriented Architecture: The Rest of REST*

- CollectionSpace: Common Services REST API documentation
- Introduction to the Atom Publishing Protocol

2.2 Programming language and application frameworks

Discussion

During the meeting, there was very little direct discussion of particular programming languages application frameworks used to build the merged application. Choice of a particular framework may assume that developers will be committed to developing the application using only one programming language. Particular languages were neither explicitly nominated nor specified as a requirement during the meeting, but much of the discussion in other architectural areas focused on implementations specific to Java. Nonetheless, the development team is not necessarily committed to choosing only one language given the flexibility provided in building the application on the principles of REST and service-oriented architecture. One attendee noted that the development team may want to consider the possibility of implementing the service layer and the application layer using two separate languages.

The Technical Architecture Consultant and the ArchivesSpace management team also believed that specific application frameworks could not be considered until attendees could make clear recommendations in other architectural areas. Additionally, their consideration depends on how "frameworks" are defined. In one sense, frameworks could include web application frameworks, which support the development of dynamic web sites, web applications, and web services. Web application frameworks are tied to a specific programming language as indicated above. Example web frameworks include Ruby on Rails, Django (for Python), Symfony (for PHP), Drupal (a PHP content management system), and Struts (for Java). Frameworks could also be defined more specifically to refer to an established architectural platform, within the archival domain or from a related domain such as the digital library community. This type of framework could include Fedora, CollectionSpace, or the Qubit Toolkit. The ArchivesSpace team should review any candidates carefully against both the requirements for the merged application and the recommendations within this report.

Resources

- Wikipedia: Web application framework
- Ruby on Rails
- Django
- Drupal
- Symfony
- Apache Struts
- Fedora
- CollectionSpace
- Qubit Toolkit

2.3 Persistence/storage layer

Potential candidates

- Relational database
- Object database
- XML database
- "NoSQL" non-relational database (usually document-oriented; often schemaless; examples include CouchDB, MongoDB, and Cassandra)
- Solr
- Triple store or graph database (usually presumes data will be modeled using the Resource Description Framework)
- Java Content Repository/Content Management Interoperability Services
- Local and/or network file system

Recommendation

- **No clear recommendation overall or specifically for metadata persistence; see discussion below**
- **Local and/or network file system and Java Content Repository/Content Management Interoperability Services for digital object persistence**

Discussion

Attendees at the technical planning meeting identified a variety of candidate solutions to serve as the potential storage and persistence layer for the merged application. However, there was some discussion about possibly separating the persistence layers for metadata and files that comprise digital objects. However, this separation was not initially established as a requirement, nor was it included within the background readings and discussion of architectural areas. Accordingly, the ArchivesSpace team explicitly recommended considering distinct technologies for each of these two areas to allow for better separation of concerns. While none of the individual candidate technologies were completely removed from consideration, some attendees expressed concern that certain technologies for persistence were simply too "bleeding edge" to be considered appropriate. These concerns, which focused mostly around using either "NoSQL" databases, Solr, or triple stores as the primary persistence layer, included consideration of finding qualified developers, leveraging community development assistance, and deployment by institutions with strict information technology policies.

Despite the lack of consensus around a single technology for persistence of metadata and other application data, the meeting attendees discussed various aspects relating to the potential use of a relational database as a persistence layer. The meeting's conveners asked if the merged application would require a database abstraction layer and/or an object relational mapping (ORM) layer if a relational database-backed persistence layer was chosen. Most attendees

seemed to concur with the decision to use a database abstraction layer, but to favor MySQL as the focus of core development. Currently, Archon supports MySQL, Microsoft SQL Server (MSSQL), and Archivists' Toolkit supports MySQL, MSSQL, and Oracle. Despite this, much of the install base for both applications relies on MySQL. By comparison, only two or three institutions that have implemented Archivists' Toolkit are using MSSQL; only one AT institution is using Oracle; and approximately 30% of the Archon deployments are using MSSQL. If there were sufficient community interest in testing or developing the database abstraction layer for other platforms, the ArchivesSpace team could either reassign resources or assist the community in creating a development plan as appropriate. Furthermore, some programming languages or application frameworks may make use of established database abstraction layers that support multiple relational database management systems.

In terms of choosing an ORM, several attendees made repeated comments to avoid the use of Hibernate, which is the ORM layer currently used by Archivists' Toolkit. Most attendees seemed to agree with these comments, and those familiar with the Toolkit stated that the application's reliance on Hibernate is one of the major barriers to the user community being able to contribute code. Additionally, several attendees noted that choice of a specific database abstraction layer or ORM will be partially dictated by the choice of programming language for the merged application. Kuali Rice's persistence layer, which uses the Apache Object/Relational Bridge as an ORM layer and the Java Persistence API, was nominated as a potential candidate if a relational database-backed persistence layer was chosen.

In addition, the ArchivesSpace team asked the attendees to consider persistence solutions for digital objects, given that Archon currently provides a mechanism for storage of digital objects as part of its functionality. The meeting's conveners scoped the discussion by stating that the merged application was not designed to act as a digital preservation system or as a full-featured digital asset management system. In terms of providing storage, two solutions were identified. The first solution was the use of a simple local or network file system, which could be presented to the application as a directory containing the digital objects. This was identified as the baseline requirement for the functionality. Additionally, several attendees nominated the use of either a persistence mechanism that complied with either the Java Content Repository specification or the Content Management Interoperability Services specification. These specifications act as abstraction layers over enterprise content management systems and ensure a high-level of interoperability. Using these specifications would allow the application to be packaged with a particular content management layer and also allow implementers to substitute another according to institutional preference or policy.

Resources

- Wikipedia: Object-relational mapping
- Kuali Rice: JPA Conversion Guide
- JSR 283: Content Repository for JavaTM Technology API Version 2.0
- OASIS Content Management Interoperability Services Technical Committee

2.4 Import/export

2.4.1 Overview

Discussion

The general consensus of the meeting attendees was that planning the architecture related to import functionality was considerably more complicated than that of any aspect of export functionality. Accordingly, much of the discussion in the breakout sessions focused on aspects related to import functionality. In particular, a few attendees articulated a clear difference between specific import "formats," or schemas for data to be imported into the application, and import "containers," or serializations of the data transmitted to the application for import. The attendees also recognized a clear need for a mapping layer that would allow translation between import formats, internal representations of the data within the application, and export formats.

2.4.2 Import formats

Potential candidates

- Exclusive use of community-based schemas, such as MARCXML and EAD
- Exclusive use of a use of ArchivesSpace-specific schemas
- Use of community-based schemas where available/applicable, with a defined application profile containing constraints, extensions, and supplemental ArchivesSpace schemas as appropriate

Recommendation

- **Use of community-based schemas where available/applicable, with a defined application profile containing constraints, extensions, and supplemental ArchivesSpace schemas as appropriate**

Discussion

The meeting attendees were not asked to evaluate individual schemas for inclusion as import formats given that baseline functionality was previously identified as part of the AT/Archon Hi-Level Requirements. Instead, attendees discussed the extent to which community-based schemas or application-specific schemas would be appropriate for the application. While reusing existing community-based schemas is a desirable option, the largest disadvantage in terms of relying on them exclusively is that there are some areas of the application's functionality that do not have a corresponding established schema. This includes both configuration-related data as well as certain kinds of data related to the archival domain, such as accession records. The Technical Architecture Consultant therefore recommends establishing a set of application profiles for the application, using a combination established schemas when available, and creating a set of

supplemental, application-specific schemas where no established schema exists. Additionally, the project team should consider establishing a set of documented constraints or extensions for the existing schemas to ensure a minimum of data loss upon import.

Resources

- Guidelines for Dublin Core Application Profiles

2.4.3 Import containers

Potential candidates

- XML
- Comma separated values (CSV)
- Binary

Recommendations

- **XML**
- **Comma separated values (CSV)**

Discussion

Attendees discussed potential "container" serializations used to import data into the merged application. The community-based schemas identified as required import functionality for the merged application are all serialized using XML. Archivists' Toolkit also supports import of comma separated values for certain types of application data. These two import container formats were thus identified as the primary recommendation for inclusion into the merged application. Nonetheless, for data in CSV, one attendee recommended the creation of clearly defined "profiles" that defined the order and structure of data to be imported. A few attendees also suggested the possibility of creating a binary serialization of application data to exchange between instances of the merged application. However, when presented with this option in the plenary discussion, this possibility did not gather significant support and was therefore not prioritized as a recommendation.

2.4.4 Mapping layer

Potential candidates

- Hard-coded mapping layer
- XSLT-based mapping
- Extract, Transform, and Load (ETL) tools from the data warehousing community

Recommendation

- **XSLT-based mapping**
- **Extract, Transform, and Load (ETL) tools from the data warehousing community**

Discussion

Currently, both Archivists' Toolkit and Archon have a hard-coded mapping layer for both importing and exporting data. The attendees recognized that this approach greatly limited the flexibility of the application and in some ways discouraged implementers from modifying the import or export mappings. A few attendees expressed a clear desire for a mechanism that would allow import and export mappings to be configurable by "non-programmers." Several attendees identified XSLT stylesheets as a possible implementation for the mapping layer. This technology was identified as a strong candidate solution particularly because it has a high level of adoption within the archival community as a means to transform EAD finding aids into HTML and other formats. Furthermore, there is anecdotal evidence that many individuals in the archival community that do not self-identify as programmers have some familiarity with XSLT.

Using XSLT as the mapping layer may also allow all imported data to be first mapped to an application-specific schema or a particular implementation profile of a community-based schema. Some attendees suggested that *all data*, even that from an established schema, be mapped to an application-specific schema that serves as the canonical representation of data within the merged application. A number of attendees also identified Extract, Transform and Load (ETL) tools as another potential, perhaps supplemental, implementation of the mapping layer. In particular, ETL tools were identified as a mechanism that may assist users of other collection management systems or tools (i.e., neither Archivists' Toolkit nor Archon) migrate data into the merged application.

Resources

- Carol Jean Godby, Jeffrey A. Young and Eric Childress: "A Repository of Metadata Crosswalks", D-Lib Magazine
- Wikipedia: Extract, transform, load

2.4.5 Additional considerations

Discussion

Beyond the subareas already identified, the attendees also discussed the issue of minimizing data loss during the import and export process. During any import process, there is a clear risk for losing data when its elements do not clearly map to fields or elements within the application's data model. One potential solution was storing a copy of the imported data in its original serialization format, such as storing a full MARCXML record after its elements are imported into the application's specified fields. A number of attendees also identified a few supplemental needs for consideration in terms of potential imports and exports, such as using the Simple Knowledge Organization System (SKOS) for controlled vocabularies, and identifying a schema to import and export rights statements. Finally, attendees also established that import and export functionality must include the migration both from Archivists' Toolkit and Archon to the merged application and between instances of the merged application.

Resources

- SKOS Simple Knowledge Organization System

2.5 Discovery

2.5.1 Overview

Discussion

Attendees at the technical planning meeting recognized that discovery could be discussed in a number of ways. The majority of the discussion on this topic was broken out into four areas: the overall architectural design of the application's discovery system, technologies for the indexing layer that allow the application's data to be searchable, specific search protocols that serve as an interface over the search index, and technologies that allow the application's data to be harvested by or syndicated to other systems. This is a particularly important consideration given that several attendees stated that discovery is likely to happen using interfaces external to the merged application, such as Internet search engines like Google and federated search systems used in academic libraries. These attendees thus stated that discovery systems designed for humans will not be the standard for access, but that harvesters and other machine-based agents would be. One attendee suggested that it may be helpful to consider the design of the discovery functionality of the merged application along two axes, with one as a continuum of design between human and machine-based agents, and the other being a continuum of optimization between search and browse.

2.5.2 Architectural design

Potential candidates

- RESTful mechanism for discovery, using best practices as defined as part of Web architecture

Recommendation

- **RESTful mechanism for discovery, using best practices as defined as part of Web architecture**

Discussion

This recommendation corresponds with the recommendation on the application's fundamental architecture, i.e., a web application, built using principles of service-oriented architecture (SOA) and Representational State Transfer (REST). Creating an overall RESTful application suggests, by extension, that the discovery architecture should also be created using RESTful principles. The discovery architecture could then also easily utilize essential parts of the World Wide Web's specifications. For example, the application could use content negotiation in any number of ways, including to provide representations of the application's resources in alternate languages, alternate output formats, or using an alternate set of templates that provide search engine optimized versions of those resources.

2.5.3 Indexing layer

Potential candidates

- Native indexing component of persistence layer
- Apache Lucene
- Apache Solr (relies upon Lucene)
- Extensible Text Framework (XTF, relies upon Lucene)

Recommendation

- **Apache Solr (relies upon Lucene)**
- **Extensible Text Framework (XTF, relies upon Lucene)**

Discussion

While both Archivists' Toolkit and Archon both use the native indexing component of the persistence layer, most attendees agreed that using this technology within the merged application would not be an ideal solution. This implementation could tie searching behavior directly into the application's logic, and thus serve as a further barrier for customization. In addition, the AT/

Archon Hi-Level Requirements identified adding a mechanism to allow for configurable indexing or search result weighting as a high priority need for the merged application. Several of the meeting's attendees identified using Apache Lucene as the basis as a key component of the indexing layer.

Given the relative complexity of using and configuring Lucene, some attendees indicated a clear preference for using Solr, a search and indexing server that provides a RESTful API for all operations. Solr is highly configurable, and it has a considerable level of adoption in the digital library community already as a general purpose indexing engine for metadata, including Encoded Archival Description. Furthermore, Solr serves as the back-end to Blacklight and VuFind, two open source discovery layers targeted at libraries. One attendee also suggested investigating the use of the indexing components of Extensible Text Framework, or XTF, an open source platform developed by the California Digital Library. XTF is also highly configurable and has a proven track record working with EAD.

In addition to discussing specific technical solutions, the attendees also discussed the difficulty of representing contextual information such as where in the hierarchy a specific keyword or phrase was located from a user's query. Unlike descriptive metadata for bibliographic resources and digital collections, archival description is extremely hierarchical and relies on principles of inheritance to provide context about the materials. Implementers of archival discovery systems have created a small number of provisional solutions to provide adequate context, but the archival community is still lacking a set of best practices for indexing archival description.

Resources

- Apache Solr
- Extensible Text Framework
- Blacklight
- VuFind

2.5.4 Search protocols

Potential candidates

- Search/Retrieve via URL (SRU)
- OpenSearch

Recommendation

- **Search/Retrieve via URL (SRU)**
- **OpenSearch**

Discussion

The meeting's attendees identified a need beyond the indexing layer for the application to present a standardized interface for queries using an established client-server search protocol. Some attendees suggested that all queries be routed through this layer rather than being submitted to the index layer directly. In other words, this protocol layer would be exposed both internally to the application, and optionally as a external interface to allow other systems to query the application's data. Several attendees recommended the possibility of using Search/Retrieve via URL, or SRU. This protocol is well-established within the library community and is RESTful in its design, which suggests that it is potentially suited for this application.

Additionally, some attendees suggested the possibility of using OpenSearch, a search protocol first developed by Amazon. The primary strength of OpenSearch is that it can return responses in RSS and Atom, which are formats that are parsed with relative ease by developers with even minimal technical knowledge. There also have been some efforts to harmonize SRU and OpenSearch to allow for potential interoperability between the two protocols. Attendees also acknowledged a similar lack of best practices for search protocols in the representation of contextual information and hierarchy as was identified during the discussion of the indexing layer for the application. Despite this gap, SRU provides a promising solution as it can return responses using any potential schema that is identified as supported by the server.

Resources

- Library of Congress: Search/Retrieve via URL
- OpenSearch
- OpenSearch SRU Extension (Draft 1)

2.5.5 Mechanisms for harvesting and syndication

Potential candidates

- Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH)
- Atom and/or RSS feeds
- Open Archives Initiative Object Reuse and Exchange (OAI-ORE)
- Topic Maps
- XML Sitemaps
- RDFa or other microformats

Recommendation

- **No clear recommendation; consider prioritizing based upon application's requirements**

Discussion

The attendees nominated a variety of potential mechanisms for harvesting and syndication but there was no clear consensus around preferring any of the individual formats or mechanisms, as each of them have distinct merits. The OAI Protocol for Metadata Harvesting is well established in the library community. Atom and RSS feeds are easily understood by both lay users and developers, and have served as one cornerstone for the development of "mash-ups" and visualizations. OAI Object Reuse and Exchange is an emerging set of specifications that allow systems to exchange information about aggregations of web resources. XML Sitemaps are a standardized representation of a set of URLs that comprise a given website. Topic Maps allow for the expression of the relationships between topics and their occurrences within a given body of knowledge. RDFa, as well as other microformats, are easy to incorporate in the output templates for human readable representations of the application's resources. Choice of one or more of the specifications does not necessarily preclude incorporating others into the application. Given the varying strengths of these standards and the relative easy of implementing more than one, the ArchivesSpace team should consider prioritizing some of these methods based upon further definition of the application's requirements and the resources available.

Resources

- OAI Protocol for Metadata Harvesting
- OAI Object Reuse and Exchange
- RFC 4287: Atom Syndication Format
- Sitemaps Protocol
- Wikipedia: Topic Maps
- W3C: RDFa Primer
- Microformats.org

2.6 User interface

Potential candidates

- "Thick" desktop client that runs natively on end-user machines (e.g. the existing Archivists' Toolkit application)
- Dynamic web application, reliant on browser-provided functionality only (e.g. JavaScript)
- Dynamic web application, reliant on browser plug-ins (e.g. Java or Flash)
- Dynamic web application, primarily reliant on browser-provided functionality, but with specific functionality reliant on plug-ins as appropriate

Recommendation

- **Dynamic web application, primarily reliant on browser-provided functionality, but with specific functionality reliant on plug-ins as appropriate**

Discussion

The meeting's attendees quickly eliminated the option of a desktop client application from the list of potential solutions. Reasons for its dismissal included the added complexity of managing and supporting individual installations of the application. Most attendees supported the creation of a dynamic web application that relied on established technologies supported by most web browsers, such as JavaScript and CSS. This option was strongly favored over creating a web application that relied on browser plug-ins such as Java or Flash for its dynamic functionality. Additionally, Fluid Infusion, a JavaScript user interface library, was identified as a potential candidate to allow for the responsiveness and application behavior comparable to desktop applications. Some attendees also noted that the ArchivesSpace development team should consider the possibility of requiring plug-ins for specific types of functionality if identified as part of the application requirements. For example, the application may want to provide basic media player functionality for video files made available in the discovery interface for the application.

The attendees also considered a few supplemental issues related to user interface development. One primary area of concern was whether the development team should declare a minimum set of browser requirements for the merged application. Some attendees noted that other comparable dynamic web applications have either defined their own set of minimum requirements or are reusing sets defined by specific JavaScript libraries. As an example, CollectionSpace has created a two-tiered categorization of browsers that establishes levels of support and whether user interface bugs are considered as issues that would block the release of a given version. A few attendees also asked the extent two which emerging Web technologies such as HTML5 Video and HTML5 LocalStorage would be supported by the new application. These decisions would likely have to be balanced between the minimum browser requirements and the functional specifications developed for the merged application.

Several attendees also reiterated the need for prioritizing the accessibility of new application, such as providing keyboard-based navigation alternatives for every area of the application's functionality. Additionally, a few attendees also suggested that user interface elements and application behavior degrade gracefully to ensure complete or nearly complete functionality when a user does not have JavaScript enabled in their browser for whatever reason. Finally, a number of attendees also urged the development team to plan for iterative user testing on the application, which could include providing feedback on wireframes and comprehensive layouts, and full usability testing as application functionality is developed. CollectionSpace currently performs weekly user testing, with approximately 15-20 active participants per week. Iterative user testing could also be a low-barrier way to have the archival community assist with the development process for the merged application.

Resources

- CollectionSpace: Supported Browser Environment
- Fluid Infusion
- Mark Pilgrim: "Video on the Web," *Dive into HTML5*
- W3C: Web Storage (Draft)

2.7 Authentication

Potential candidates

- Local, application-specific authentication
- Exclusive use/recommendation of an individual or set of single sign-on systems
- Creation or reuse of an existing pluggable authentication system

Recommendation

- **Local, application-specific authentication**
- **Creation or reuse of an existing pluggable authentication system**

Discussion

During the technical planning meeting, nearly all attendees agreed that both a local authentication mechanism, backed by data stored in the persistence layer, was a key baseline requirement for the merged application. Additionally, attendees clearly stated that the application needs an abstracted, pluggable authentication layer that allows for implementers to use a particular authentication system supported by their institution. With this in mind, much of the discussion did not focus on providing a detailed evaluation of particular authentication mechanisms. However, attendees recommended potential support for a number of authentication systems, such as the Central Authentication Service (CAS), LDAP, Pubcookie, Shibboleth, WebAuth, and OpenID. Attendees representing the perspective of information technology managers ranked IP address-based authentication as an extremely low priority. Use of a pluggable authentication mechanism would also allow implementers to create additional modules that authenticate against other single sign-on systems not supported in the merged application's initial development phases, such as institution-specific systems.

The attendees identified a number of potential issues for the architecture of the authentication subsystem of the merged application. The primary issue was considering how to provide support for multiple authorization regimes in a given deployment. One means to do this would be to allow for "stacked authentication," which would allow the application to attempt authenticating users using an administrator-ranked selection of identity providers. An alternative solution, implemented in CollectionSpace, would be to allow individual users to specify a specific identity provider which they authenticate against. It may also be possible to combine these two approaches to allow for greater flexibility in the merged application. Attendees also emphasized

the need to define a base authorization role for users authenticating using a particular mechanism. For example, given an archives that is part of a university library system, users could be defined as being part of a "university community member" role, which may have a certain level of researcher access and would exclude any staff-specific or administrative functionality from that role.

Resources

- CollectionSpace: Authentication Service Description and Assumptions
- CollectionSpace: CollectionSpace Identity Provider
- DSpace: Authentication (re: stackable authentication)

2.8 Authorization

Potential candidates

- Local, application-specific role and access control list (ACL) definition and management
- Extensible Access Control Markup Language (XACML)

Recommendation

- **Local, application-specific role and access control list (ACL) definition and management**
- **Extensible Access Control Markup Language (XACML)**

Discussion

The discussion about the authorization during the meeting was divided primarily between the consideration of options for role and access control list (ACL) management and persistence and architectural concerns about authorization. A few authorization frameworks, such as the Zend ACL module for PHP and Spring Security for Java, were mentioned, but there was no detailed discussion of them clearly as candidate technologies given the lack of discussion of using specific programming languages. One attendee stated that existing libraries may not allow for a full abstraction of authorization for the merged application, and therefore the developers would therefore need to tie authorization-related code more closely to the application logic than desired.

As with the authorization requirement, the ArchivesSpace team initially proposed a local, application-specific role and access control list management tool that would store the role and ACL definitions in the application's persistence layer. While Archon and Archivists' Toolkit both have role functionality, currently only Archon allows administrators to configure those roles or assign permissions granularly to individual users for individual application modules. Some attendees also noted other applications, such as Drupal and ICA-AtoM, provide a similar level of granularity for the role definition and permission assignment. The attendees therefore recommended providing a similar level of functionality within the merged applications. As with

both AT and Archon, the merged application should be released with a set of five to ten predefined roles that will suit the needs of most implementers.

One meeting attendee asked the development team to consider how the application's user interface layer or the application's service layer would be enforcing the defined access control lists. This attendee noted that this enforcement would need to occur at both layers. Service layers would have to enforce the ACLs given that the would be relatively agnostic about the clients that accessed them; providing no ACL enforcement at the service layer would therefore make the application fundamentally insecure. By comparison, ACL enforcement would also need to occur at the user interface level to suppress user interface components that individual users may not be authorized to see or access. Given these divergent needs, this attendee suggested the use of the Extensible Access Control Markup Language, or XACML, as a potential candidate to describe access control policies for merged application.

Resources

- Drupal 6: Permissions
- OASIS Extensible Access Control Markup Language Technical Committee
- CollectionSpace: Authorization Description and Assumptions
- CollectionSpace: Design notes for authorization
- CollectionSpace: Design notes for multi-tenancy

2.9 Workflow management

Discussion

While the meeting had breakout sessions and plenary discussion dedicated to workflow management, project staff chose to table discussion for the second day as the meeting's attendees identified significant issues within this area. As identified in the Background Readings and Discussion of Architectural Areas, neither Archivists' Toolkit nor Archon have tightly defined workflow components to allow for flexibility in institutional practices. As a first step, several attendees recommended that the development team consider categorizing potential workflows for incorporation into two categories: automated workflows and human-based workflows, such as those involving approval processes. Attendees also emphasized that adding a workflow authoring tool would likely be too complicated for an initial release. Accordingly, if workflow was indeed a necessity for the merged application, the attendees suggested that support for an established workflow language such as the Business Process Execution Language (BPEL) be provided in the application.

The largest concern, however, was establishing whether the application actually needed a workflow component, as it will largely be just as much work to add "hooks" to implement workflow into the application as it would to actually include it in the application's implementation. Accordingly, the attendees advised the ArchivesSpace project team to establish the workflow needs for the merged application before committing to its implementation. Several

attendees also urged the project team to survey the community about their workflow requirements. One attendee added that some community members may view the addition of a workflow component negatively, as they may believe it could remove needed flexibility from the application.

Resources

- J. Gordon Daines and Cory Nimer, "Integrating Process Management with Archival Management Systems: Lessons Learned," *The Code4lib Journal*

2.10 Reporting

Discussion

As with workflow management, project staff tabled further discussion of the merged application's reporting component after the breakouts and plenary discussion about it. While attendees acknowledged that the merged application would have to ship with a number of prepackaged reports similar to those provided by Archivists' Toolkit, there was a lack of agreement about what actually comprised the activity of reporting. Some attendees asked the project team to determine if reporting was just an alternate representation of search results or if it was a clearly distinct activity that needed a tightly defined set of requirements. One attendee suggested creating a set of criteria that would assist in the project team in distinguishing between reports for managerial decision making and reports needed for archivists to perform their day-to-day work.

A few attendees added that there are considerable integration concerns about reporting. Most of the existing report platforms assume that the data for reporting is stored in a relational database; however, no conclusive recommendation was identified for the merged application's persistence layer (see "Storage/persistence layer" above). Other attendees also suggested that there may be little demand for "self-serve" custom reports. If this is the case, this could have significant impact on how the development team prioritizes the implementation of functionality related to reporting. Therefore, several attendees recommended that the project team should outsource the creation of reports during the application's development cycle. These attendees also suggested that the project team provide a standardized set of instructions and configuration information to allow interested implementers to develop their own reports.

2.11 Additional recommendations by meeting attendees

2.11.1 Overview

Discussion

In addition to discussing candidate architectures for the merged application, the attendees at the technical planning meeting were given the opportunity to provide feedback to the ArchivesSpace

team on any additional area of concern. Attendees provided additional recommendations in three key areas: project management and development methodology, community development and involvement, integration concerns, multi-tenancy, and licensing.

2.11.2 Project management and development methodology

Discussion

Many attendees suggested that the development team for the merged application should choose a particular development methodology as early as possible. In particular, several attendees suggested that the development team consider using an agile software development methodology, such as Scrum. One attendee noted that if a non-agile methodology was chosen, the project team should have business reasons that can be articulated to stakeholders, including both funders and the archival community. Even after choosing a particular development methodology, the attendees also noted that the development team would need to determine details such as the appropriate duration for development sprints and how to offset the work of the various development teams. For example, in some cases, a team responsible for development of the service layer will need to be ahead of the application and user interface layer team, and the functional specifications team will need to be ahead of the service development team. A few attendees also stated that development of core application architecture may be difficult in a distributed setting, and therefore urged the development team to consider doing in-person code sprints for architecture, if possible. Integration of the application's stack may become more complicated as development progresses, and accordingly another attendee advised the development team to perform this integration as early as possible. Finally, several attendees also stressed the importance of establishing a clear testing methodology for the application, possibly using additional tools such as a continuous integration server.

Resources

- Wikipedia: Agile software development
- Mike Cohn: *Agile Estimating and Planning*
- Mike Cohn: *User Stories Applied: For Agile Software Development*
- Wikipedia: Test-driven development
- Wikipedia: Continuous integration

2.11.3 Community development and involvement

Discussion

Attendees stressed the importance of developing a community around the application and providing means by which the community can be involved in the application's development. In particular, the attendees suggested that the ArchivesSpace team develop a regular schedule for issuing announcements about the project. Several attendees specified that the development process needed to be transparent; means to achieve this included an active and open e-mail discussion list for developers, use of wiki software such as Confluence, and issue tracking

software such as JIRA. A few attendees also suggested that the project team create a variety of low-barrier and low-commitment opportunities for community members and potential implementers to become involved. These opportunities could include active solicitation of individuals to participate in user testing or the writing of documentation. Providing "non-technical" users seemingly informal opportunities for participation in the application's development may allow those individuals to have a greater sense of ownership in the merged application.

2.11.4 Integration concerns

Discussion

Despite the meeting's overall emphasis that application required a modular design, meeting attendees discussed a few integration concerns for the application's development. Attendees asked the development team to consider specifying the ways in which the application would need to integrate with other systems, such as digital repository software like Fedora and DSpace, as well as identifier services such as ARK and the Handle System. In addition, given the development by the Archivists' Toolkit community, a few attendees recommended that the application should be extensible and provide a command-line interface wrapper to the application's service layer.

Resources

- CollectionSpace: Accessing Services from your programs or scripts

2.11.5 Multi-tenancy

Discussion

Multi-tenancy is a method within software architecture that allows a single instance of an application to serve multiple customers. Each customer, or "tenant," can configure the application as needed or desired. Application data for a given tenant is in virtual isolation from the data of other tenants. Multi-tenant deployment can be desirable solution if a given organization or service provider would otherwise need to maintain several instances of a single-tenant application for a given base of clients. In addition, it provides for better scalability for service providers since the application's architectural components can be scaled based on the needs of individual tenants. For example, some tenants may be given a higher performance persistence layer than most if they have large, complex descriptions. Given that multi-tenant deployment was identified as a potential goal in the AT/Archon Hi-Level Requirements, several attendees suggested that the development team commit to accommodating multi-tenancy from the beginning of the development process. However, attendees also advised the ArchivesSpace team that multi-tenant development was complicated. In particular, one attendee urged that the team hire a software architect that has done multi-tenant development before, preferably multiple times. Additionally, if the project is committed to supporting multi-tenant deployment, it should also be incorporated into the testing framework for the application.

Resources

- Wikipedia: Multitenancy
- Salesforce.com: The Force.com Multitenant Architecture: Understanding the Design of Salesforce.com's Internet Application Development Platform
- CollectionSpace: Design notes for multi-tenancy in CollectionSpace

2.11.6 Licensing

Discussion

The attendees also spent a considerable amount of time discussing the implication of choosing an appropriate open source software license for the project. Potential nominees included the Apache Software License (ASL), version 2.0, and the Educational Community License (ECL), version 2.0, which is a variant of the Apache Software License. Several attendees suggested avoiding viral licenses, such as the GNU Public License (GPL). Licensing will most likely have an impact on how the application is distributed, particularly in terms of the ability of the ArchivesSpace team to create a packaged, installer-based version of the merged application. One attendee noted that given the application's code base was released under a non-viral license such as the ASL or the ECL, a "one-click installer" could be packaged under the GPL. This would allow developers or implementers who were not comfortable with the viral licensing of the packaged application to download the application's dependencies that may be under viral licenses separately. Another attendee suggested explicitly excluding dependencies that are under viral licenses, which is the approach taken by the Sakai project. Finally, one attendee suggested the possibility of using Creative Commons licenses for components like templates or reports that implementers may want to share with the larger community.

Resources

- Apache Software License, Version 2.0
- Educational Community License, Version 2.0
- Wikipedia: Viral license
- Sakai: 3rd Party Licenses and Software
- Creative Commons

3 Addenda

3.1 List of meeting attendees

- David Ackerman, Associate Vice President, ITS .edu Services/Executive Director, NYU Libraries DLT Services, NYU
- Robin Dale, Director of Digital & Preservation Services, LYRASIS
- Luc Declerck, Associate University Librarian, Technology Services, UC San Diego
- Jon Dunn, Associate Director for Technology, Indiana University Digital Library Program, Indiana University
- Tom Habing, Programmer/Analyst, Digital Library Research, University of Illinois
- Susan Harum, Director, ArchivesSpace, University of Illinois
- Brian Hoffman, Digital Library Publication and Access Manager, NYU
- Mark Matienzo, Digital Archivist in Manuscripts and Archives, Yale University, and Technical Architecture Consultant, ArchivesSpace
- Al Matthews, Programmer at Robert W. Woodruff Library, Atlanta University Center
- Mark McFarland, Associate Director for Digital Initiatives, University of Texas / TDL
- David Millman, Director, Digital Library Technology Services, NYU
- Megan Forbes, Collectionspace Project Manager, Museum of the Moving Image
- Joseph Pawletko, Software Systems Architect/Technical Lead, Digital Library Technology Services, NYU
- Patrick Schmitz, Semantic Services Architect/CollectionSpace Co-Technical Lead, UC Berkeley
- Scott Schwartz, Archivist for Music and Fine Arts, Sousa Archives and Center for American Music, University of Illinois
- Paul Sorenson, Programmer, Archon, University of Illinois
- Nathan Stevens, Programmer/Analyst, Digital Library Technology Services, NYU
- Mike Teets, Vice President, Innovation, OCLC
- Zach Thomas, software engineer, Aeroplane Software/ Sakai
- Brian Tingle, Technical Lead, Digital Special Collections, California Digital Library
- Peter Van Garderen, President/Systems Archivist, Artefactual Systems
- Brad Westbrook, Head, Metadata Analysis and Specification Unit, UC San Diego Libraries, University of California, San Diego
- Mike Winkler, Director, Information Technology and Digital Development, University of Pennsylvania

3.2 Background Readings and Discussion of Architectural Areas

Background information

Archives

Archives are the non-current records of individuals, groups (including families), and organizations (including corporations and government bodies) that are retained by an archival repository based upon their enduring value. Materials held by these repositories exist in a wide variety of formats, including traditional paper records, photographs, sound recordings, films and video, and digital records held on physical media or kept as files or recordkeeping systems on networked storage. Archivists use a distinct set of professional principles and practices that serve as workflows for the materials under their care. Archivists *select or appraise* material based upon its context of creation and analysis of the evidence and information the records contain as well as their physical condition. When a repository chooses to accept a body of records from a donor, archivists *accession* the records, taking legal and physical custody from the donor and documenting information about the donor and the transferred records.

After accessioning, archivists *arrange and describe*, or *process*, the materials. Archivists arrange materials, often into a multiple-level hierarchy, to maintain and convey their context of creation and to achieve physical and intellectual control over the records. Through the process of description, archivists create *finding aids*, which characterize the physical and intellectual arrangement of the materials and provide contextual information about the creator or collector of the records such as biographies and administrative histories. Finding aids and other descriptive tools that follow the *General International Standard for Archival Description* (ISAD(G)) describe the collection as a whole and then allow the subsequent lower levels to either inherit information or define information relevant to that level. Archivists make these descriptive tools to ensure *access* and provide *reference services* for these materials for researchers. Given the historical, and some cases, financial, value of a repository's collections, potential users are often required to register before materials are made available to them.

More information

- Lisa Spiro, "Archival Management Software"
    - Archival workflow (~ 2 p.): http://archivalsoftware.pbworks.com/Archival-Workflow
        - Read in detail if unfamiliar with archives
- ISAD(G): http://www.ica.org/sites/default/files/isad_g_2e.pdf
    - Read pp. 7-12 and pp. 36-38 if unfamiliar with archival description

Encoded Archival Description

Encoded Archival Description (EAD) is an XML DTD and schema for descriptive tools such as finding aids and registers of archival collections. It was originally maintained by the EAD Working Group, which included members from the Society of American Archivists (SAA) and selected individuals from the international archives community. The first version of EAD (1.0) was released as an SGML and XML DTD in 1998. The second version, EAD 2002, was released in late 2002, which deprecated some elements, added other elements, and changed the logical structure of others. In part, some of the changes reflected the need to align the DTD and related entity files with best practices in XML implementation. In late 2006, a subgroup of the EAD Working Group released EAD 2002 as a schema in both Relax NG and XML Schema formats.

EAD reflects the hierarchical nature of archival description. Since its inception, EAD has been designed to be flexible and accommodating to allow institutions to encode their finding aids without significant reengineering of their descriptive data. In part, this has allowed EAD to become moderately successful. However, this flexibility has also limited the ability to some extent for EAD to serve as a successful transmission format between institutions given the potential for variation in local encoding practice. Some repositories have chosen to at least achieve institutional or consortial consistency by developing encoding guidelines.

EAD is now maintained by a standing Technical Subcommittee of SAA and the SAA Schema Development Team, which is now responsible for maintaining a five-year review cycle for EAD. While much of this work still has yet to get started, the groups have already expressed significant interest in the possibility of defining two versions, "loose" and "strict", that may determine the possibility for individual finding aids to be more widely interoperable.

More information

- Library of Congress EAD pages: http://www.loc.gov/ead/
    - History of EAD development (~ 7 p.): http://www.loc.gov/ead/eaddev.html
        - Skim if unfamiliar with EAD
    - Design Principles for Enhancement of EAD (~ 1 p.): http://www.loc.gov/ead/eaddesgn.html
        - Read if unfamiliar with EAD
- "Over, Around, and Through: Barriers to EAD Implementation" (OCLC/RLG Programs report, 44 p.): http://www.oclc.org/research/publications/library/2010/2010-04.pdf
    - Quickly skim most if unfamiliar with EAD, but all participants should read pp. 8-14 and pp. 26-27 closely

Data model

Both the AT and Archon data models are highly relational, but this is not a requirement in the new application if a superior solution exists. The current models form parts of the core "modules" or "packages" in both AT and Archon, and include the following:

- Names (AT)/Creators (Archon) - controlled vocabulary/authority file for persons, families, and corporate bodies
- Subjects (AT and Archon) - controlled vocabularies, with assignment of individual terms to particular vocabulary sources
- Resources (AT)/Collections (Archon) - description of archival collections plus their subcomponents
- Accessions (AT and Archon) - includes deaccessioning facility
- Digital objects (AT)/Digital library (Archon) - management and description of digital objects
- Locations (AT and Archon) - space management and assignment of resources, etc. to specific rooms, shelves, etc.
- Staff user management (AT and Archon)
- Researcher user management (Archon) - user information, appointment assignment, etc.
- Repository information (AT and Archon)
- Classification (Archon) - facility to manage a system of intellectual organization of records for a large institution, e.g. a university
- Rights management (AT)
- Assessments (AT; Archon only has assessments for AV material) - records high-level condition or organization information for unprocessed materials
- Reports (AT)

Additionally, the team created draft specifications for data models for date statements, extent statements, and references to external documents. These additional specifications were necessary based upon limitations in the current data models, and thus the new specifications allow for multiple instances of this information to be related to instances of other models, such as resource descriptions and accession records. In the case of date and extent statements, certain existing data models did not allow for the assignment of multiple structured values. The specification for external document references is a new requirement that expands upon existing functionality available in the AT. The team has also created a new specification for the accessions module that describes the relationships of these new types of subrecords to a core module of the application.

More information

- AT/Archon Hi-Level Requirements
    - See section on "archives functions"
    - Also see "API includes generic method..." and "Integrated data / record validation rules, processes, and messages" under "application features"

- Merged application specifications - read all, but focus especially on accession specification
    - Date statement specification
    - Extent statement specification
    - External document reference specification
    - Accession specification
- Archon API documentation: http://archon.org/doc/2.20/index.html
    - Quickly skim descriptions of classes only
- AT record validation rules (~ 3 p.): http://archivistst oolkit.org/sites/default/files/Record%20Validation%20Rules.pdf
    - Quickly skim only
- AT API documentation: http://archivistst oolkit.org/sites/default/files/javadocs/javaDocs2_0_0/org/archivistst oolkit/model/package-summary.html
    - Quickly skim descriptions of models only
- NLA Service Framework (~ 45 p.): https://wiki.nla.gov.au/download/attachments/15989/Service%2Bframework20081124%2BV%2B0.91.doc
    - Read sections 3.1-3.5, 5.1-5.4, 6.5, and 7.1-7.5 (~ 9 pp. total)
- CollectionSpace
    - Candidate Services and Related Notes (~ 9 p.): http://wiki.collectionspace.org/display/collectionspace/Candidate+Services+and+Related+Notes
        - Read

Deployment scenarios

The merged application will have to support a wide range of potential deployments. AT and Archon implementers range in size from single "lone arranger" repositories to large academic archives with 20+ FTE staff members. These could include installation of the application plus supporting technology stack on a single desktop machine, deployment of the application and its stack to a central server for a given institution (currently the model used by most AT and Archon deployments), to multitenant or multi-instance deployments managed by consortia. Currently, there are very few multitenant deployments; at this point, most deployments of this type rely on AT and Archon's ability to have multiple repositories defined within one database instance of the applications.

More information

- California Digital Library's multi-tenant AT/Archon deployment (~ 2 p.): http://www.cdlib.org/services/dsc/tools/at-archon.html
    - Skim
- 2008 Archivists' Toolkit User Community survey (8 p.): http://archiviststoolkit.org/sites/default/files/AT%20User%20Group%20SurveyResultsFD.pdf
    - Read pp. 1-2 for overview of types of archives, level of IT support, and number of FTEs
    - Skim rest
- CollectionSpace:
    - Design notes for multi-tenancy (~ 19 p.): http://wiki.collectionspace.org/display/collectionspace/Design+notes+for+multi-tenancy+in+CollectionSpace
        - Read
    - Nuxeo multi-tenancy issues (2 p.): http://wiki.collectionspace.org/display/collectionspace/Nuxeo+multi-tenancy+issues
        - Skim

Potential implementations

- Bare code install with instructions (currently provided by Archon)
- GUI-based installation/configuration helper (currently provided by Archon)
- GUI-based installer for application only (currently provided by AT)
- Installer for application and required technology stack
- Virtual disk image containing application and required stack (initially for evaluation/development and for single-user instances without a network connection, perhaps eventually for multi-instance deployments running on e.g. Amazon EC2 or VPS)

Architectural Areas

Authentication and authorization

AT and Archon currently rely solely on a database-backed authentication and authorization system. AT currently only has five distinct "levels" of access or roles that are not modifiable by users or administrators. Archon, by comparison, has a "usergroup manager" as well as a granular mechanism for granting individual permissions to individual users.

The ArchivesSpace archives team believes that the functional specification work required would be to combine the existing AT and Archon specifications. However, members of the AT community have also expressed significant interest in having a mechanism to allow for integration with existing authentication/authorization mechanisms such as LDAP. Given that other institutions use other authentication mechanisms, we will likely want to provide a pluggable authentication system that is easily extensible. We likely want to investigate the possibility of providing users with multiple authentication mechanisms (e.g. for an institution-maintained SSO system and approved "external" users who don't have a credentials for said SSO system), perhaps in a "stackable" model that allows multiple AuthN models to be tried in order until one succeeds.

It is a larger, open question if the application will use external group definitions (e.g. from LDAP or ActiveDirectory) to connect to specific roles within the application or if the policy definitions will be solely defined and managed from within the application.

More information

Skim most unless explicit sections are otherwise specified.

- AT Access Levels (3 p.): http://archivistst oolkit.org/sites/default/files/User%20Permissions.pdf
- Archon user manual (60 p.): http://www.archon.org/UserManualv2.21.pdf
    - Read section 4.2 (p. 15) for brief overview of user manager and user group manager.
- Discussion on AT list about LDAP (~ 3 p.): https://mailman.ucsd.edu/pipermail/atug-l/2010/thread.html#2226
- CollectionSpace
    - Authentication Service Description and Assumptions (8 p.): http://wiki.collectionspace.org/display/collectionspace/Authentication+Service+Description+and+Assumptions
        - Read
    - Authorization Service Description and Assumptions (7 p.): http://wiki.collectionspace.org/display/collectionspace/Authorization+Service+Description+and+Assumptions

        - Read
    - Authentication and Authorization User Stories (1 p.): http://wiki.collectionspace.org/display/collectionspace/Authentication+and+Authorization+User+Story+Summary
    - Roles and Permissions Requirements (1 p.): http://wiki.collectionspace.org/display/collectionspace/Roles+and+Permissions+Requirements
- Kuali Rice Identity Management (KIM) (1 p.): http://rice.kuali.org/kim
    - KIM Use Cases (2 p.): https://test.kuali.org/confluence/display/KULRICE/KIM+Use+Cases
    - KIM Requirements (3 p.): https://test.kuali.org/confluence/display/KULRICE/KIM+Requirements
    - KIM Architecture (1 p.): http://rice.kuali.org/documentation/1.0.1.1/UG_KIM/Documents/kimarchitecture.htm
    - KIM Features (2 p.): http://rice.kuali.org/documentation/1.0.1.1/UG_KIM/Documents/kimfeatures.htm
- Sakai
    - Sakai 3 Groups (4 p.): http://confluence.sakaiproject.org/display/GROUPS/Sakai+3+Groups+Home+Page
    - Sakai Nakamura Users/Groups (6 p.): http://confluence.sakaiproject.org/display/KERNDOC/KERN-488+Users+and+Groups
- DSpace
    - Stackable authentication methods (2 p.): http://wiki.dspace.org/confluence/display/DSPACE/StackableAuthenticationMethods
        - Read

Potential implementations

- Java Authentication and Authorization Service (JAAS)
    - JAAS white paper: http://www.research.ibm.com/people/k/koved/papers/acsac.pdf
    - JAAS Reference Guide: http://java.sun.com/javase/6/docs/technotes/guides/security/jaas/JAASRefGuide.html
    - Central to FeSL, Fedora's replacement for their legacy AuthN/AuthZ framework - http://www.fedora-commons.org/confluence/display/FCR30/Fedora+Security+Layer+%28FeSL%29
- Apache Shiro (currently in incubation): http://incubator.apache.org/shiro/
- Spring Security (formerly Acegi Security): http://static.springsource.org/spring-security/site/
- Kuali Identity Management: http://rice.kuali.org/kim

Storage/persistence layer

Archivists' Toolkit and Archon both use a relational database system as a persistence layer for all data. AT uses Hibernate as an object-relational mapping layer, and currently supports MySQL, MS SQL Server, and Oracle as backends. Archon uses MDB2 as a database abstraction layer with no ORM framework, and currently supports MySQL and SQL Server.

Additionally, Archon currently has an optional file system-backed storage component for digital objects. The AT/Archon Hi-Level requirements specify that the merged application include either integrated digital object storage or a migration pathway to another repository system for objects currently stored in Archon instances.

According to the Hi-Level requirements, the merged application – and by extension, the persistence layer – should be scalable to millions of objects and allow for implementation of a core data model that enables "quick search, retrieval, and display of information of any kind." This scalability requirement relates directly to the hierarchical nature of the data managed by archives repositories. For example, current limitations in AT require all resource records and their component records to be retrieved depth-first in their entirety. In other words, AT will retrieve data down to the deepest child record in a given hierarchy rather than only retrieving a parent record and its immediate children. Accordingly, the scalability of the merged application relies not only on the persistence layer, but also how the application implements the data model in relation to that persistence layer and the mapping layer that allows the application to manipulate instances of those data models.

More information

- AT/Archon Hi-Level Requirements
- CollectionSpace
    - Design notes for multi-tenancy - Storage: http://wiki.collectionspace.org/display/collectionspace/Design+notes+for+multi-tenancy+in+CollectionSpace#DesignnotesformultI-tenancyinCollectionSpace-Storage
        - Read this section in detail (~ 3 p.)
- Sakai
    - Sakai 3 overview presentation: http://www.slideshare.net/mkorcuska/sakai-3-version-8
        - See slides 30 & 31

Potential implementations

- RDBMS
    - Continue support of MySQL, MSSQL, and Oracle
    - Consider support of PostgreSQL

- File system
    - Scope of implementation - only legacy support for digital objects, or full file management capability going forward?
- Java Content Repository (JCR; JSR-170/JSR-283)
    - Backs both Sakai 3 (Apache Jackrabbit/Sling) and CollectionSpace (Jackrabbit through Nuxeo)
    - Still requires an RDBMS or some other backend storage

Import and export

AT and Archon both support importing data to and exporting data from the core modules in a variety of formats. The implementation of the import and export architecture will largely be dependent on the persistence layer chosen for the merged application.

Currently available import and export formats include the following:

- Imports
    - Resources (MARCXML, EAD)
    - Subjects and names (MARCXML, extraction from EAD)
    - Accessions (CSV, AT-specific XML schema)
    - Digital objects (CSV)
- Exports
    - Resources (MARCXML, EAD, CSV [for container labels])
    - Digital objects (MARCXML, Dublin Core, MODS, METS-wrapped DC and MODS)

In addition, the archives team has identified the following new import and export formats as part of the Hi-Level requirements for the merged application:

- Imports
    - Names (EAC, MADS, CSV)
    - Subjects (XML [which format?], CSV)
    - Digital objects (XML [which format?])
    - Locations (CSV)
- Exports
    - Resources (CSV [for folder labels])
    - Names (EAC, MADS)
    - Digital objects (MARCXML, PREMIS Rights metadata, PREMIS Technical metadata)

More information

- AT/Archon Hi-Level Requirements
    - Read sections on imports and exports
- CollectionSpace
    - Import and Export Use Cases (2 p.) : http://wiki.collectionspace.org/display/collectionspace/Import+and+Export+Use+Cases
        - Skim
    - Import and Export User Stories (1 p.: http://wiki.collectionspace.org/display/collectionspace/Import+and+Export+User+Story+Summary
        - Read in detail

- OLE Final Report (99 p.): http://oleproject.org/wp-content/uploads/2009/11/OLE_FINAL_Report1.pdf
    - Read "Obtain Metadata" process description (p. 45) and workflow/process diagrams (p. 46)
    - Read "Expose Metadata" process description (p. 53) and workflow/process diagrams (p. 54)

Reporting

For the sake of architectural discussion, reports could be considered as a type of export, but they also merit separate discussion because many archival repositories use reports to support business processes of archival repositories and decision making by department or organizational administrators. AT provides reporting capability through JasperReports, a Java-based reporting system. Currently, AT is packaged with a number of different reports that have become an essential information to report core operations statistics for archival repositories. However, users have had significant difficulty designing new reports for the AT given the difficulty of configuring JasperReports and iReport (JasperReports' end-user UI) for development. Archon does not currently have an explicit reporting module.

The merged application should likely have a report management component and/or integrate with an external application or platform that allows users to define new and modify existing report formats.

More information

- AT/Archon Hi-Level Requirements
    - See list of reports
- AT report descriptions (4 p.): http://archiviststoolkit.org/sites/default/files/Report%20Samples_Descriptions.pdf
- ATUG-L thread on JasperReports (~ 4 p.): https://mailman.ucsd.edu/pipermail/atug-l/2010/thread.html#2327
- CollectionSpace
    - Reporting Requirements (1 p.): http://wiki.collectionspace.org/display/collectionspace/Reporting+Requirements
        - Read
    - Reporting Service Description and Assumptions (1 p.): http://wiki.collectionspace.org/display/collectionspace/Reporting+Service+Description+and+Assumptions
        - Read
- Kuali Rice
    - Reporting Guide (2 p.: https://test.kuali.org/confluence/display/KULRICE/Reporting+Guide
        - Skim
- NLA Service Framework: https://wiki.nla.gov.au/download/attachments/15989/Service%2Bframework20081124%2BV%2B0.91.doc
    - See section 1.8 - "Report" (< 1 p.)

Potential implementations

- JasperReports: http://jasperforge.org/projects/jasperreports

- BIRT: http://www.eclipse.org/birt/phoenix/

Workflow management

Given the modular implementation of both AT and Archon, most aspects of application workflow are loosely defined. For example, to allow for divergence in local practice and requirements, both AT and Archon do not require an archivist to create accession records before they describe archival resources and their components. Nonetheless, there are some areas that require the incorporation of workflow management as specified by the archives team. Both AT and Archon impose record validation requirements, and the Hi-Level Specifications have identified the need to add configurable record validation requirements to the merged application. The merged application additionally may require functionality adapted from the Archon specifications for e-commerce and researcher request management, both of which will have workflow components.

More information

- OLE Final Report: http://oleproject.org/wp-content/uploads/2009/11/OLE_FINAL_Report1.pdf
    - Read "Manage User Relationship" (p. 84-88)
- NLA Service Framework: https://wiki.nla.gov.au/download/attachments/15989/Service%2Bframework20081124%2BV%2B0.91.doc
    - Read section 2, "Business Services" (p. 15-18)
- Sakai
    - Discussion of third-party workflow engines (2 p.): http://confluence.sakaiproject.org/display/SAKDEV/Workflow
        - Read
- Kuali Enterprise Workflow (1 p.): http://rice.kuali.org/kew/
    - Read

Potential implementations

- Java Workflow Tooling: http://www.eclipse.org/jwt/
- Ruote: http://ruote.rubyforge.org/
- Kuali Enterprise Workflow: http://rice.kuali.org/kew/

User interface

Currently, Archon's user interface is accessed through a web browser, and is primarily driven by PHP templates with some JavaScript. AT's user interface exists solely in the form of a Swing-based client application. As the Hi-Level requirements document specifies, the participants in the planning webinars expressed a strong preference for a browser-based web application. Nonetheless, certain participants also expressed the preference for a client application.

There is some confusion about the potential difference and/or need for both a browser-based application and a dedicated client application. Some of these issues may center around needs for current functionality as available in the AT, such as customizable rapid data entry screens and the ability to modify inline help and field labels. Additionally, there also is a strong need for better internationalization and localization for the merged application; the developers could leverage the community early on by allowing them to assist with the translation of the user interface elements.

Additional user interface needs as described by the Hi-Level Requirements include context-specific menus and a tab-system for each functional area, drag and drop for notes and hierarchical displays, auto-adjusting filters for search results, a wrap in XML tag functionality for editing certain fields of certain record types, configurable browse/return screens, search editors, and lookup lists, and consistent UI elements throughout the application.

More information

- AT/Archon Hi-Level Requirements
    - Read throughout, but see especially pp. 9-16
- CollectionSpace
    - UI Framework Evaluation (4 p.): http://wiki.collectionspace.org/display/collectionspace/UI+Framework+Evaluation
        - Read in detail
    - Choosing a UI Framework (3 p.): http://wiki.collectionspace.org/display/collectionspace/Choosing+a+UI+Framework
        - Skim - overview of UI frameworks for web applications; not an in-depth feature analysis
- Sakai
    - Sakai 3 UX Development Guidelines (12 p.): http://confluence.sakaiproject.org/display/3AK/Sakai+3+UX+Development+Guidelines+and+Information
        - Skim only

Discovery

Both AT and Archon support discovery of the data they main to varying extents. In the case of AT, the discovery functionality is limited to the client application itself and is targeted toward assisting staff to locate specific records. AT also integrates with external discovery systems indirectly through its ability export EAD and MARCXML instances of resource records and by the its addition of supplemental XSLT and XSL:FO stylesheets that allow implementers to produce styled versions of their detailed descriptive information. Archon provides discovery functionality for both staff users and researchers directly through its web-based interface.

Like AT and Archon, the merged application will need to provide both browse and search facilities as part of its discovery offerings. The existing Archon functionality has been identified as the candidate specification for adaptation for the merged application. Archon currently provides a themable, template-based platform for the browse and record display functionality. Individual records link to related content as specified in the data models. Additionally, the merged application should adapt and expand upon the ability to omit and/or suppress fields from display in output for unauthenticated/unauthorized users. There is also some interest from the community to add a mechanism for Web 2.0 features such as bookmarking and end-user annotations, but these have been identified as low priority for implementation.

While the existing applications currently rely on substring searching in the form of SQL-based "LIKE" queries, the Hi-Level requirements for the merged application are much more demanding. Desired features include the ability to limit search queries by date and configurable field weighting for search results. Accordingly, the merged application may require a more sophisticated search and indexing platform such as Solr. The application will likely need to ship with a well-documented and robust configuration that can be controlled from the administrative user interface.

More information

- AT/Archon Hi-Level Requirements
    - Read application features and web access (pp. 9-13)
- CollectionSpace
    - Keyword Search User Story: http://wiki.collectionspace.org/display/collectionspace/Keyword+Search
        - Read and click through to related user stories (~ 5 p. total)
    - UI Framework Evaluation (4 p.): http://wiki.collectionspace.org/display/collectionspace/UI+Framework+Evaluation
        - Read in detail
- NLA Service Framework: https://wiki.nla.gov.au/download/attachments/15989/Service%2Bframework20081124%2BV%2B0.91.doc
    - Read sections 4, 5.5, 5.8, 5.9, and 7.5-7.10 (~ 10 p.)

Potential implementations

- Lucene (indexing): http://wiki.apache.org/lucene-java/FrontPage
- Solr (indexing): http://wiki.apache.org/solr/

Other integration concerns

In addition to the architectural areas already specified, the Hi-Level Requirements also touch on a few larger needs for the merged application. In particular, the requirements specify the need for an "interface [or] method for connecting with, and dynamically updating, records in other web-based digital object access and/or repository systems." Some initial integration work between Archon, Drupal, and the Fedora repository platform has been done by developers at Northwestern using JMS messaging. The requirements also specify the need for a command line interface, and initial work has been done by a developer at the University of Southern California to add this functionality to the AT.

In general, the concerns seem to be driven around two types of integration needs. The first involves the need for a messaging mechanism to allow the merged application make announcements toother applications and platforms regarding changes of state in workflow processes and changes in values of metadata. The other involves ensuring that the services layer is developer-friendly enough to serve as a general API to retrieve and modify data maintained by the application in programmatic ways (e.g. bulk exports, updating individual records, etc.). Exposing the service layer as an official API will allow institutions to leverage the power of the merged application as an extensible platform for further development.

Meeting participants should also consider potential technologies that assist in exposure of metadata managed by the merged application to allow for seamless reuse within other platforms. This requirement in part is derived from the previously mentioned demand for functionality to connect with records in other applications. This need can be connected to both export requirements and discovery requirements. Potential implementations could include OAI specifications like the Protocol for Metadata Harvesting or Object Reuse and Exchange.

More information

- CollectionSpace
    - Common Services REST API (18 p.): http://wiki.collectionspace.org/display/collectionspace/Common+Services+REST+API+documentation
        - Read
    - Nuxeo API Issues (2 p.): http://wiki.collectionspace.org/display/collectionspace/Nuxeo+API+Issues
        - Skim
- Kuali Rice
    - Kuali Service Bus (1 p.): http://kuali.org/rice/ksb
        - Read

Potential implementations

- Java Message Service (JMS): http://java.sun.com/products/jms/

- Atom Syndication Format: http://www.ietf.org/rfc/rfc4287.txt
- PubSubHubBub: http://code.google.com/apis/pubsubhubbub/
- OAI Protocol for Metadata Harvesting: http://www.openarchives.org/pmh/
- OAI Object Reuse and Exchange: http://www.openarchives.org/ore/