“Where are We From, Where Are We Going?”
Permanent Objects, Disposable Systems

Stephen Abrams
Patricia Cruse
John Kunze
David Loy

California Digital Library
University of California
D'où venons nous, que sommes nous, où allons nous?

Paul Gauguin, 1897-98, Museum of Fine Arts Boston, 32.270
Where are we from, what are we, where are we going?

Paul Gauguin, 1897-98, Museum of Fine Arts Boston, 32.270
Where [is our stuff] from, what [is it], where are we going [with it]?

Paul Gauguin, 1897-98, Museum of Fine Arts Boston, 32.270
Where From? What? Where To?

Producer ➔ Archive ➔ Consumer
Where From? What? Where To?

Producer  ➔  Archive  ➔  Consumer

Ingest  ➔  Data management / archival storage  ➔  Access / preservation planning
Where From? What? Where To?

Producer  ➔  Archive  ➔  Consumer

Ingest  ➔  Data management / archival storage  ➔  Access / preservation planning

Provenance  ➔  Characterization  ➔  View paths
Repository Landscape

Increasing diversity in types and uses of content
Content arising from non-library contexts
Inevitable technological change
Design Goals

Devolve repository function into a set of independent, but interoperable, services

– Since each is small and self-contained, they are more easily developed and maintained

– Since the level of investment is lower, they are more easily replaced

Provide complex function through the flexible combination of atomistic services
Design Goals

Support interaction through procedural APIs, command line applications, and web interfaces

- Let content managers and curators interact with the services without requiring changes to existing work practices

Rather than force content to come to the services, push the services out to the content

- Easy deployment centrally or locally, either independently or in strategic combinations
Design Goals

Defer implementation decision making until needs and outcomes are clearly articulated

- Requirements are first stated as sets of *values* and *strategies* that promote those values
- Strategies are then embodied as abstract services, and, finally, instantiated in technical systems
## Object-Centric Values and Strategies

<table>
<thead>
<tr>
<th>Value</th>
<th>Justification</th>
<th>Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identity</td>
<td>To distinguish an object from others</td>
<td>Persistent naming, actionable resolution</td>
</tr>
<tr>
<td>Viability</td>
<td>To recover an object from its medium</td>
<td>Redundancy, heterogeneity, media refresh</td>
</tr>
<tr>
<td>Fixity</td>
<td>To ensure that an object is unchanged from its accepted state</td>
<td>Redundancy, error-correcting codes, message digests</td>
</tr>
<tr>
<td>Authenticity</td>
<td>To ensure that an object is what it purports to be</td>
<td>Provenance, cryptographically-secure signatures</td>
</tr>
<tr>
<td>Ontology</td>
<td>To understand an object’s significant nature</td>
<td>Syntactic, semantic, and pragmatic characterization</td>
</tr>
<tr>
<td>Visibility</td>
<td>To enable users to find objects of interest</td>
<td>Public discovery systems and registries, exposure for web harvest</td>
</tr>
<tr>
<td>Utility</td>
<td>To expose an object’s underlying content</td>
<td>Behavior-rich delivery</td>
</tr>
<tr>
<td>Portability</td>
<td>To facilitate content sharing and succession planning</td>
<td>Self-contained, self-documenting objects, packaging standards</td>
</tr>
<tr>
<td>Appraisalment</td>
<td>To understand the consequences of time</td>
<td>Analysis and assessment</td>
</tr>
<tr>
<td>Timeliness</td>
<td>To know when a preservation value is threatened</td>
<td>Technology watch, stakeholder engagement</td>
</tr>
</tbody>
</table>
## Service-Centric Values and Strategies

<table>
<thead>
<tr>
<th>Value</th>
<th>Justification</th>
<th>Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Availability</strong></td>
<td>To provide access at a time of user choosing</td>
<td>Redundancy, automated failover</td>
</tr>
<tr>
<td><strong>Responsivity</strong></td>
<td>To provide appropriate throughput</td>
<td>Redundancy, load-balancing</td>
</tr>
<tr>
<td><strong>Security</strong></td>
<td>To enforce appropriate use of services and content</td>
<td>Cryptographically-secure identity and role management</td>
</tr>
<tr>
<td><strong>Interoperability</strong></td>
<td>To facilitate creative reuse of content and services</td>
<td>Standard interfaces</td>
</tr>
<tr>
<td><strong>Extensibility</strong></td>
<td>To enable graceful evolution over time</td>
<td>Granularity, orthogonality, virtualization</td>
</tr>
<tr>
<td><strong>Trustworthiness</strong></td>
<td>To promote users’ sense of predictability and reliability</td>
<td>Transparency, audit, certification</td>
</tr>
<tr>
<td><strong>Sustainability</strong></td>
<td>To ensure ongoing access and use</td>
<td>Commodity components, institutional commitment, financial cost-recovery, professional development</td>
</tr>
</tbody>
</table>
Micro-Services

“Lots of uses keeps stuff valuable”

“Lots of services keeps stuff useful”

“Lots of description keeps stuff meaningful”

“Lots of copies keeps stuff safe”
Design Process

What are the conceptual entities underlying the service?

What are their state properties?

What are their behaviors?
Storage Service

Storage service
- An aggregation of storage nodes

Storage node
- A particular configuration of object storage

Object
- An aggregation of files over time

Version
- A particular configuration of files at a point in time

File
- A formatted bit stream
## Storage Service Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Idempotent</th>
<th>Safe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Help</td>
<td>idempotent</td>
<td>safe</td>
</tr>
<tr>
<td>Get-state</td>
<td>idempotent</td>
<td>safe</td>
</tr>
<tr>
<td>Get-node-state</td>
<td>idempotent</td>
<td>safe</td>
</tr>
<tr>
<td>Get-object-state</td>
<td>idempotent</td>
<td>safe</td>
</tr>
<tr>
<td>Get-object</td>
<td>idempotent</td>
<td>unsafe</td>
</tr>
<tr>
<td>Get-version-state</td>
<td>idempotent</td>
<td>safe</td>
</tr>
<tr>
<td>Get-version</td>
<td>idempotent</td>
<td>unsafe</td>
</tr>
<tr>
<td>Get-file-state</td>
<td>idempotent</td>
<td>safe</td>
</tr>
<tr>
<td>Get-file</td>
<td>idempotent</td>
<td>unsafe</td>
</tr>
<tr>
<td>Add-version</td>
<td>non-idempotent</td>
<td>unsafe</td>
</tr>
</tbody>
</table>
## Storage Service Interfaces

<table>
<thead>
<tr>
<th>METHOD</th>
<th>Get-File-State</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th></th>
<th>[idempotent, safe]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node</td>
<td>Identifier</td>
</tr>
<tr>
<td>Object</td>
<td>Identifier</td>
</tr>
<tr>
<td>Version</td>
<td>Identifier</td>
</tr>
<tr>
<td>File</td>
<td>Identifier</td>
</tr>
<tr>
<td>ReponseForm</td>
<td>Enum</td>
</tr>
<tr>
<td>RETURN</td>
<td>Response form</td>
</tr>
</tbody>
</table>

GET `/node/object/version/file?m=state` HTTP/1.1
Accept: application/json

% store –get `node/object/version/file` –m state –f JSON

File.getState ("node/object/version/file", Format.JSON);
Storage Service Implementation

c.a. 1989

- FTP
- POSIX
- SQL

c.a. 2029?

- HTTP
- URI
- XML

Due to their inherent abstracting nature, protocols and interfaces last longer than systems
**Storage Service Implementation**

Using the file system as the controlling managerial abstraction, what is the thinnest smear of additional functionality that will make it an effective object store?

- Namaste
- CAN
- Pairtree
- Dflat
- ReDD
Name As Text (Namaste) Tags

Directory-level signature files extending Dublin Core Kernel metadata

- [ Tag h0 ] \(0=name\_version\)
- Who h1 \(1=who\)
- What h2 \(2=what\)
- When h3 \(3=when\)
- Where h4 \(4=where\)
Content Access Node (CAN)

File system conventions (structure and reserved names) for an object store

can/
    0=can_0.2
    can-info.txt
    log/
    store/
        pairtree...
Pairtree

Use a bigram decomposition of an object’s identifier to determine its file system path

pairtree/
    0=pairtree_0.1
    pairtree-info.txt
    pairtree_root/
        id/
        en/
        ti/
        fi/
        er/
        dflat...
Dflat

A “digital flat” for object data and metadata

dflat/
  0=dlfat_0.11
dflat-info.txt
v001/
  d-manifest.txt
delta/
  redd...
v003/
  f-manifest.txt
full/
  data/
  metadata/
  enrichment/
  annotation/
Reverse Delta Directory (ReDD)

- File-level reverse delta compression

```
redd/
  0=redd_0.1
  add/
  delete.txt
```
Performance Scaling

Modern file systems, e.g. ZFS, exhibit good performance characteristics at reasonable scale.

- 2,272,000 files = 28.5 TB
- 127,058,820 files = 25.7 TB
Status

We are completing development of the foundational Storage and Identity services

- Identity is based on N2T (name-to-thing) and Noid systems

We are planning for the Ingest, Catalog, and Characterization services

- Characterization is based on JHOVE2

As these services become available they will be deployed centrally and locally on campuses
Summary

Safety through redundancy
Meaning through description
Utility through service
Value through use

Code to interfaces
Orthogonality, but interoperability
Composition, not addition
Bring services to content, not content to services
Questions?

http://www.cdlib.org/inside/diglib/

Stephen.Abrams@ucop.edu
Patricia.Cruse@ucop.edu
jak@ucop.edu
David.Loy@ucop.edu