Digital Curation: Curation Micro-services approach to building repositories

Mark Phillips
UNT Libraries
November 8, 2010
Digital Curation

“Digital curation is the set of policies and practices focused on maintaining and adding value to trusted digital content for use now and into the indefinite future. Curation encompasses preservation and access, and can be applied to the humanities, social sciences, and sciences.”

http://www.cdlib.org/services/uc3/about/index.html
Preservation is the goal, it is like the finish line in a relay, you hand the responsibility off to various actors with an overarching goal being that the content us viable at a later point in time.
Focus on doing the smartest thing you can do right now.
Smartest does not equal “best”
Like many areas,
“the perfect is the enemy of the good”.
Digital Curation
Digital Stewardship
Digital Preservation
• Access
• Preservation

• Near-term
• Long-term

• Maintain
• Add value

• Cross discipline
Curation Micro-Services

• California Digital Library
• Methodology for building infrastructure to support curation and preservation
• Thinking about the components and interactions in a repository as a set of smaller services
• Loosely coupled services
• Reaction to large monolithic repository systems
• Unix philosophy for system design
• Output of one service is the input for another yet to be created service
• Swap out pieces as needed
• Focus on simple tools that do one thing.
• Often referred to as “building blocks” or “Legos”
At this time it isn’t exactly clear what is and what isn’t a Curation Micro-Service
Kind of sounds like a Web-service
Or any other service for that matter
This really hasn’t been “answered” in the community
CDL “services”

- Identity Service
- Storage Service
- Fixity Service
- Replication Service
- Inventory Service
- Characterization Service
- Ingest Service
- Index Service
- Search Service
- Transformation Service
- Notification Service
- Annotation Service
Some example components

- Anvl
- Namaste
- Pairtree
- BagIt
- CAN
- D-Flat
- ReDD
- Checkm
- Cutie
- ERC
Pairtree

- filesystem hierarchy for holding objects
- Identifier strings mapped to object directory
- Two characters at a time
  - abcd -> ab/cd/
  - abcdefg -> ab/cd/ef/g/
  - 12-986xy4 -> 12/-9/86/xy/4/

- Object folder at the end of the mapping
Full Example

current_directory/
  |  pairtree_version0_1  [which version of pairtree]
  |  ( This directory conforms to Pairtree Version 0.1. Updated spec: )
  |  ( http://www.cdlib.org/inside/diglib/pairtree/pairtreespec.html )
  |
  |  pairtree_prefix
  |  ( http://n2t.info/ark:/13030/xt2 )
  |
  \--- pairtree_root/
       |--- aa/
       |  |--- cd/
       |  |  |--- foo/
       |  |  |  | README.txt
       |  |  |  | thumbnail.gif
       |  |  ... 
       |  |--- ab/ ...
       |  |--- af/ ...
       |  |--- ag/ ...
       |  ... 
       |--- ab/ ...
       ...
      \--- zz/ ...
           | ...
Namaste

- NAMe AS TExt
- file naming convention
- primitive directory-level metadata tags exposed directly via filenames
- Answers the following question
  - “What kind of directory is this?”
- Examples
  - 0=bagit_0.96
  - 0=untl_sip_1.0
  - 0=untl_aip_1.0
  - 0=untl_acp_1.0
Building a repository

• UNT Libraries
• Two separate systems with similar components
• Access system = Aubrey
• Preservation system = Coda

• Built as a set of “services”
UNT and micro-services

• Modular
• Build out in stages
  – Presentation System
  – Preservation System
  – Other services as we need them
• Replace in the future as needed
• Easy to implement, easy to discard
Identity Service

• Archival Resource Keys (ARK) for identifiers
• Number Server for minting names for objects
• Implemented as a Web service
• Query a URL and get a new unique name
  – metapth12604
• Append that to UNT’s NAAN
  – ark:/67531/metapth12604
• Currently 5 name spaces for identifiers
Vocabulary Service

• Simple system for providing canonical versions of names
• Unique identifiers for each vocabulary term
• Provided as Linked data in RDF/XML
• Other serializations
  – Legacy XML format
  – Json
  – Python object
• Easy to integrate into code
• Promotes reuse of vocabularies
Storage Service

- Provide a consistent way of requesting an item
- Use http for communication
- Read only currently
- Makes use of public specifications
  - CAN
  - PairTree
  - BagIt
- Exposed with Apache
Storage Service Example

- For a known identifier, and a known storage service coda1gel on coda-005
- http://libdigiarch1.library.unt.edu/coda-005/store/pairtree_root/co/da/1g/el/coda1gel/

```
|-- 0=can_0.10
|-- admin
|-- can-info.txt
|-- log
 `-- store
   |-- pairtree_index
   |-- pairtree_prefix
   |-- pairtree_root
    `-- co
     `-- da
      |-- 1g
       `-- el
        |-- coda1gel
            |-- 0=untl_aip_1.0
            |-- bag-info.txt
            |-- bagit.txt
            |-- coda_directives.py
            `-- data
```
Storage Service Example

• Proxy for abstracting “which node”
• We expected to never have all of our data in one place
• Shifts the problem from infrastructure/storage to a software problem

http://coda.library.unt.edu/ark:/67531/coda1gel/
http://coda.library.unt.edu/ark:/67531/coda1gel.html
http://coda.library.unt.edu/ark:/67531/coda1gel.urls
http://coda.library.unt.edu/ark:/67531/coda1gel.atom.xml
http://coda.library.unt.edu/ark:/67531/coda1gel.rdf
http://coda.library.unt.edu/ark:/67531/coda1gel/bag-info.txt
Storage service

• Coda repository application has a list of active content nodes
• Coda queries each content node for desired object, (http head requests)
• Primary and secondary content nodes are usable for increased fault tolerance
• Coda streams content to end user to allow for very large files to be transferred
Replication Service

- Software neutral content replication
- Master nodes in Library server room
- Secondary nodes at Library Annex server room
- Coda instance at each location
- Different number of content nodes 6 vs 3 currently
- Different content node sizes 9TB vs 25TB
- Need to balance content across content nodes
Replication Service

• Series of conventions for making content available for replication

• Three requirements
  – Provide a list of objects you want to replicate
  – Point to a manifest defining all files of an object
  – Provide a way to validate an object when replicated
Replication Service – Coda Implementation

• Restful replication service

• Components
  – Replication queue – Queue of objects to replicate
  – Collector – Adds object to the Replication queue
  – Harvester – queries queue for objects to harvest
  – Coda Metadata Store

• As content is replicated, a validation and replication event is logged centrally.
Event Service

• Based on the PREMIS Event Model
• Restful interface for creating new events
• Provides an interface for creating and maintaining PREMIS Agents
• Collects and provides access to events important to the lifecycle of the object
• Currently setup to capture ingest, replication, fixityCheck and virusCheck events
Ingest Service

• A more complex workflow for accessioning content into the repository
• Uses BagIt as a packaging container
• Validation of content each network or disk hop
• Sanity check after atomic moves
• Folder based workflow with python management scripts
Folder Workflow

pth_dropbox/
|-- 0.Staging/
|-- 1.ToAIP/
|-- 2.ToAIP-Error/
|-- 3.ToACP/
|-- 4.ToACP-Error/
|-- 5.ToArchive/
|-- 6.ToAubrey/
|-- 7.ToAubrey-Sorted/
|-- 8.ToAubrey-Sorted-Error/
|-- dropbox_config.py
  |-- makeACP.py -> /home/digitalprojects/coda/makeACP.py
  |-- makeACPSort.py -> /home/digitalprojects/coda/makeACPSort.py
  |-- makeAIP.py -> /home/digitalprojects/coda/makeAIP.py
`-- moveToLibDigiArch_coda-005.sh
1. ToAIP

- Objects start in this directory, typically using rsync from local machines.
- Full validation of Bag
- Check that Bag is a Submission Information Package (SIP)
- Check for coda_directives.py for processing instructions
- Request identifier from Number Server
- Create METS document from supplied files
- Create PREMIS record, JHOVE stream, File stream
- Move to 3. ToACP on success or 2. ToAIP-Error on failure
3. ToACP

- Check that Bag is an Archival Information Package (AIP)
- Check for coda_directives.py for processing instructions
- Process METS structure and create Web derivatives based on current practice
- Move AIP to 5. ToArchive on Success, Move AIP to 4. ToACP-Error on failure
- Move ACP to 6. ToAubrey on Success
5.ToArchive/6.ToAubrey

- Run bash script to rsync contents of 5.ToArchive over to current archival dropbox
- Run makeACPSort.py to sort contents of 6.ToAubrey into odd and even folders, upload to appropriate content node on delivery system
Ingest Service

- Archival Information Package (AIP) is ingested into Coda in a very similar fashion, it has the following steps
  - Verify Bag
  - Check bag is AIP
  - Assign coda identifier

- Access Content Package (ACP) is moved to the Aubrey content delivery platform and made available in the following systems
  - http://texashistory.unt.edu/
  - http://digital.library.unt.edu/
Current statistics for UNT systems

• Coda
  – 27,552,721 files
  – 139,062 objects
  – 42.3 TB in use / 120 TB capacity

• Aubrey
  – texashistory
    • 125,721 objects
    • 114,847 “live”
    • 1,248,416 “fileSets”
  – digital.library
    • 38,755 objects
    • 38,451 “live”
    • 2,253,031 “fileSets”
Questions?