Diving into the Technology of Hydramata:

A Pluggable, Extensible Hydra Solution for Research Repositories

Rick Johnson, University of Notre Dame
Linda Newman, University of Cincinnati
Open Repositories, 2014
Why are We Here Again?
New World of Scholarship

• Multimedia Student Projects
• Non-text Thesis
• Big Data
• Any Format as Data
• Everything is connected
Shared Needs

DATA CURATION EXPERTS

- DCE
- Notre Dame
- Northwestern
- Cincinnati
- Indiana
- Virginia

...
Curate Gem Mixed Architecture

Shared Repository Services

Ingest/Management
- Content Mgmt
- Ingest/Transformation

Data
- Video
- Document
- Images

... Work/Collection

Digital Repository

Search index

IR Public View
- Browse and Search
- Document Viewer
- Data Viewer
- Image Viewer
- Audio/Video Streaming
Emerging Goals / Refreshed Vision

• Each institution’s implementation is part of a larger ecosystem not merely a standalone application, and thus the code should support institutional customization.

• The core system should be separate from discrete components and provide a reference implementation for each component. Each component can then be extended or exchanged according to the needs of an individual institution.

• Component customization may also be shared among like-minded institutions because the interface between the components and the core will remain consistent.
Hydramata

A set of separate hydra-based automata, “hydramata”, that can be combined to form a single Hydra Head. Each hydramataton can be mixed and matched with other hydramata to provide wide format support tailored to fit your institutional or digital repository needs.
Transformation

Hydramata Example Hydra Head

Publish
- Digital Exhibits
- VIVO
- Online Journal
- Google Scholar
- ORCID
- SHARE

Discovery
- Images
- Audio/Video
- Data
- ETD

Viewers

Metadata

Workflow

Core
- Batch Ingest
- Local Discovery
- Submission Workflow
- Self Deposit
- Description

June 13, 2014
Open Repositories 2014
Problems To Be Addressed

➢ Hard to understand
  ○ Unclear data flow
  ○ Hard to assign story points

➢ Hard to maintain
  ○ Engine design hard to test
  ○ Slow tests

➢ Hard to configure
  ○ Configurable only at install time
  ○ Monolithic; lacking modularity

➢ Ingest unreliable and opaque
  ○ Fedora/Solr sync problems
  ○ Orphaned Fedora objects

➢ Not set up for advanced work
  ○ Role-based Access Control
  ○ Batch ingest
  ○ Multi-stage workflow

Decreasing velocity
Increasing errors
Diminishing ROI
Preservation failure
Design Approach

➢ Separation of concerns into functional units:
  ○ Determine their responsibilities
  ○ Define their interfaces
  ○ Minimize coupling between functional units

➢ Tie units into a conceptual framework
  ○ Build out from ingest processing at the core
  ○ Treat works as arbitrary sets of metadata/content
  ○ Support interactive and batch processing equally
  ○ Implement an Asynchronous transaction model
Design Approach – Core IR

- Users can login to the system using their university credentials
- We are confident about the accuracy, persistence, security, and recoverability of what users upload
- Users can upload works via UI or batch
- Users can assign a license agreement to their works
- Works can be described, organized, discovered, viewed
- Works can be shared with (or restricted from) the right people
- Users can collaborate with others with individual works
- Users will get metrics about how their works are viewed
- There will be a variety of roles for people to interact with the IR
- Contributors names will link to standard identifiers (e.g. ORCID)
- DOIs can be assigned by users to works
Beyond Core IR

Components considered outside of the core:

- IIIF
- TEI support
- Specialized viewers
- ORCID plugin (minting ORCIDs from Hydramata)
- Minting DOIs from Hydramata
- Syndication to external systems (using OAI, ResourceSync or similar): SHARE, ORCID, Google Scholar, DPN, APTrust, DPLA
- Exhibits (Spotlight)
- Contributor profiles, links to campus research information systems (VIVO)
Single Asynchronous(?) Transaction
Single Asynchronous(?) Transaction
Ingest Processing

Build on OAIS Reference Model concepts:

○ **SIP - Submission Information Package**
  ■ Gather user input into a SIP until validated
  ■ Maintain SIP until validated

○ **AIP - Archival Information Package**
  ■ Package work into Fedora Object(s)
  ■ Manage storage into Fedora

○ **DIP - Dissemination Information Package**
  ■ Aggregate work content and metadata in a common form
  ■ Provide in-progress and submitted works consistently
Single Asynchronous(✗) Transaction
Rebuild the Core

Yet to determine where some elements fit in:
Role-Based Access Control
Multi-stage workflow support
Problems Addressed

➢ Easier to understand
  ○ Design model clarifies overall function
  ○ Modularity makes tasking less difficult

➢ Easier to maintain
  ○ Created as an app not an engine
  ○ Decoupling helps test creation/execution

➢ Easier to configure
  ○ More intelligence in the core
  ○ Variation in works encapsulated in data

➢ Ingest process built to handle errors
  ○ Fedora/Solr updated only with valid data
  ○ No orphaned Fedora objects

➢ Accommodates advanced work
  ○ Reference implementations of modules allow programming to the interface while deferring full implementations

Steady velocity
Decreased errors
Increasing ROI
Preservation
What About Fedora 4?
Swappable Hydramata

<table>
<thead>
<tr>
<th>Images</th>
<th>Datasets</th>
<th>Articles</th>
<th>ETDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deposit</td>
<td>Image Metadata Fields</td>
<td>Dataset Metadata Fields</td>
<td>Article Metadata Fields</td>
</tr>
<tr>
<td>Mediate</td>
<td>Image Approval Workflow</td>
<td>Dataset Approval Workflow</td>
<td>Article Approval Workflow</td>
</tr>
<tr>
<td>Ingest</td>
<td>Image Processing Pipeline</td>
<td>Dataset Processing Pipeline</td>
<td>Document Processing Pipeline</td>
</tr>
<tr>
<td>Discovery</td>
<td>Image Gallery Views</td>
<td>Default Search Behavior</td>
<td>Default Search Behavior</td>
</tr>
<tr>
<td>Viewer</td>
<td>Zoomable Image Viewer</td>
<td>Dataset Visualization Tools</td>
<td>Paginated Document Viewer</td>
</tr>
<tr>
<td>Disseminator</td>
<td>Tiling Image Server</td>
<td>Default File Disseminator</td>
<td>Default File Disseminator</td>
</tr>
</tbody>
</table>

June 13, 2014 Open Repositories 2014
Plugging In and Out

Hydramata Application and Discovery Layer

- Spotlight
- SHARE
- ORCID
- ... (other components)

Hydramata Persistence Layer

- DIP Mapper
- ArchivesSpace
- VIVO
- Hydramata Work Data Resolver
- DIP Mapper
- DIP Mapper
- DIP Mapper
- DIP Mapper
- Hydramata DIP Mapper
- Avalon
- ... (other components)

Hydra

June 13, 2014
Open Repositories 2014
Plugging In and Out

- Other Hydra Head
- Hydramata Application and Discovery Layer
- Hydramata Work Data Resolver
- Hydramata DIP Mapper
- Hydramata Persistence Layer
Plugging In and Out

ETD
Images
Video
Hydramata Work
Data Resolver
Hydramata DIP Mapper
Hydramata Persistence Layer
Acknowledgements

• Jeremy Friesen, Technical Lead
• Dan Brubaker Horst, Product Owner
• Paul Clough, Developer
• Ray Lubinsky, Product Owner
• Sue Richeson, Developer
• Julie Rudder, Lead Product Owner
• Mike Stroming, Developer
• Entire Hydramata Project Team (15+ individuals)
Contact Info

• Rick Johnson, rick.johnson@nd.edu
• Linda Newman, newmanld@ucmail.uc.edu
• Project Hydramata wiki:
  – https://wiki.duraspace.org/display/hydra/Hydramata+Project
• Curate code base:
  – https://github.com/projecthydra-labs/curate
• Hydra Community:
  – http://projecthydra.org/
Q & A

http://en.wikipedia.org/wiki/Helsinki_Workers%27_House#mediaviewer/File:Paasitorni_by_the_sea.jpg
http://creativecommons.org/licenses/by-sa/3.0/