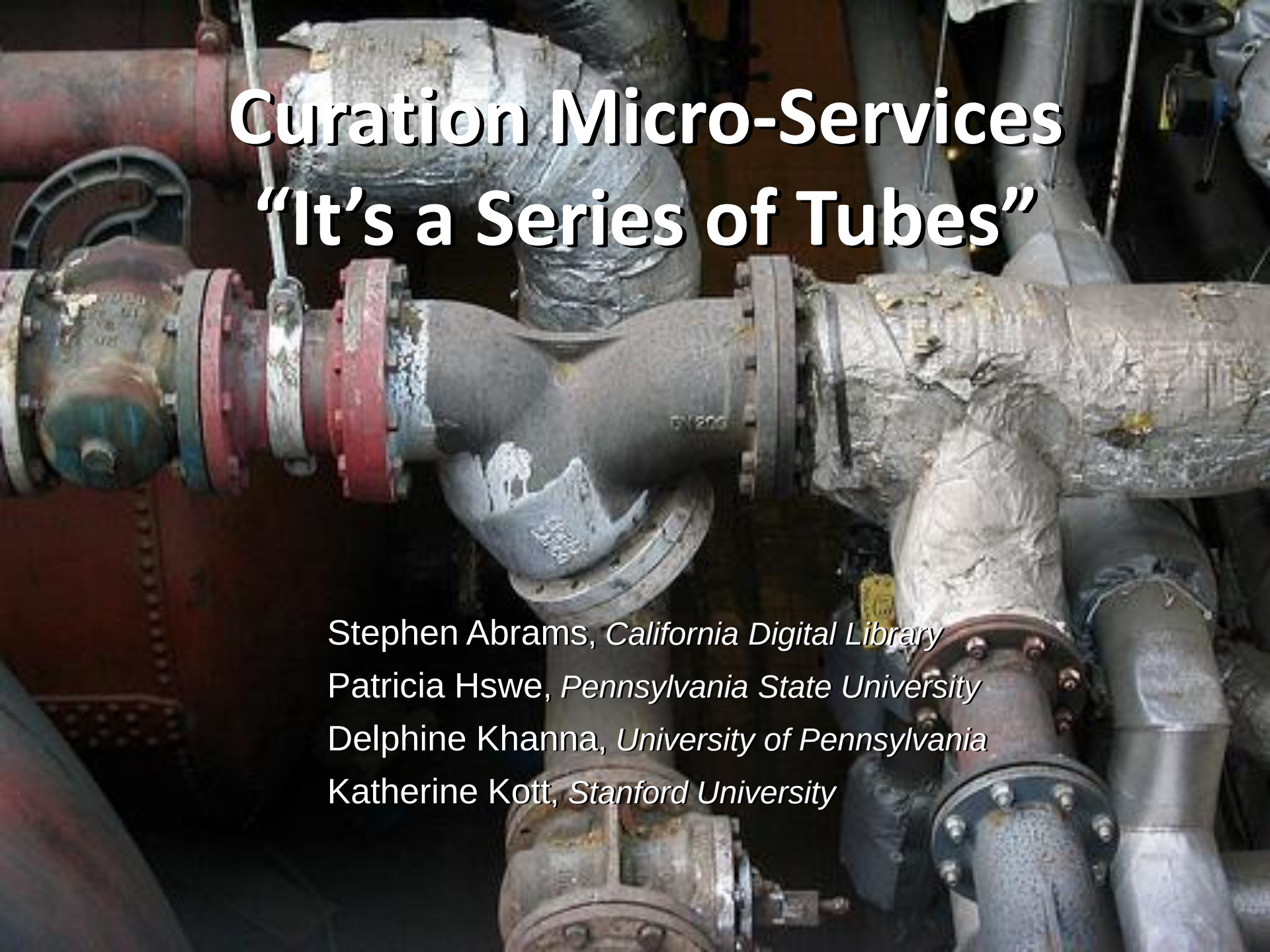


Curation Micro-Services "It's a Series of Tubes"

Stephen Abrams, *California Digital Library*

Patricia Hswe, *Pennsylvania State University*

Delphine Khanna, *University of Pennsylvania*

Katherine Kott, *Stanford University*

The Unix philosophy

"Make each program do one thing well"

"To do a new job, build afresh rather than complicate old programs by adding new features"

"Expect the output of every program to become the input to another, as yet unknown, program"

"Design and build software … to be tried early"

"Don't hesitate to throw away the clumsy parts and rebuild them"

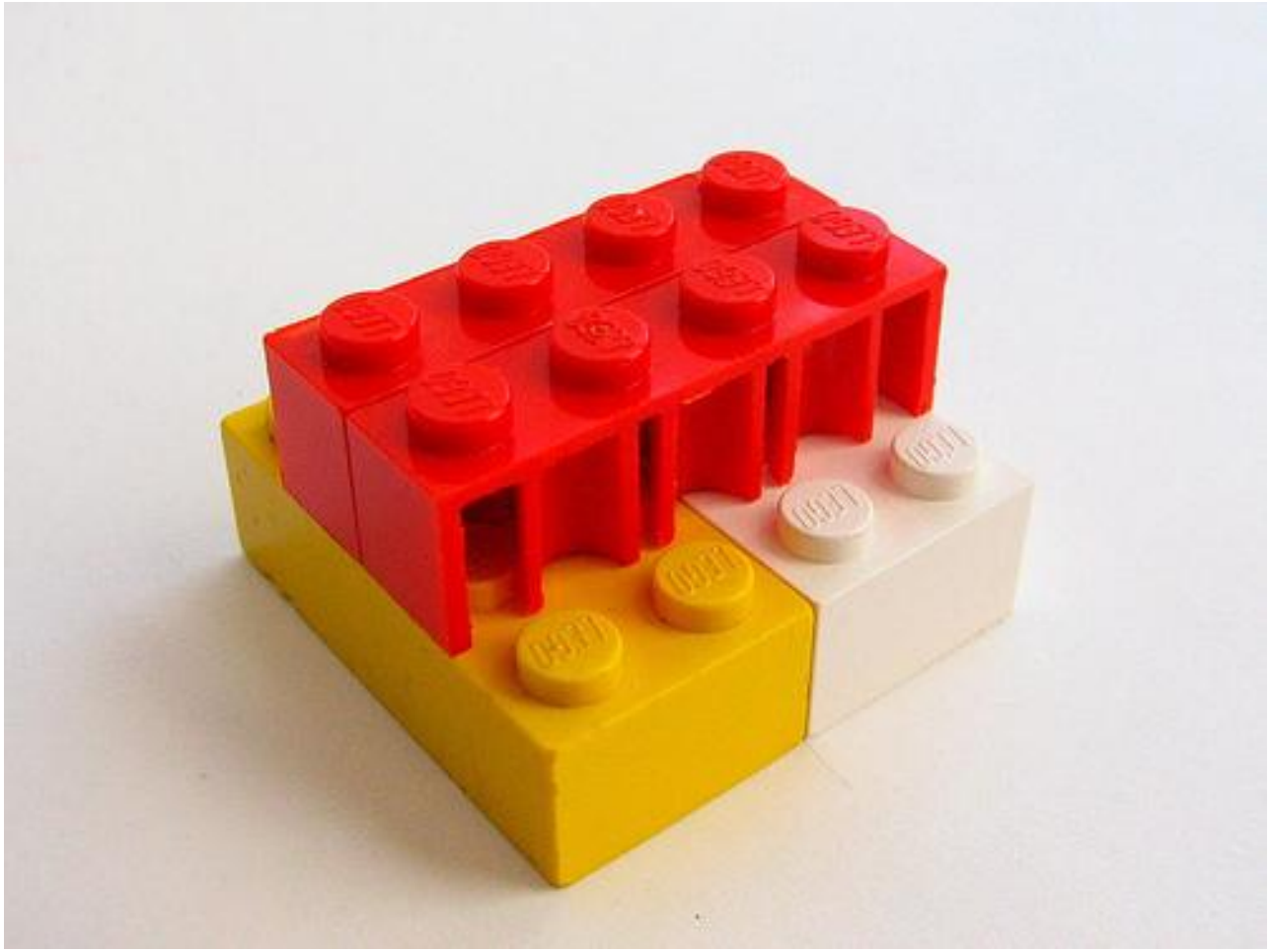
— D. L. McIlroy et al., "Unix time-sharing system forward," *Bell System Technical Journal* 57:6, part 2 (1978): 1902

The micro-services "philosophy"

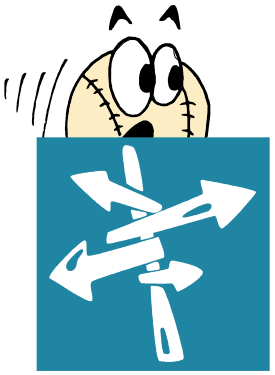
Metaphors	*Assumptions*	*Principles*	*Preferences*	*Practices*
Pipeline	Safety through redundancy	Modularity	The small and simple over the large and complex	Focus on outcomes, not means
Lego bricks	Meaning through context	Granularity	The minimally sufficient over the feature laden	Complexity through composition, not addition
	Utility through service	Orthogonality	The configurable over the prescribed	Policy neutral, platform and protocol independent
	Value through use (and reuse)	Emergence	The proven over the (merely) novel	Approach sufficiency through incrementally necessary steps
	Stewardship is a relay	Evolution		Early prototyping, frequent refactoring
		Parsimony		Code to interfaces

http://groups.google.com/group/digital-curation

The micro-services “philosophy”



Design goals



Principle of least surprise

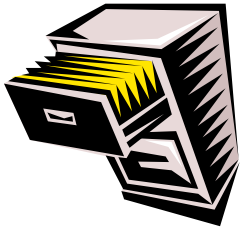
Multiple interface modalities

- RESTful HTTP
- Command line
- Procedural (Java, Perl, Ruby, …)

Linked data

Stable URL references http://example-store/state/default/1234/3/xyz

The file system is the database



Micro-services @ CDL

Mode	*Focus*	*Value*	*Service*		*Valence*	*Visibility*
Curation	Value	Accretion	Annotation	UI / Access control / Message queuing	Interoperation	User-facing
		Visibility	Notification			
	Utility	Accessibility	Access		Application	
		Derivation	Transformation			
		Selectivity	Search			
		Actionability	Index			
		Stewardship	Ingest			
Preservation	Context	Epistemology	Characterization		Interpretation	Provider-facing
		Ontology	Inventory			
	State	Reliability	Replication		Protection	
		Fixity	Fixity			
		Stability	Storage			
		Identity	Identity			

Micro-services @ CDL

First wave	*Second wave* ✓	*Third wave*	*Fourth wave* ✓	*Fifth wave*	*Sixth wave* ✓
Identity	Inventory	Index	Search	Notification	Annotation
Storage	Ingest / Access	Fixity	Replication	Characterization	Transformation
IDm / Authn / Authz			Metadata standards		
Object / collection modeling			Semantic interoperability		
Policy / business model development					

http://www.cdlib.org/uc3/curation
http://merritt.cdlib.org/

Micro-services pilot project @ PSU

Background

- Four legacy systems (platforms) for delivering digital content
- Platform review in spring 2010
- Key gaps/needs that are priorities
- Oh and by the way: PSU Libraries have never had an institutional repository

Micro-services pilot project @ PSU

"Gang of Four"

- Architect
- Programmer
- Archivist
- Curator

Goals of the pilot (begun in summer 2010)

Where we are (as of fall 2010)

Still to be completed (by late 2010/early 2011)

Overall take-away – prototyping a curation / stewardship service

Micro-services @ UPenn

Our Core Team

- Peter Cline
- Michael Gibney
- Delphine Khanna

Deciding on micro-services

- Needed to reorganize our “repository” functionalities
- Reading a lot about CDL’s work on micro-services
- Liked the possibility of *incremental* development

Micro-services @ UPenn

Implementation

- Decided to start implementing functionalities, based on Can/Pairtree/Dflat
  - ✓ One step at a time.
- Chose Java, already our language of choice for other pieces of architecture.
- Bumped into early version of CDL code for their “Storage Service” (mentioned on the wiki)
  - ✓ A portion of what we needed to develop
- Decided to use it, because it would save us soooo much time. And it did. Thank you CDL! :-)

Micro-services @ UPenn

1st phase of development

- Now in production
- Modest goals (proceed incrementally!)
- Implemented as we migrated from MrSid to JPEG 2000 for delivery format:
  - ✓ Create derivatives for all master TIFFs and store them in a Can/Pairtree/Dflat repository.
  - ✓ At first a Digital Object = mostly a JPEG 2000 file + a thumbnail)
- It works great!

Micro-services @ UPenn

2nd phase of development (happening now)

- Add the TIFFs
  - ✓ Put them in the Dflat objects, instead of keeping them on separate storage
  - ✓ No more direct access to the TIFFs
  - ✓ Need to change the workflows accordingly
- Add new format: PDF
- Add JHOVE and more quality-control steps

Micro-services @ UPenn

Experience so far

- It has been a great experience
- We certainly want to go on with this!

Collaboration

- We are hoping to contribute some of our code back at some point
  - ✓ But it is not clear how we should/could do it, given CDL's current model

What do you want to talk about?