IRODS and Federation

Adil Hasan

University of Liverpool

Max Plank e-science Seminar Repository Systems

25-26 June 2009

Digital Repository?

- Digital Repository: used to store a communities digital content (documents, audio-visual, structured data, etc).
- Recognition that digital content important, increasing real-time (global) collaboration.
- Recognition that not all the important digital content is 'publishable'.

Content Management System?

- Used to store a communities digital content (documents, audio-visual, etc).
- Primarily focussed on managing the content not on disseminating the content.
- Focus is on the content production side:
  - e.g managing multiple updates to a digital object.

Digital Archive?

- Digital Archive: used to store a communities digital content that has been appraised as being worthy of long-term preservation.
- Requires consideration about storage and access.
- Now all repositories holdings considered important.
- Seems now that Digital Repository → Digital Archive.

Digital Repository Requirements

- Can see general set of requirements:
    - To reliably store data for a defined period of time.
    - To allow discovery of the data by the designated community.
    - To allow sense to be made of the data.
    - To ensure data are accessible.
    - To allow relationships to be made between data.
    - To allow updates to the data to be made.

iRODS?

- Based on considerable experience from Storage Resource Broker (SRB) developed by DICE group.
    - Found many groups used SRB to store large quantities of data.
    - A lot of server-side post-processing of the data is required (e.g. replicate files, convert to different format, checksum etc).
    - Almost all management is Policy driven.

iRODS?

- SRB experience motivated requirements for a new data management system:
  - Contained all SRB functionality.
  - Add work-flow to manage server-side post-processing.
  - Configurable – only include the 'services' you need.
  - Open-source – SRB license imposed sever restrictions on the academic community.

iRODS?

- integrated Rule Oriented Data Management System.
- Developed by Data Intensive Cyber Environments (DICE) group at UNC and UCSD.
- Can be seen as a basis for a Digital Repository/Archive.
- Digital Repository/Archive is a Policy Driven System.

iRODS

- Client-server middleware
- Consists of database holding metadata information.
- Server applications – one for each storage resource.
- Rule engine applications – one for each storage resource.
- One server application interfaces to database.
- Client applications/API: C, Java, Python, PHP.

iRODS

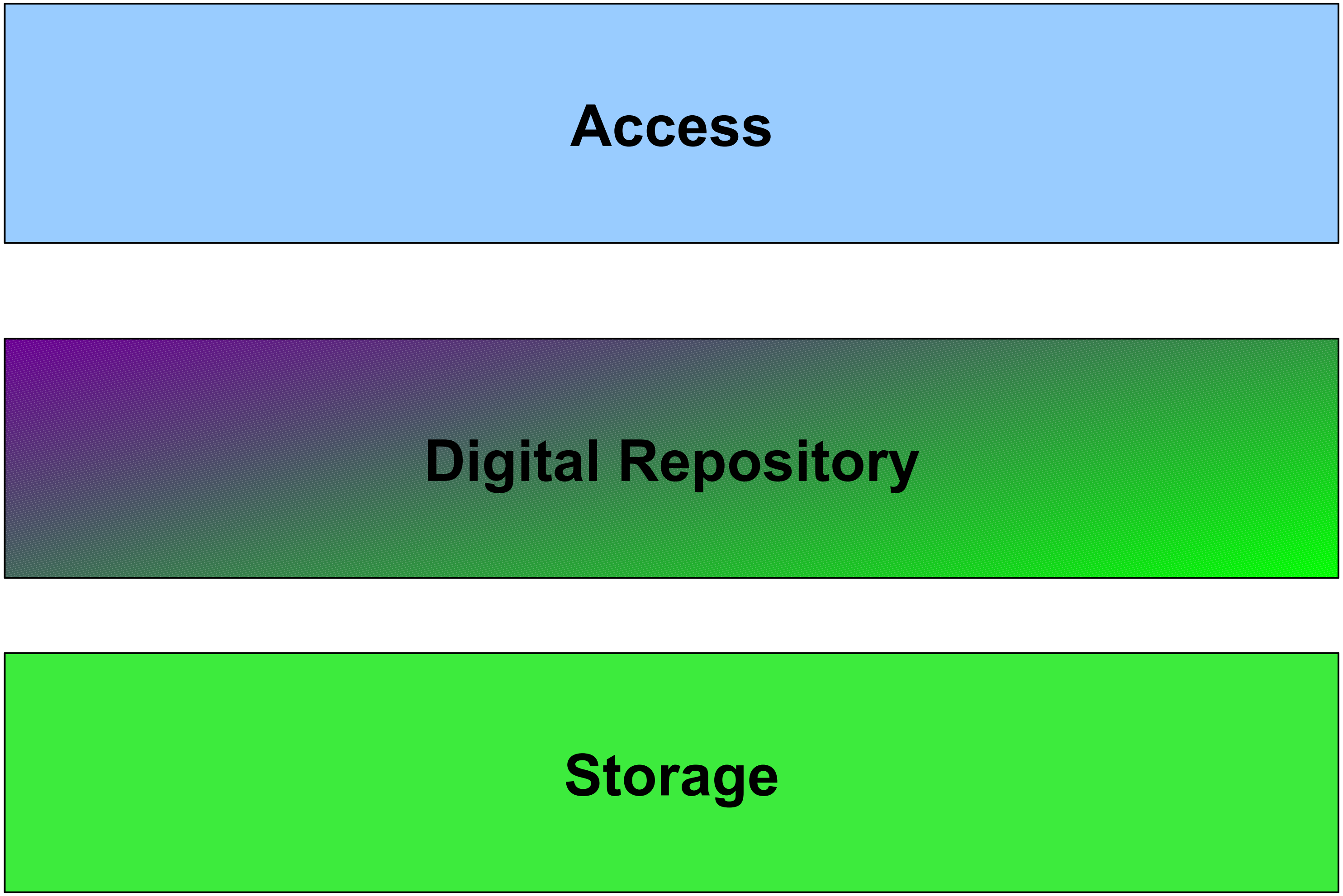
- Support for user-defined metadata
  - Useful for adding project-specific metadata
  - In triplets (attribute, value, unit).
  - Support schemas such as Dublin Core, FITS, DICOM.
  - Rules can extract metadata stored in XML files and populate user-defined metadata.

IRODS

- Provides features essential for Repositories:
    - Storage virtualization
    - Data location virtualization
    - Policy virtualization
- Features provide a flexible, scalable system that is robust to change.
- All operations carried out by micro-services on objects in iRODS.

Where iRODS fits?

Client interacts with digital repository to access data



Access

Digital Repository

Storage

IRODS Spans Digital Repository and Storage Domains

Where iRODS fits?

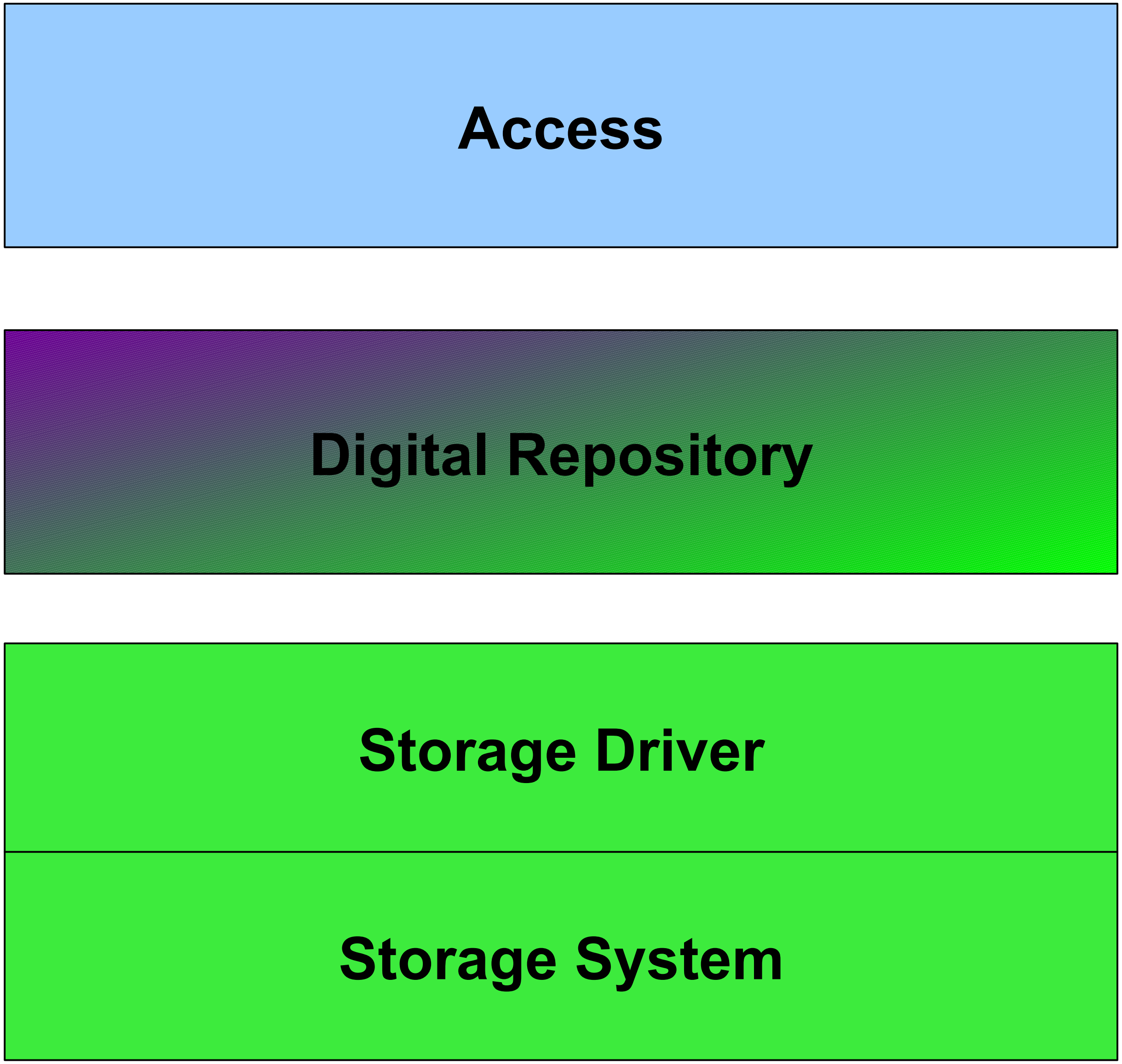
- IRODS provides infrastructure to manage data.
- Policies implemented as computer actionable rules which control the execution of remote micro-services.
- Micro-services interact with data.
- Covers the Storage Management and Storage part of the digital repository.

Storage Virtualization

- Problem: over time storage will change (e.g. new HSM, new tape systems, etc).
- Solution: insulate repository from changes through interface/driver.
- IRODS provides drivers to storage that expose POSIX standard API.
- Interaction with data performed by micro-services that communicate with data through the drivers.

Storage Virtualization

Access to Storage through driver.
Provides POSIX standard interface.



Access

Digital Repository

Storage Driver

Storage System

Storage Virtualization

- Also want to be insulated from changes to storage name/address.
  - Provide logical storage resource name.
  - Logical-to-physical resource name mapping.
  - All iRODS interactions with Storage use logical name.

Data Location Virtualization

- Problem: physical structure of digital objects on storage may change in the future.
- Solution: create logical file-path to insulate from changes to physical path.
- IRODS provides logical-to-physical mapping to insulate from changes.
- All iRODS interactions use the logical name.

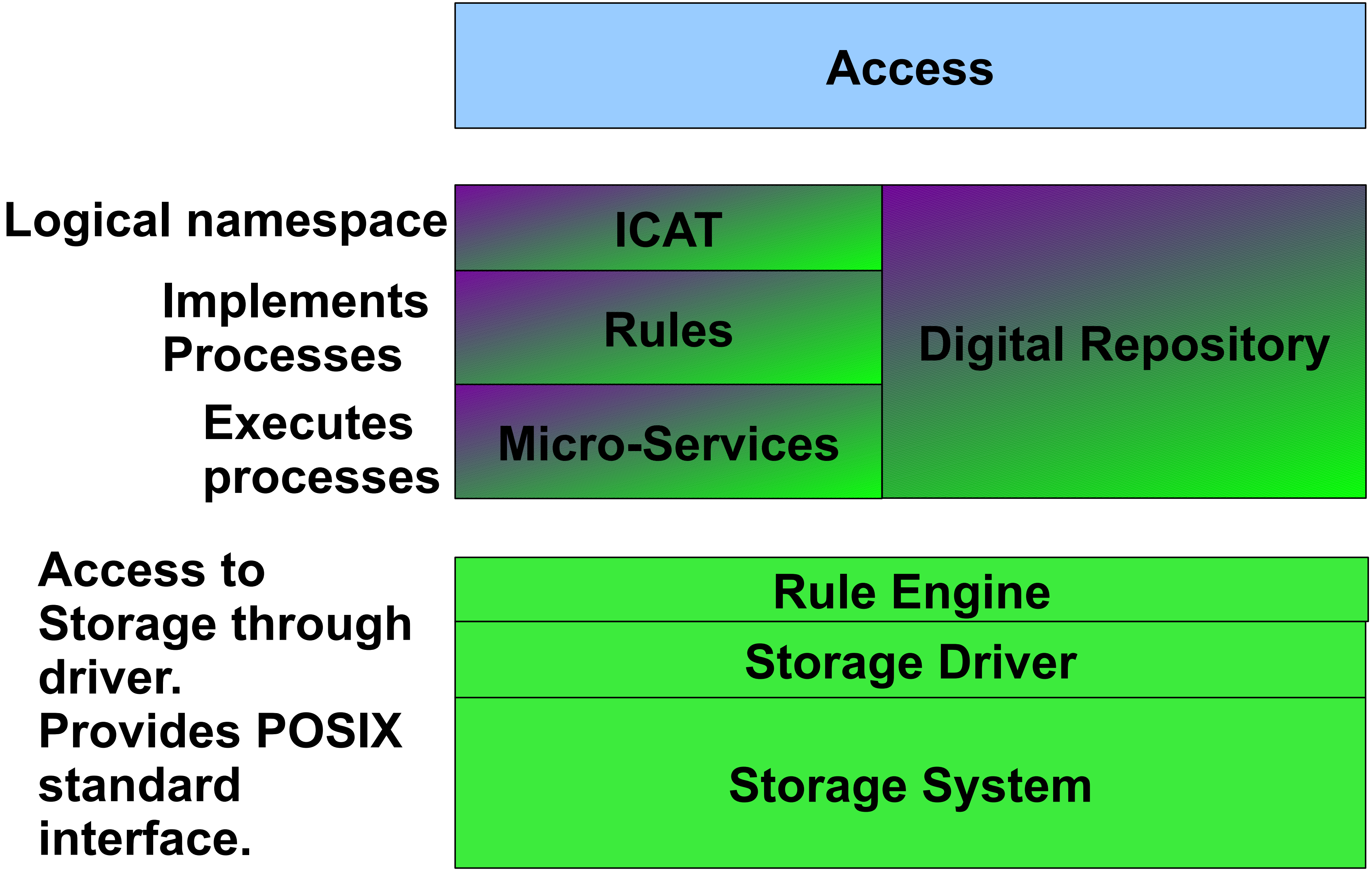
Data Virtualization

- Can group data objects into logical collections.
- Logical collections can span multiple resources.
- Can create a logical collection that spans zones.
- Can register events against collections.
  - Notified when data is updated/moved/etc.

Policy Virtualization

- Problem: management applications may change over time.
- Solution: abstract processes such that it is possible to replace processes without altering workflow.
  - IRODS encodes process as a micro-service (C-application)
  - Create workflow by compositing multiple micro-services.
- Identify locations in data management framework where policies should be checked.
  - Specify a rule that is checked on each invocation of a framework management hook.
  - Support pre-process hooks for authorization
  - Support post-process hooks for audit trails

Storage Virtualization



Rules

- Policies are implemented in iRODS as rules.
- Rule is a series of logically connected steps.
- Each step realised as a micro-service.
- IRODS rules fully featured:
    - Contain loops and branches.
    - Can have rules contained within rules.
- IRODS rules read from a rule-file (called core.irb by default).

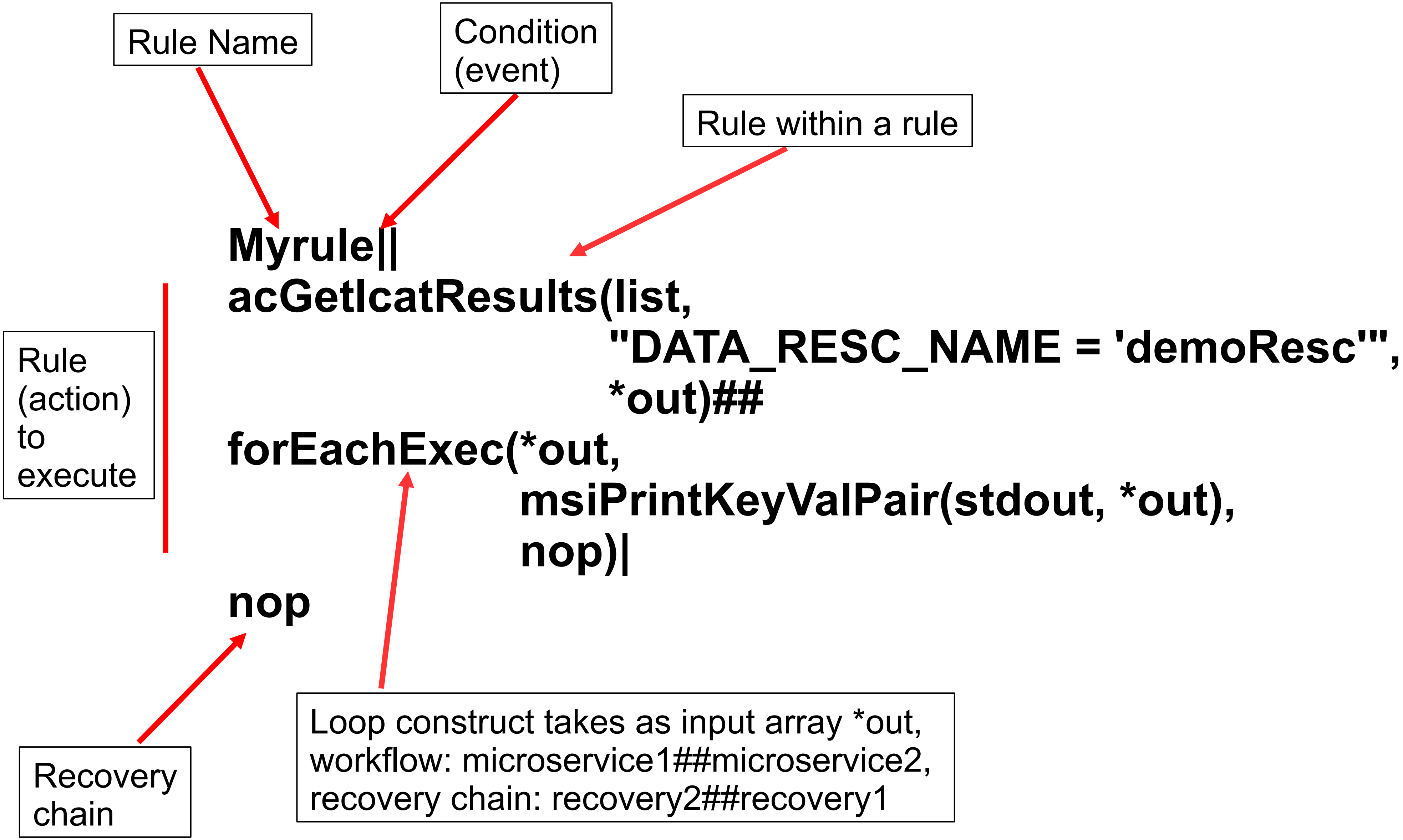
Rules

- Rule follows Event-Action-Recovery chain.
- Event, Action, Recovery domains separated by '||'.
- Rule executed from left-to-right.
- All micro-services in a rule separated by '##'.
- Each action micro-service must have a recovery (even if it's a nop).
- Input and output variables start with '*'.

Example Rule

- Look at an example rule:
- Rule to query the catalogue to find and print all data objects that are on the demoResc resource.
- Make use of the core.irb rule acGetIcatResults to return list of results.
- Use ForEachExec loop to loop over results and print the values.

Example Rule



```
Myrule||
acGetIcatResults(list,
                          "DATA_RESC_NAME = 'demoResc'",
                          *out)##
forEachExec(*out,
                    msiPrintKeyValPair(stdout, *out),
                    nop)|
nop
```

Rule Name

Condition (event)

Rule within a rule

Rule (action) to execute

Recovery chain

Loop construct takes as input array *out, workflow: microservice1##microservice2, recovery chain: recovery2##recovery1

Rules

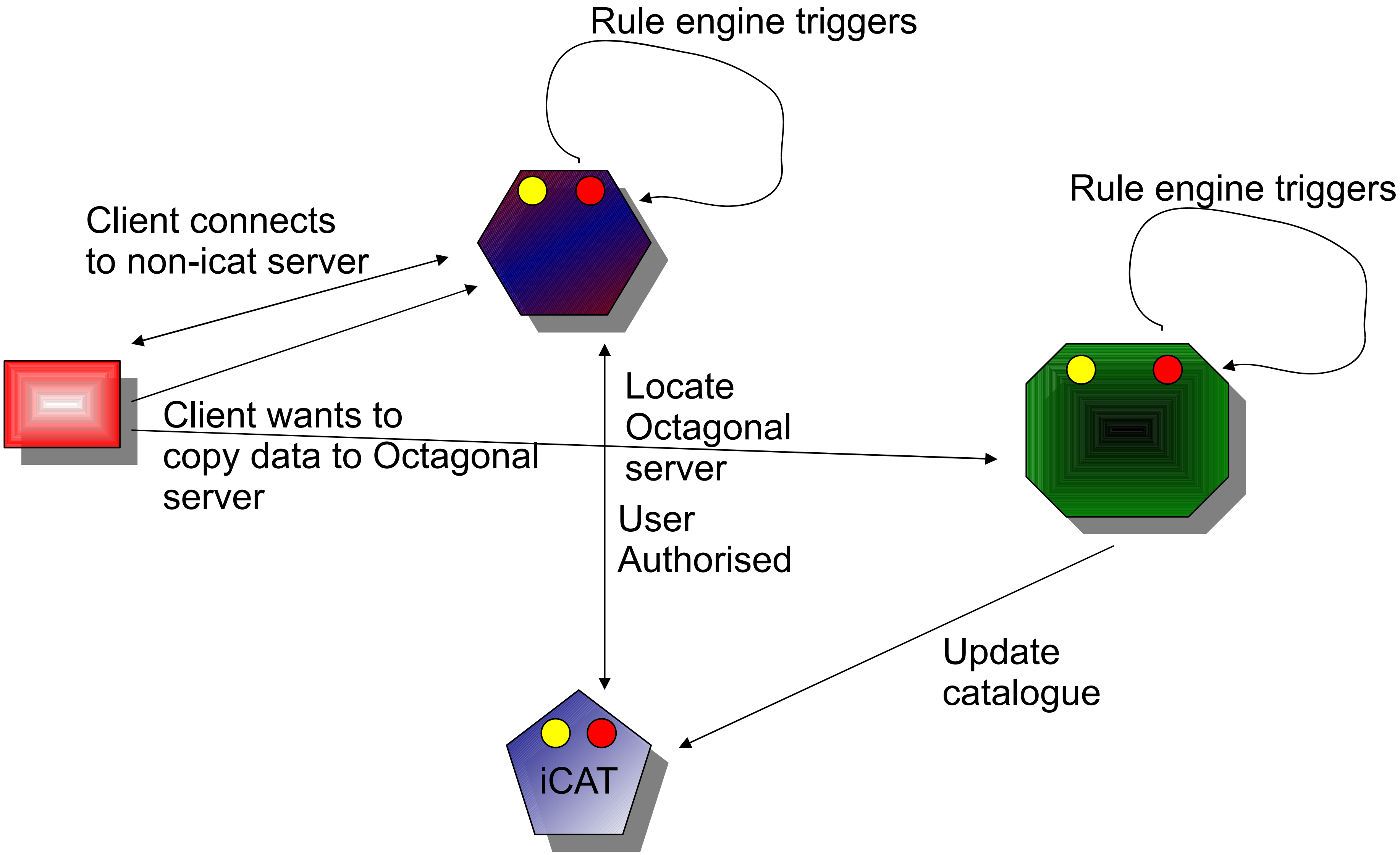
- Rules stored in a rule file (default is core.irb).
- Rules read from file top-to-bottom.
- First rule that satisfies event is executed.
- Only one successfully executed rule per event.
- Can override a rule, but overridden rule must appear later in the rule file.

Rule Engine

- Rules in rule file executed by the rule engine.
- Engine running on each iRODS resource.
- Rule engine triggered by any interaction with the iRODS server (copy, put, get, etc).
- *Except* for queries of the catalogue (listing).
  - Mainly due to performance reasons.
  - But can be overridden.
- Rule engine on server client connects to runs by default.

Rule Engine

- Also delayed execution rules supported.
  - Can execute a rule later.
  - Can execute a rule periodically.
- Delayed execution rules are run stored in catalogue.
- By default rule engine polls for delayed execution rules every 30secs.
- Can direct the rule engine closest to data to execute.



Rule engine triggers

Client connects
to non-icat server

Rule engine triggers

Locate
Octagonal
server

Client wants to
copy data to Octagonal
server

User
Authorised

Update
catalogue

iCAT

Repositories Requirements

- To reliably store data for a defined period of time.
  - Allows rules to be placed on data (replication, checksums).
- To ensure data are accessible.
  - Rules to migrate data.
- To validate repository assessment criteria
  - Rules to parse audit trail, verify integrity, verify retention and disposition

Federation

- Union of independently administered repositories.
- Useful for:
    - Interoperation with other remote repositories that are independently administered.
    - Access to data in different repositories in seamless manner.
    - High-availability system.

Federation Issues

- Rights to access remote repository data (all, some).
- Rights to store data in remote repository.
- Rights to access applications from remote repository.
- Rights to store applications in remote repository.

IRODS Federation

- IRODS system consists of one iCAT and 0 or more storage systems (each with its own iRODS server and rule engine).
- Each iRODS system has its own name-space called a *Zone.*
- IRODS allows interoperation between Zones (Federation).
- IRODS federation at the storage management level.

IRODS Federation

- Creation of iRODS federation essentially:
  - Register zones.
  - Register users as remote users.
  - Grant access to data to remote-zone users.
- Remote user has access to local user data.
- Users authenticated locally then given remote access.
- Currently any interaction (except 'ls') will cause rule to trigger on remotely accessed data.

Resources 'federation'

- Useful if just want to interoperate storage repositories.
- Each repository part of iRODS system.
- Only one zone needed.
- Each site manages its own resource.
- But, each site needs admin privs to manage its resource.

Zone Federation

- Useful when 'organizations' wish to interoperate iRODS systems.
- Each Zone controls it's own storage and data.
- A Zone may house data that's part of more than one repository.
- Can more easily add new resources as opposed to 'resource federation'.

Producer-Reader Federation

- Effectively have one system that is filled with all the data (the Producer Zone).
- Reader Zones consist of just iCAT server.
- Replicate data of interest to Reader Zones.
- Useful for high-read rate systems where readers are not interested in cross-zone collections.

Hub-Spoke Federation

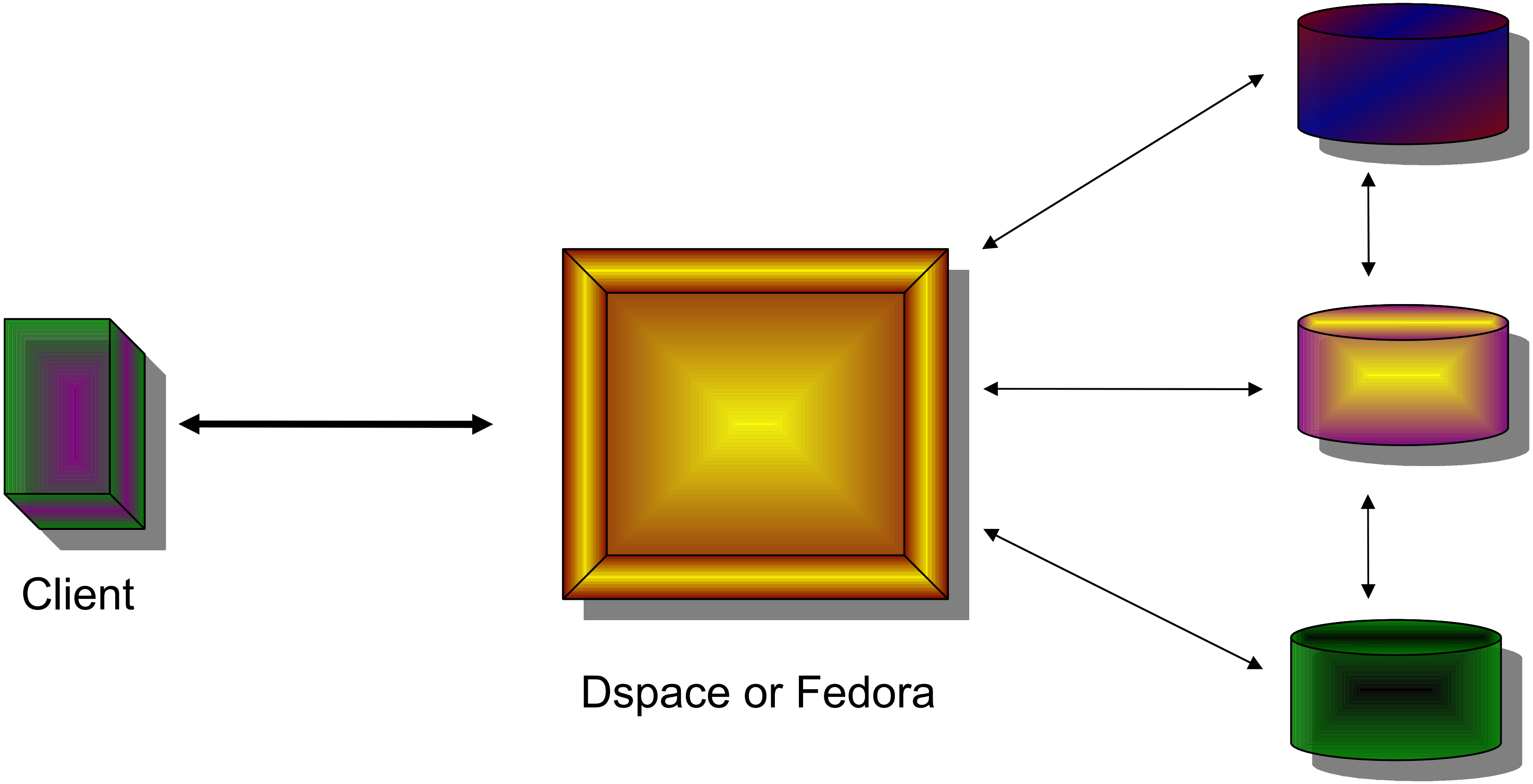
- Useful where there may be many producers and readers.
- Centralized model.
- Each iRODS Zone contains readers and writers.
- Central Zone has all Zones registered.
- Users access central and access data from remote Zones.

Repository Interoperation

- Federation very useful across iRODS systems.
- In the case of wishing to interoperate with a different type of system:
  - Look at writing an iRODS driver that knows how to talk to the system so it can appear as an iRODS resource. Need an iRODS server running on resource.
  - Look at writing an iRODS micro-service that can interact with the system. No iRODS server needed, but thought required about workflow.

IRODS and Fedora and Dspace

- Projects currently looking at integration of Fedora repository framework with iRODS manage the back-end storage.
  - D-Grid, DARIAH
  - Duke Medical Archives
  - Carolina Digital Repository
- DICE provides an interface to Dspace to allow iRODS to be used as managed storage.



Client

Dspace or Fedora

IRODS System

Dspace and Fedora use iRODS as distributed
Back-end storage. Discovery and indexing handled
by other tools.

Digital Archive

- The SHAMAN (Sustaining HeritAge through Multivalent ArchiviNg) project
- Looking at digital preservation.
- FP7 integrated project funded until end of 2011.
- 17 EU partners. 2 US collaborators.
- Partners from academia and industry.
- Aim to provide a digital preservation framework.

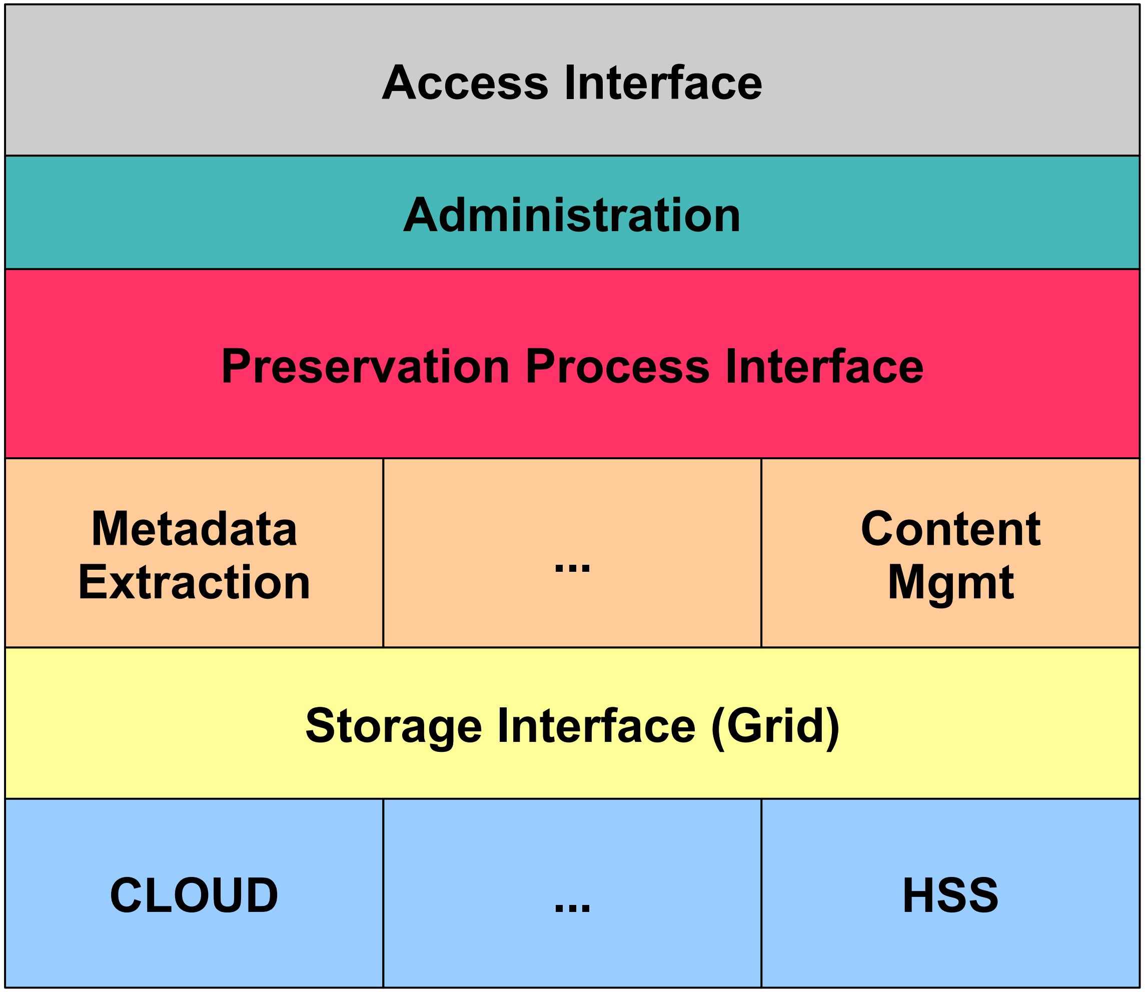
SHAMAN

- Looking to describe the preservation environment sufficiently well.
- Such that it's possible to replace services without impacting preservation of the data.
- In addition looking at the use of Multivalent technology to 'render' the object stored in the original format.
- Multivalent Java-based 'render' tool has adaptors (media engines) capable of reading different formats.

SHAMAN

- Make use of Cheshire Digital Library tool-kit to index data.
- Make use of iRODS to provide a means of abstracting the preservation process and providing underlying storage.

SHAMAN



Access Interface

Administration

Preservation Process Interface

Metadata Extraction

...

Content Mgmt

Storage Interface (Grid)

CLOUD

...

HSS

Provides uniform access to data.

Interface provides uniform access to different preservation processes.

Grid interfaces to different Types of storage. Provides Uniform Interface

SHAMAN

To ensure data usable in the long-term:

- Insulate from hardware changes.
- Insulate from changes to processes.
- Insulate from changes to data format.
- Insulate from changes to description.
- Ensure as much information as possible about data is captured.
  - Ideally test data is understandable without ANY external dependencies.

SHAMAN aims to provide a framework that accounts for these issues.

Summary

- IRODS can provide a basis for which digital repositories or archives can be constructed.
- Have illustrated some of the features of iRODS.
- Have illustrated how some of these features can be of use in repositories and archives.
- Have illustrated how iRODS can interoperate with existing systems.

Acknowledgements

- Thanks to:
  - Reagan Moore
  - Arcot Rajasekar (Raja)
  - Mike Wan
  - Wayne Schroeder
  - Paul Watry
  - Jean-Yves Nief