CLARIN D
Common Language Resources and Technology Infrastructure

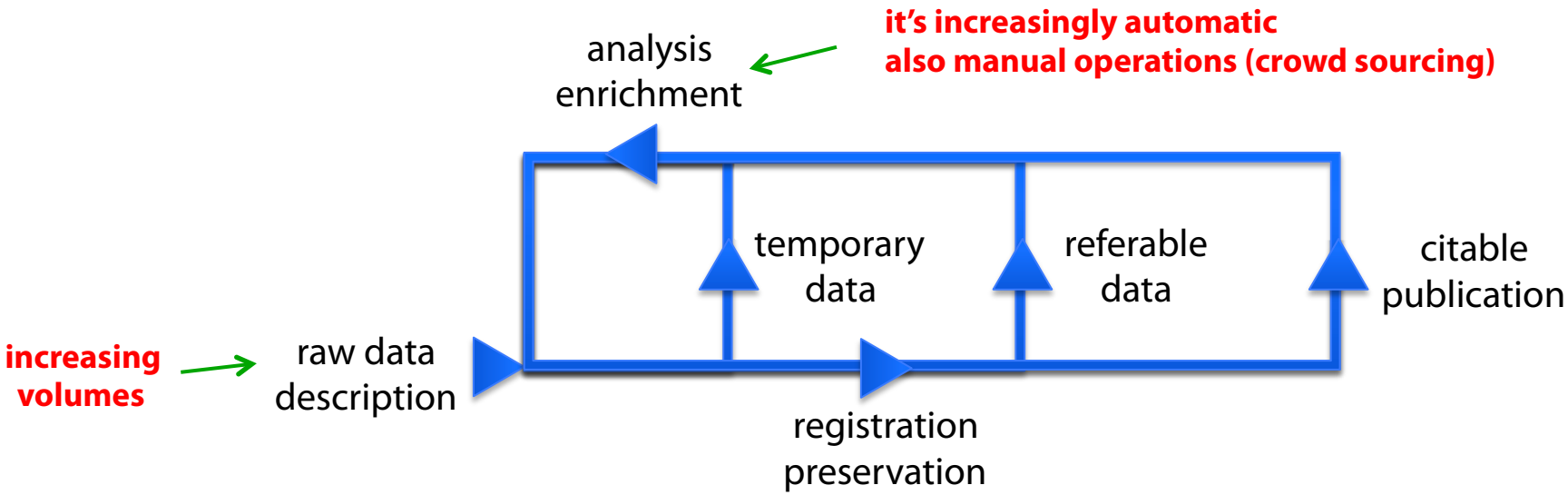
Repositories - an overview

Peter Wittenburg
The Language Archive
MPI für Psycholinguistik

verschiedene Erwartungen

- Changing data domain
- What is a repository and requirements
- What does it store and offer
- Trust, persistence, mass of data, policy rules
- LAMUS a short analysis
- What kinds of solutions are around
- What are the costs
- Relevant standards

Changing data domain



- need to anticipate a highly dynamic data world where data needs to be referable and citable (re-usability, re-purposing)
- need to anticipate increasing volumes and complexity

What is a repository?

- Bob Kahn: “repository is a network accessible storage to store objects for later access”
- JISC: A digital repository is a managed, persistent way of making research, learning and teaching content with continuing value discoverable and accessible. Repositories can be subject or institutional in their focus. Putting content into an institutional repository enables staff and institutions to manage and preserve it, and therefore derive maximum value from it. A repository can support research, learning, and administrative processes.
- Forrester Research: Knowledge workers spend 40% of their time trying to find information and 70% of that time is spent recreating information that cannot be found. A digital repository offering refined categorisation and search tools that help locate information quickly provides quantifiable savings in terms of time and resources.

ESFRI Requirements

ESFRI (European Strategy Forum on Research Infrastructures)

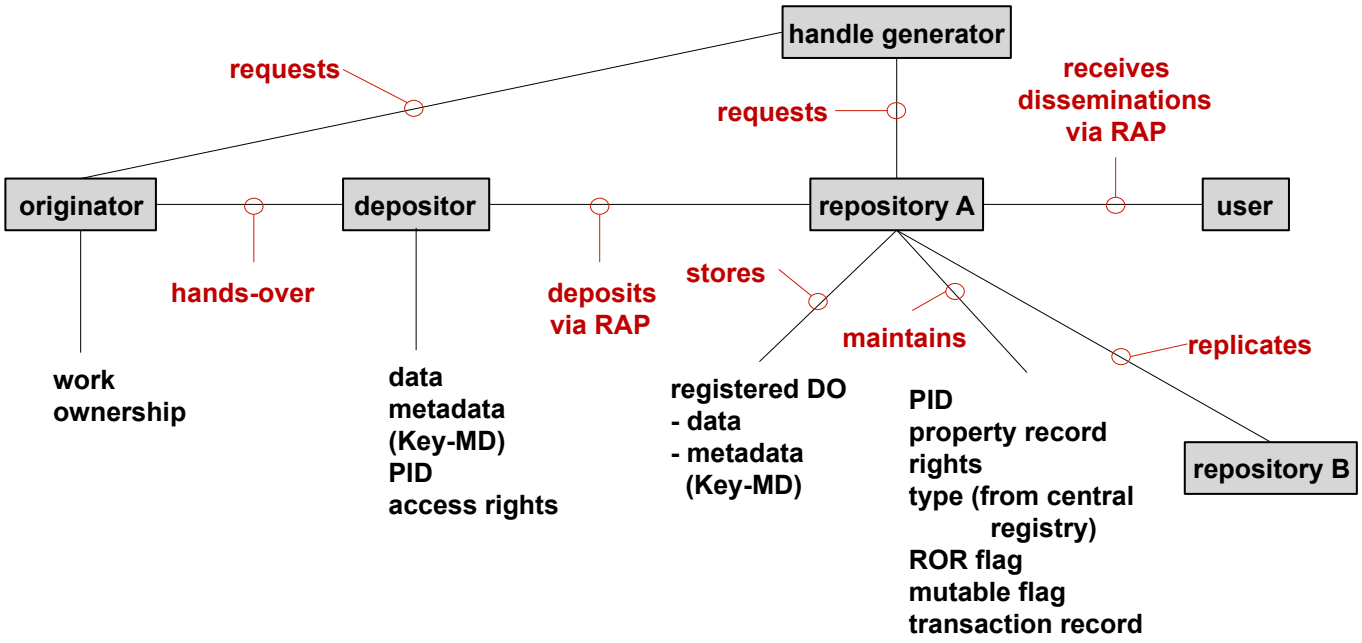
- WG established criteria for digital repositories
  - Availability: data and metadata must be available
  - Permanency: preservation, management and curation
  - Quality: policy for data quality
  - Rights of Use: clear statements about accessibility
  - Interoperability: support of open standards

Requirements

- A repository is a structured container that
  - stores large volumes of data objects and collections
  - allows users to upload objects and collections
  - carries out checks at upload, requires metadata descriptions and associates a PID at upload time (OAIS - SIP?)
  - allows managers to carry out data lifecycle management including long-term archiving
  - supports persistence of objects and collections, preserves their integrity and authenticity (versioning, presentations, etc)
  - carries out curation and maintains provenance
  - allows users to access objects and collections supporting access restrictions (must be easy!!!!!)
  - performs regular quality assessments
- if the user sends a PID the repository needs to provide same bit-stream even after many years

What does it store and offer?

Kahn's Digital Object Architecture (95, 2006)



digital object (DO) = instance of an abstract data type with 2 components (typed data + metadata including a Handle); can be elementary and composed; registered DOs are such DOs with a Handle;

RAP (Rep access protocol) = simple access protocol with minimal functionality required for DOA;

Dissemination = is the data stream a user receives upon request via RAP

ROR (repository of record) = the repository where data was stored first; controls replication process

Meta-Objects (MO) = are objects that store mainly references (collection descriptions)

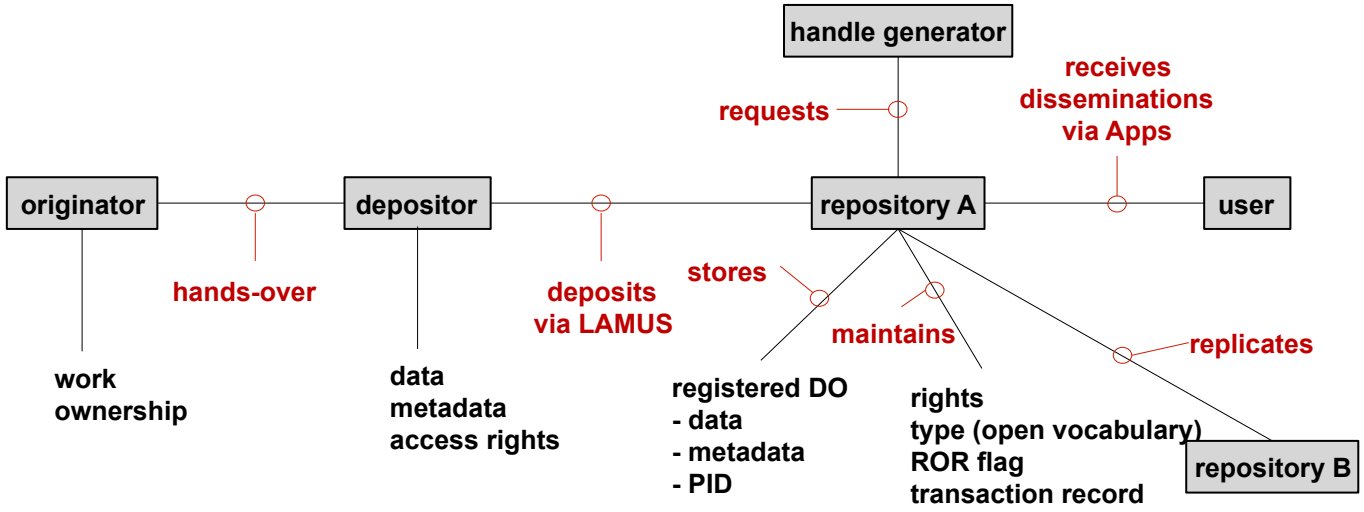
mutable DOs = some DOs can be modified, others not

property record = contains various info about DO (metadata, etc)

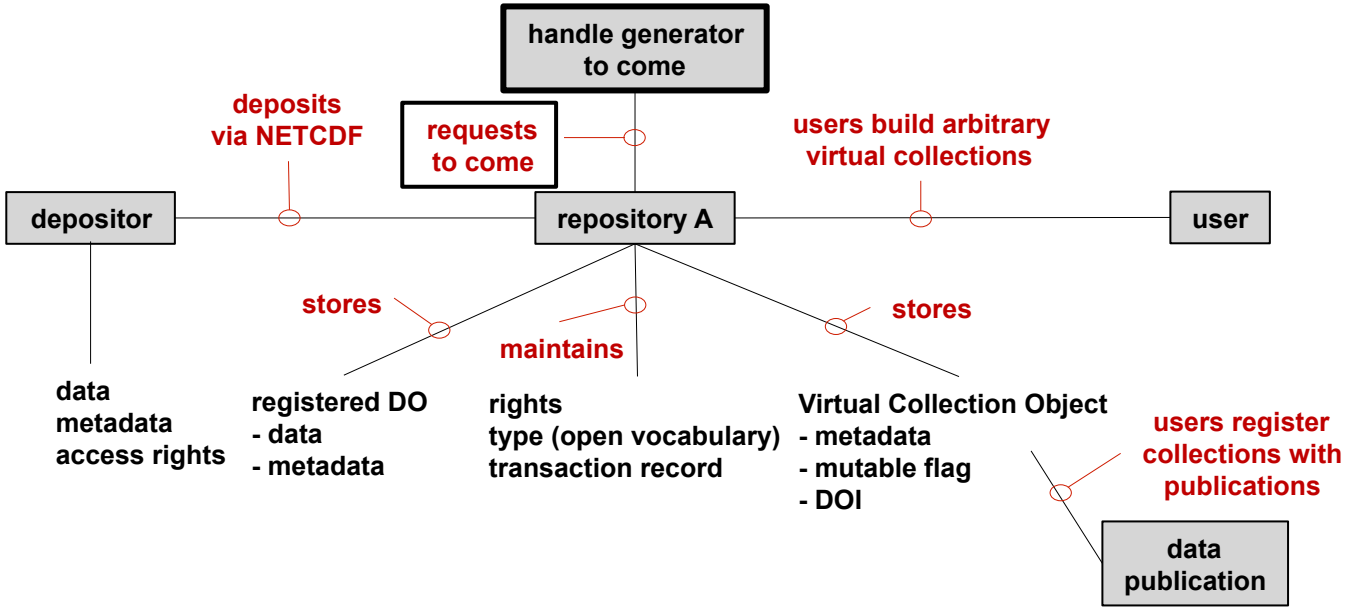
type = data of DOs have a type

What does it store and offer?

DOBES Object Architecture (2002)



Climate Res. Object Architecture (2006)



What does it store and offer?

Planets categories:	DOA:
• 1 object = 1 file = 1 format	1 metadata desc., 1 PID, simple type
• 1 object = 1 file = m formats	1 metadata description, 1 PID, complex type
• 1 object = n files = 1 format	collection of simple objects
• 1 object = n files = m formats	collection of complex objects

- type checks are essential for lifecycle management (curation)
- needs to be done automatically (therefore JHOVE and other libraries)
- important is provenance information in metadata description

- composite objects require encapsulation (see PDF)
- data encapsulation is a nightmare
- databases (XML, relational) are composite structures and do encapsulation
  - DB require application logic to fulfill requirements (who will maintain it?)

What about relations?

- data objects have many external relations
  - audios, videos, annotations of the same recording event
  - texts from the same newspaper issue
  - recordings from the same trip
  - all videos of a certain format
  - etc.

these are all collection types

- each object can be part of many collections (re-usability, re-purposing)
- how to store relations and why
  - associate categories to easily filter/search (metadata)
  - explicit organization for management (canonical + other collection)
  - explicit relations as assertions (external RDF database)
  - GIS path (external view)
  - Conceptual Spaces (external view)
  - etc.

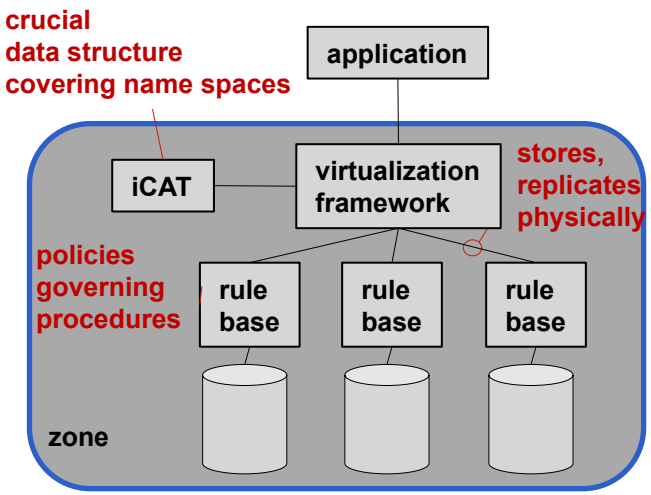
meta data

Trust, Persistence, Mass of Data

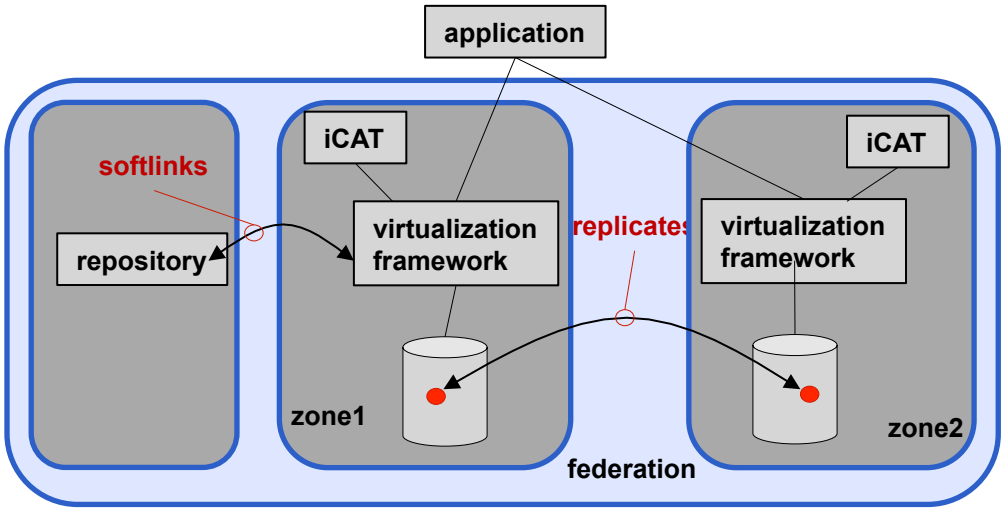
- Repository implies trust relations:
  - depositor: that rep takes care of life-cycle management, persistent data access respecting restrictions
  - user: data object is what it claims to be (identity, authenticity), it's there independent of time, context and provenance information is available for interpretation
  - in an anonymous Internet scenario quality assessments are required to establish/maintain trust
- let's not speak about repositories when there is no clear persistence offer - make your policies explicit
- there is no need for a repository if you just have a few objects
- mass of data require automatic procedures for management and curation - how to check state and quality

Policy rule based operation

Grid Architecture - Within Zone



Grid Architecture - Across Zones



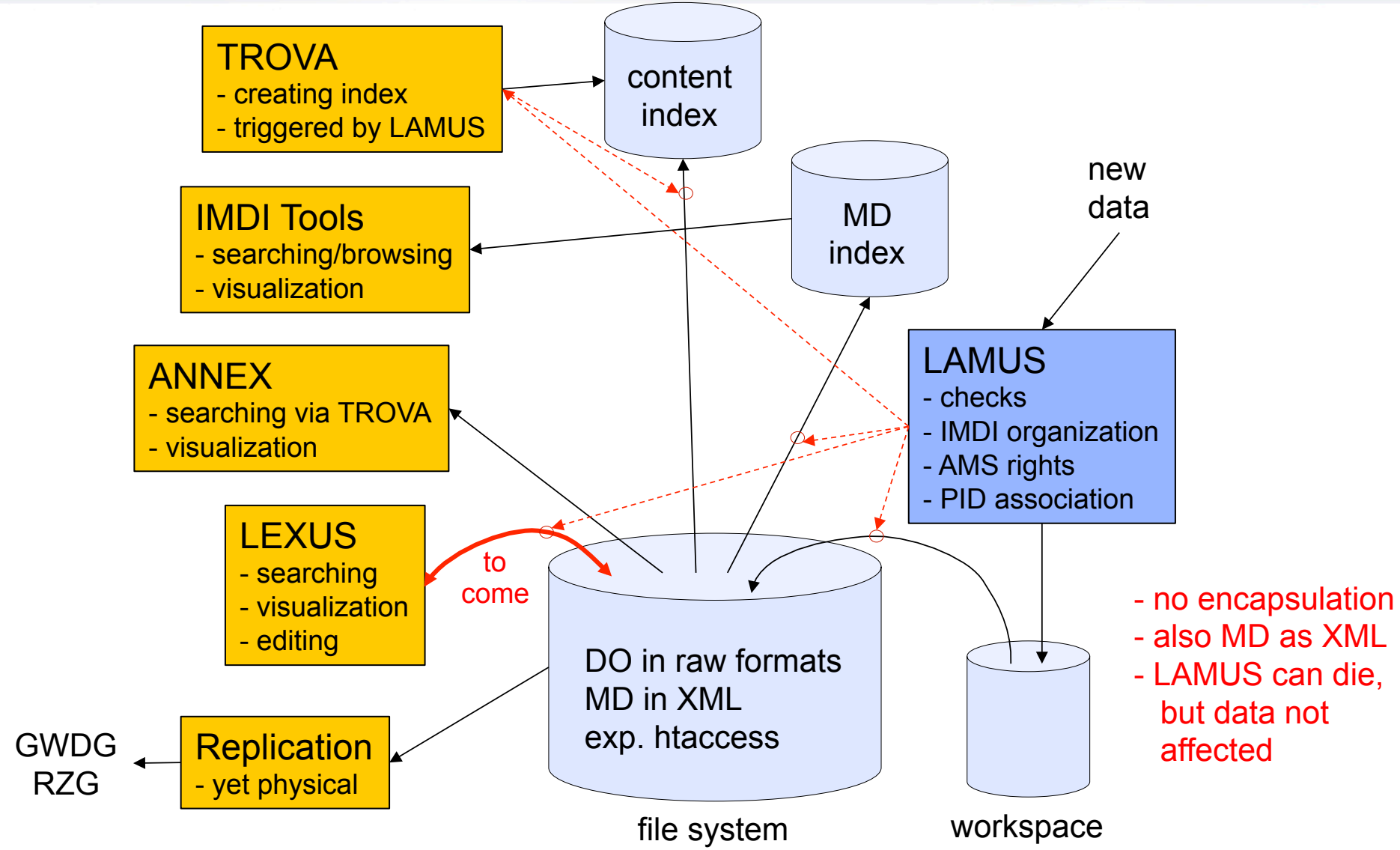
Definitions/Entities

- central is a data structure called iCAT which manages state information relevant for the virtualization framework
- iCAT is meant to handle persistent name spaces for known data objects in a zone, persons accessing objects in a zone, storage resources in a zone and policies controlling collection properties; due to virtualization all is dynamic including the storage resources
- the virtualization framework can store and execute rules in a local rule base for enforcing collection management policies; rules are composed by chaining a sequence of micro-services into an executable workflow; example rules include: AIP definition, retention, disposition, distribution, replication
- the execution of the micro-services can be carried out based on additional federation policies
- soft links can be created for registering information and digital objects from non-iRODS managed repositories into a local collection

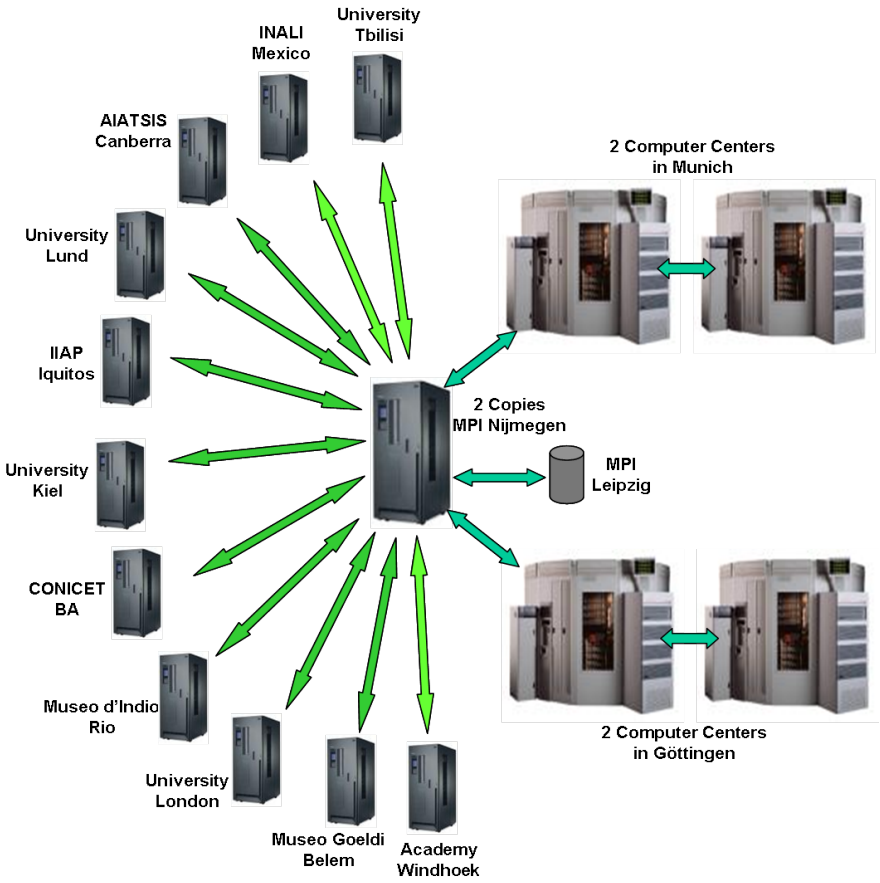
Characteristics of implementation (iRODS)

- the notion of policies that are turned into executable rules is a very attractive concept in particular with respect to quality assessment of repositories and controlling creation of collections that span multiple data grids and storage repositories
- iCAT is designed to define and control the namespaces within the grid; externally defined namespaces can be registered as attributes on each file, enabling identification by PIDs or iRODS logical name, or values of descriptive information
- in many cases institutions and communities involved in CDI have already built data infrastructures that address virtualization, metadata, etc.; for these communities iRODS can be used to build a collection that spans multiple repositories, and enforce community policies across the federated systems
- important namespaces should be maintained outside of specific software components to not create dependencies
- thus with iCAT iRODS comes with an integrated solution for name spaces, while SOA starts from independent components

LAMUS - a short analysis



LAMUS - short analysis



- at MPI start in 2000 as bottom-up process
  - repository has ~ 100.000 lines of code
  - utilization software has much more lines of code
  - rep. system now used by about 13 institutes
  - sw maintenance costs for repository system about 60k€/y
  - is it persistent?
  - is this relevant (no encapsulation)?

- eScidoc has more than 1 Mio lines of code
- much higher maintenance costs
- is it persistent - is it stable?

What kinds of solutions are around?

- ready made solutions (but take care - often no application logic)
  - D-SPACE: evolved in the domain of libraries as a repository of static documents; lots of developments to adapt it to dynamic data world; used worldwide
  - ePrints: same; mainly used in UK
  - eSciDoc: development for dynamic research data based on Fedora; comes with layered APIs to quickly develop applications; extensive code set; creators: MPDL, FIZ
  - LAMUS: specialized on language resources based on file system; comparatively simple code base fulfilling all CLARIN requirements; comes along with a number of applications; not tuned to large text sets
- tool kits
  - Fedora: object store library; used worldwide mainly to store metadata

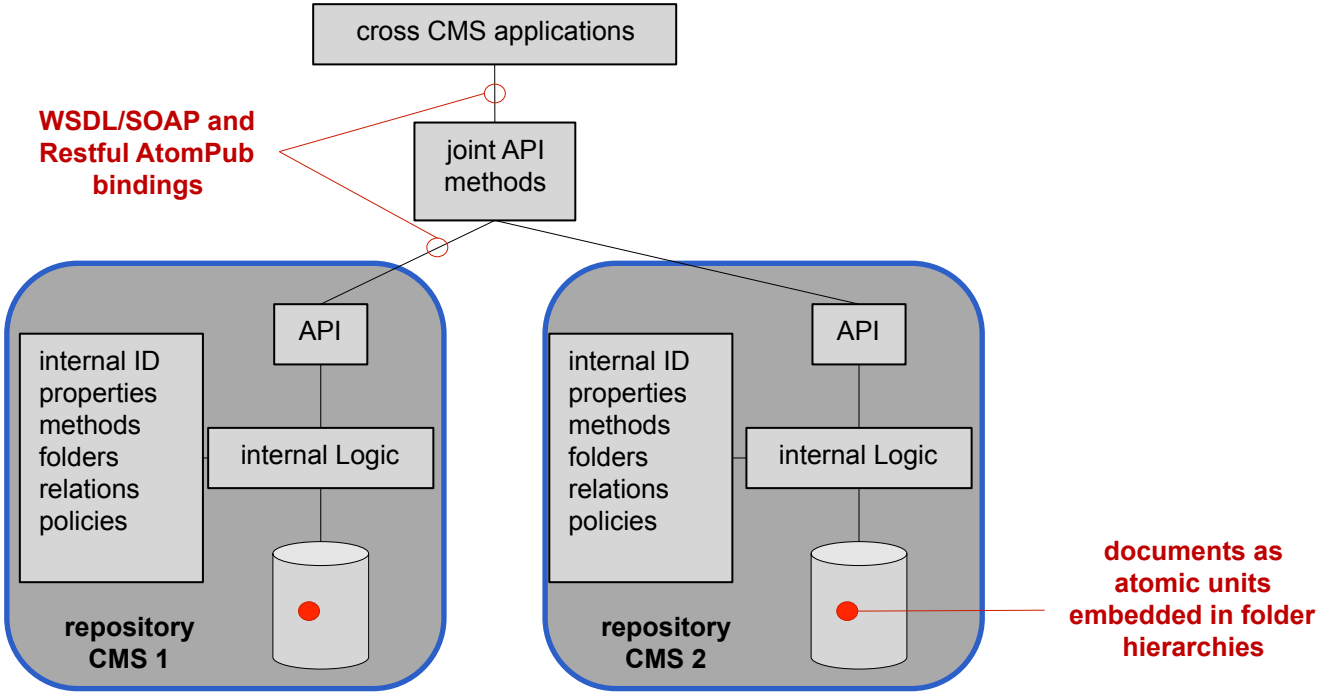
What kinds of solutions are around?

- grid and database solutions
  - iRODS: not really a repository system, but useful to federate repositories (earlier SRB solution most widely used system)
  - mySQL, Postgres, etc: powerful rDBMS etc; encapsulation, different objects in one big pot incl. metadata; complex application logic required; often used for metadata
  - xBase, eXIST etc: powerful XML-DB; encapsulation, different objects in one big pot incl. metadata; complex application logic required; often used for metadata

- commercial solutions
  - ORACLE etc: powerful rDBMS with excellent application builders etc; encapsulation, commercial dependence
  - CMS: a large variety of Content Management Systems mostly coming with own ideas about metadata etc., commercial dependence

CMIS as example of closed world

CMIS Model (OASIS)



Costs - MPI

type	k€/y	comment
basic IT infrastructure	80	4-8 years innovation cycle
digitization and workflow	10	new recorders, capturing dev
copies at large computer centers	<5	
system management	60	shared for different activities
archive management	80	advice, curation, consistency
repository software maintenance	60	without new functionality
utilization software maintenance	>120	wide spectrum of tools
building, energy, etc	?	ignored here
total	415	

- economy of scale applicable, currently ~80 TB
  (linguistic support, SW development, head etc. not calculated)
- management more expensive than rep SW maintenance

Costs - Beagrie

Institutional Repository (e-publications):	Staff	Equipment (capital depreciated over 3 years)
Annual recurrent costs	1 FTE	£1,300 pa

Federated Institutional Repository (data): Annual recurrent costs	Staff	Equipment (capital depreciated over 3 years)
Cambridge	4 FTE	£58,764 pa
KCL	2.5 FTE	£27,546 pa

Costs - Beagrie

Acquisition and Ingest	Archival Storage and Preservation	Access
c. 42%	c. 23%	c. 35%

Relevant Standards

- OAIS Reference Model

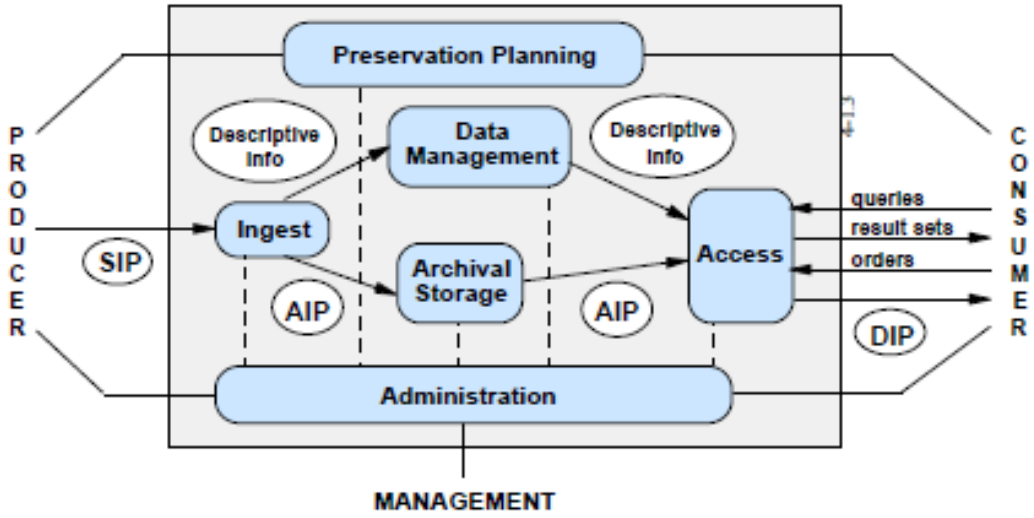


Figure 4-1: OAIS Functional Entities

- RAC (Rep. Audit and Certification) http://cwe.ccsds.org/moims/
- DSA (Data Seal of Approval) http://www.datasealofapproval.org/
- OAI-PMH (Prot. for Metadata Harvesting) http://www.openarchives.org/OAI/openarchivesprotocol.html

Summary

- proper repository is
    1. a matter of good persons, stable environment and proper organization
    2. a matter of a reliable storage system
    3. a matter of proper software
- building trust is essential - trust is a result of years/decades/...
- software choice
    - what are your objects - look for similar and PROVEN installations
    - go a pragmatic way and prevent too much own development
    - application development much more costly than core rep system
    - prevent encapsulation to be able to change software
- procedure
    - visit some institutes with experience

Thanks for the attention.

Characteristics

	long-term	accessibility	sharing	trust	costs
private disk	low	high	low	high	?
institutional rep	low	high	moderate	moderate	?
organization rep	high	moderate	high	?	?
community rep	?	moderate	high	?	?
commercial rep	low	?	high	low	?

- is it all new?
- AMOS (Advanced Multi-User Operating System, Friedrich Hertweck, 70 ies, IFIP award) to manage and access large amounts of fusion data