Sixth International Conference on Preservation of Digital Objects
Mission Bay Conference Center, San Francisco
October 5-6, 2009

An Emergent Micro-Services Approach to Digital Curation Infrastructure

Stephen Abrams
John Kunze
David Loy

California Digital Library
University of California

The new curation landscape

Increasing number, size, and diversity of content, and content producers and consumers

- More stuff, smaller budget

Inevitability of disruptive changes in technology, user expectation, and institutional mission and resources

- "My grant requires a data sustainability plan"
- "I know I should be doing something more to protect my stuff, but I don't know what"
- "I don't want to preserve my stuff, just store it forever"

Assumptions

Curated content gains

- Safety through redundancy
- Meaning through context
- Utility through service
- Value through use

Decentralized curation can be as effective as centralized

Curation stewardship is a relay

Imperatives

Do more with less

Enable curation at the point of use

Plan for change

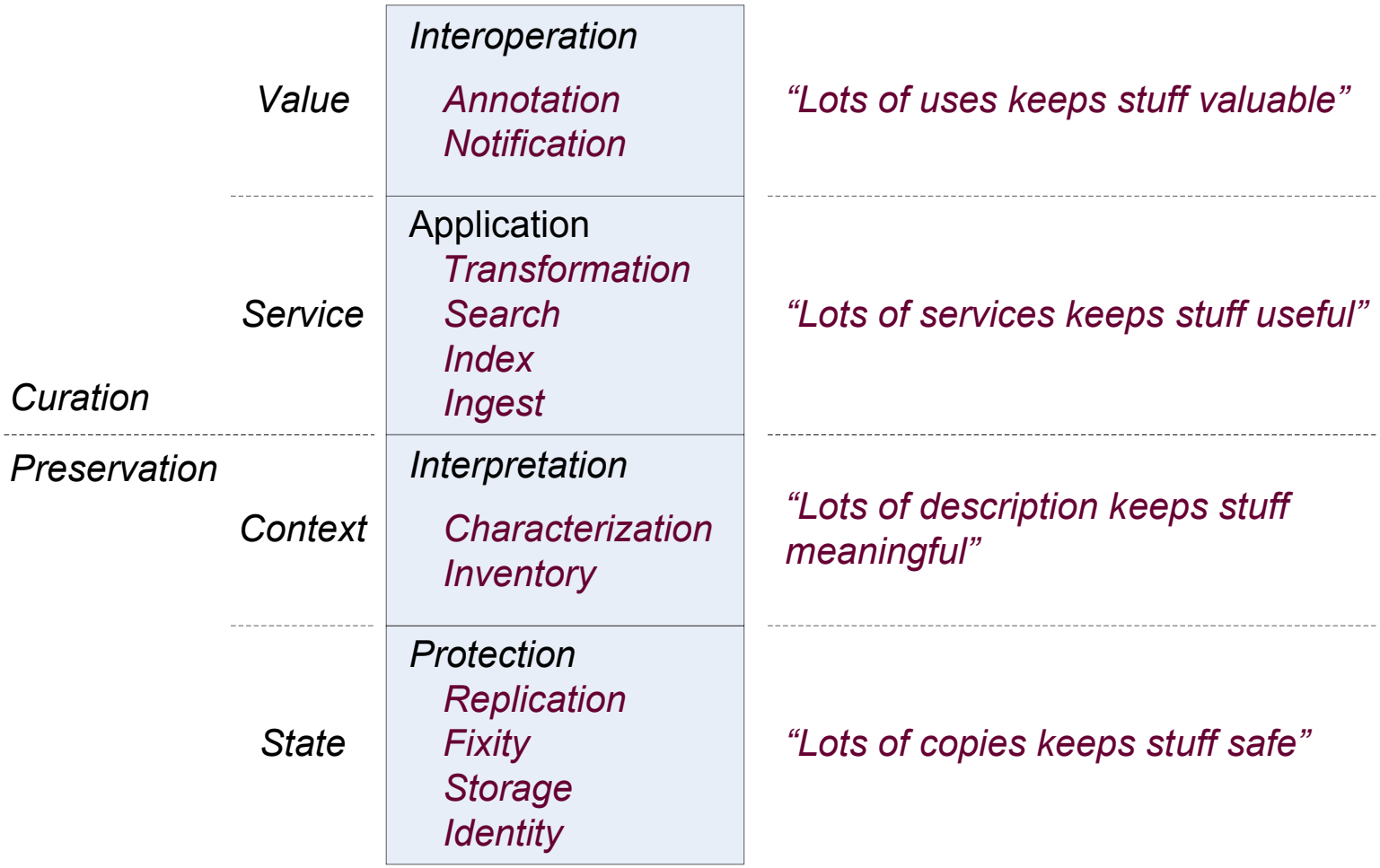
- Focus on content, not the systems in which that content is managed
- Ockham's Razor and Murphy's Law suggest
  - ✓ Favor the small and simple over the large and complex
  - ✓ Favor the proven over the (merely) novel

Curation micro-services

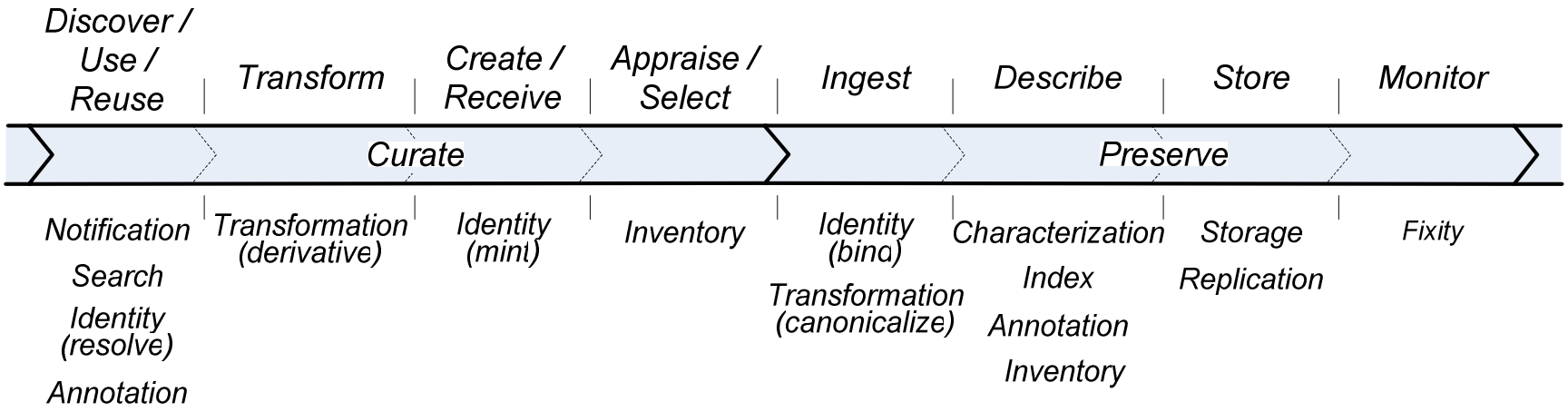
Devolve curation function into a granular set of independent, but interoperable micro-services

- Since each is small and self-contained, they are collectively easier to develop, deploy, maintain, and extend
- Since the level of investment in and commitment to any given service is small, they are easier to replace when they have outlived their usefulness
- Although the scope of each service is limited, complex behavior *emerges* from the strategic composition of individual, atomistic services

Curation micro-services



Curation lifecycle



Design principles

Model the major conceptual entities embodying a given service

- Defined in terms of state properties and behaviors that can access and manipulate that state

Assertions of persistence of curation function are made relative to interfaces

- Underlying implementation can and will evolve over time without invalidating interface service “contract”

Defer implementation decision-making until needs and outcomes are well understood

Storage service

Service

- Central broker to an arbitrary number of storage nodes

Node

- Object store encapsulating a particular technology, policy regime, or administrative scope

Object

- Digital representation of a coherent unit of abstract content

Version

- Set of files representing a discrete object state

File

- Named, *but not typed*, byte stream

Storage service

- Help
- Get-service-state
- Get-node-state
- Get-object-state
- Get-version-state
- Get-file-state
- Get-object
- Get-version
- Get-file
- Add-version
- Delete-object
- Delete-version

Storage service

METHOD Get-file-state			[*idempotent*, *safe*]
Parameter	Type	Obligation	Description
Node	Identifier	Mandatory	Storage node
Object	Identifier	Mandatory	Object identifier
Version	Identifier	Mandatory	Version identifier
File	Identifier	Mandatory	File identifier
Form	Enum	Optional	Response form
RETURN	State	Mandatory	File state
SIDE EFFECTS	*Not applicable*		
ERRORS	…		

```
GET /fileState/node/object/version/file HTTP/1.1
Accept: application/json
```

```
% store getFileState node object version file –f json
```

```
File.getState(node, object, version, file, Form.JSON);
```

Storage service

The general principles of granularity and orthogonality continue to apply to subsidiary specifications and conventions

- Content Access Node (CAN)
- Pairtree
- Dflat / Dnatural
- Checkm
- Reverse Directory Deltas (ReDD)
- Namaste (Name-as-text)

www.cdlib.org/inside/diglib

Storage service

Rely on the file system as the paradigmatic storage abstraction

- Modern file systems exhibit excellent scaling properties
  - ✓ Constant read/write time independent of number and size
  - ✓ Traversal time scales linearly with number and size
- The file system holds the “copy of record” of object metadata
- A duplicated subset of metadata is managed in the higher-level Inventory service as an optimization for routine administrative and curatorial queriess

Storage service

```
store/
      0=store_0.7
      admin/
      log/
      nodes.txt
      store-info.txt
```

```
can/
    0=can_0.8
    admin/
    can-info.txt
    log/
    store/
          0=pairtree_0.1
          pairtree-info.txt
          pairtree_root/
                        12/
                           34/
                              1234/
                                    0=dflat_0.16
                                    admin/
                                    current@
                                    dflat-info.txt
                                    log/
                                    v001/
                                         0=redd_0.1
                                         d-manifest.txt
                                         delta/
                                               add/
                                               delete.txt
                                    v002/
                                         0=dnatural_0.12
                                         manifest.txt
                                         admin/
                                         annotation/
                                         data/
                                         enrichment/
                                         metadata/
```

Development milestones

First wave ✓	Second wave ✓	Third wave	Fourth wave ✓	Fifth wave	Sixth wave ✓
Identity	*Inventory*	*Index*	*Search*	*Notification*	*Annotation*
Storage	*Ingest*	*Fixity*	*Replication*	*Characterization*	*Transformation*
Object and collection modeling			*Authentication and authorization*		
Policy and business model development					

Summary

- Provide for
  - Safety through redundancy
  - Meaning through context
  - Utility through service
  - Value through use
- Decentralization of applicability
- Granularity and orthogonality of service
- Complexity through composition, not addition
- Persistent interfaces, evolving implementations
- Reliance on the file system

Questions?

www.cdlib.org/inside/diglib

Stephen.Abrams@ucop.edu
John.Kunze@ucop.edu
David.Loy@ucop.edu