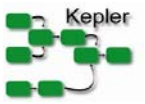
Workflow Exchange and Archival: The KSW File and the Kepler Object Manager

Shawn Bowers
(For Chad Berkley & Matt Jones)
University of California, Davis
May, 2005



Outline

1. The Kepler Object Manager
2. Archival and Exchange via KSW Files
3. Local Object Cache
4. Work in Progress (literally)

Motivation

In SEEK (and generally, Kepler) we:

Envision a large number of science-specific actors
- E.g., ecology, biodiversity, geoscience, bioinformatics, chemistry

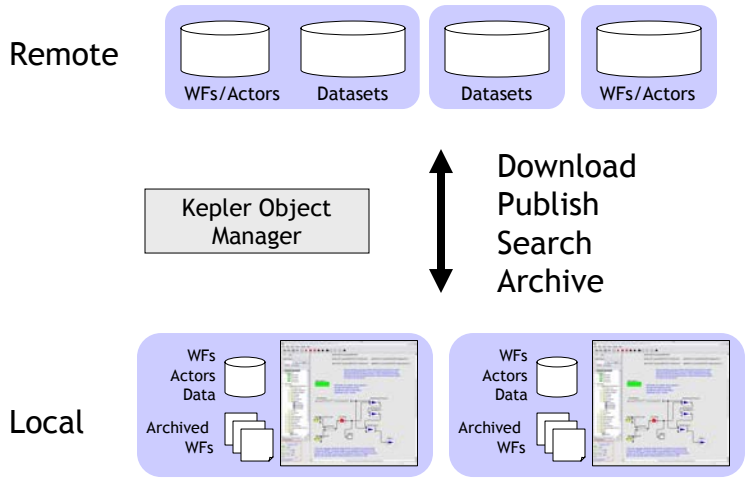
Envision a large number of available data sets
- Some may be fairly large; already various "collections" (e.g., LTER, GEON)

Want to enable search / discovery of actors, datasets, and workflows

Want to enable wide-scale sharing and re-use of data and workflows (and their components) across disciplines

… and have desired more than CVS to accomplish these goals

Managing distributed collections of workflow "objects"



Main Issues and Goals

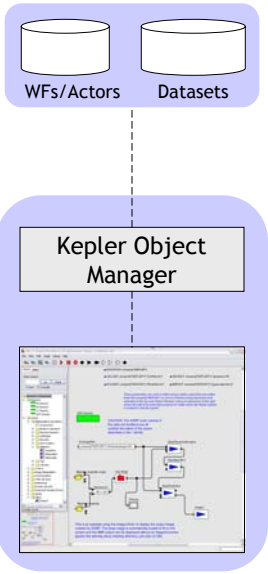
Handle various kinds of objects
- Workflows, Actors, Datasets, Libraries, Applications, Ontologies, Metadata, …

Transport objects
- Publish to and download from remote repositories
- Recognize local versus remote objects

Search for objects
- Both local and remote repositories w/in Kepler

Version objects
- E.g., managing conflicts in dependency chains

Enable "functional" groups
- Core group plus packages, e.g., biodiversity, geospatial, R, etc.



Overall Architecture

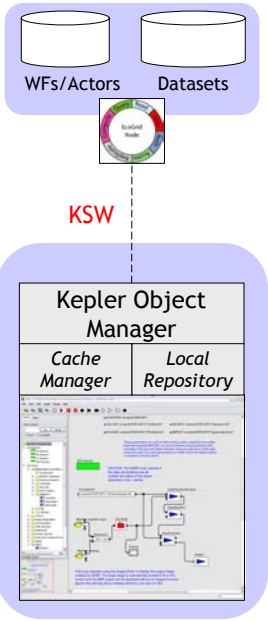
Life-Science Identifiers (LSIDs)
- Similar to DOIs (urn's, etc.)
- Includes (some) versversioning support
- Well-defined APIs

Kepler Scientific Workflow (KSW) Files
- An archive/jar file of objects
- KSW Metadata (a la MoML)
- For publish/download/archive/groups

EcoGrid
- Remote Access/Query
- A "thin" client / protocol

Local Object Repository
- Object retrieval, indexing, etc.

Local Object Cache
- Managing local vs. remote objects
- Dynamic class loader



KSW Files

An archive/jar file of objects

- Actor code and metadata, libraries; Dataset, metadata, etc.
- A manifest (what's in the file)

All objects in KSW files have associated LSID identifiers

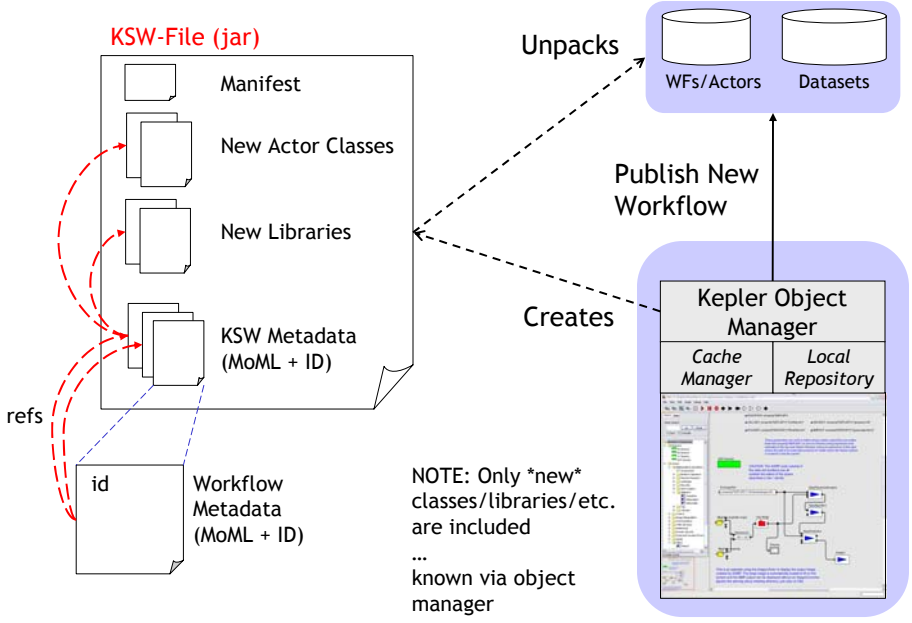
- An LSID is a urn, e.g., urn:lsid:kepler.org:actor:1000:2

Authority Namespace OID Version (opt.)

Objects have associated metadata files (basically MoML)

- Distinguish between "object definitions" and "object references"
  - Objects within a KSW file have corresponding definitions
  - Dependent objects not included are denoted via references
- Semantic types, dependency information, ports and types, etc.

KSW Metadata example (notional)



Local Object Cache

- Helps manage LSIDs
  - Creates them locally (for local actors, datasets, etc.)
  - Resolves LSIDs (both local and remote)
  - Handles publication (local id -> repository id, etc.)
- Support for packing/unpacking KSW files
- Provides simple database-like capabilities
  - "Persistent" storage / file indexing
  - Result caching (e.g., for remote queries)
  - Temporary files
  - Object indexing (e.g., dependency graphs, etc.)
- Handles multiple formats (within Kepler)
  - Stream versus File access
  - Representation conversion (e.g., binary interleaved, ascii grid)

Work In Progress

Still much to do …

- Still finalizing Object-Manager interfaces / APIs
- The object cache is partly implemented
- Simple KSW files can be packed / unpacked
- The KSW metadata format still being finalized
  - Working on MoML parsing, handling ids, etc.
  - Defining properties, like semtypes, dependencies, refs, etc.
  - We have a simple implementation of lsids for actor lib.

Our goal is to leverage Ptolemy's strengths …

- Most of what we have added / plan to add is layered on top of Ptolemy
- We want well-defined, generic interfaces
- We are soliciting volunteers !