Workshop C1: Stand Fedora 6 und Linked Data

Andreas Nef, Thomas Bernhart, Olten FHNW 30.10.2019

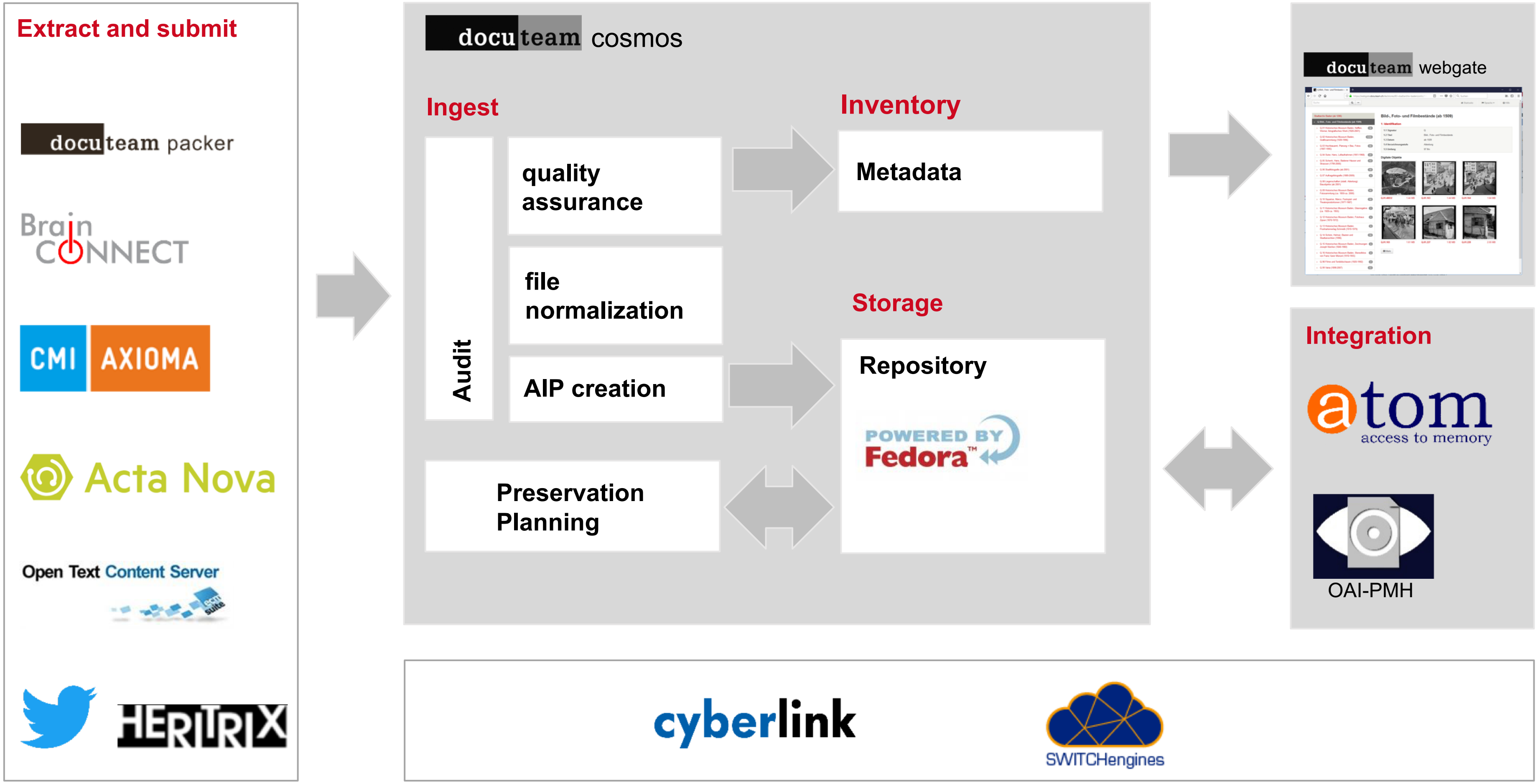
Program

- Introduction, History of Fedora Commons
- Data model, Matterhorn RDF
- Fedora 6, Oxford Common File Layout

Introduction, History of Fedora Commons

docuteam cosmos: overview

docuteam



Extract and submit

docuteam packer

Brain CONNECT

CMI AXIOMA

Acta Nova

Open Text Content Server

HERITRIX

docuteam cosmos

Ingest

Audit

quality assurance

file normalization

AIP creation

Preservation Planning

Inventory

Metadata

Storage

Repository

POWERED BY Fedora™

docuteam webgate

Integration

atom access to memory

OAI-PMH

cyberlink

SWITCHengines

What Fedora is (not)

- **Fedora is the flexible, modular, open source repository platform**
  - **since version 4: ...with native linked data support.**

- What it is:
  - adding functionality and semantic to the storage layer
  - supporting concepts to model simple to complex objects in a flexible way
  - offering standardized and controlled access to the objects stored
  - solid base for complex digital resources platforms

- What it is not:
  - an end-user platform with a flashy user interface
  - an out-of-the-box digital reading room
  - an ingest workflow engine

What's wrong with Fedora 3?

- Good experiences with Fedora 3
  - Stable and reliable product used for ten years
  - Active/international community
  - Focus on core functionality
  - Platform independent, flexible infrastructure requirements
  - Good support of community standards (XML based)

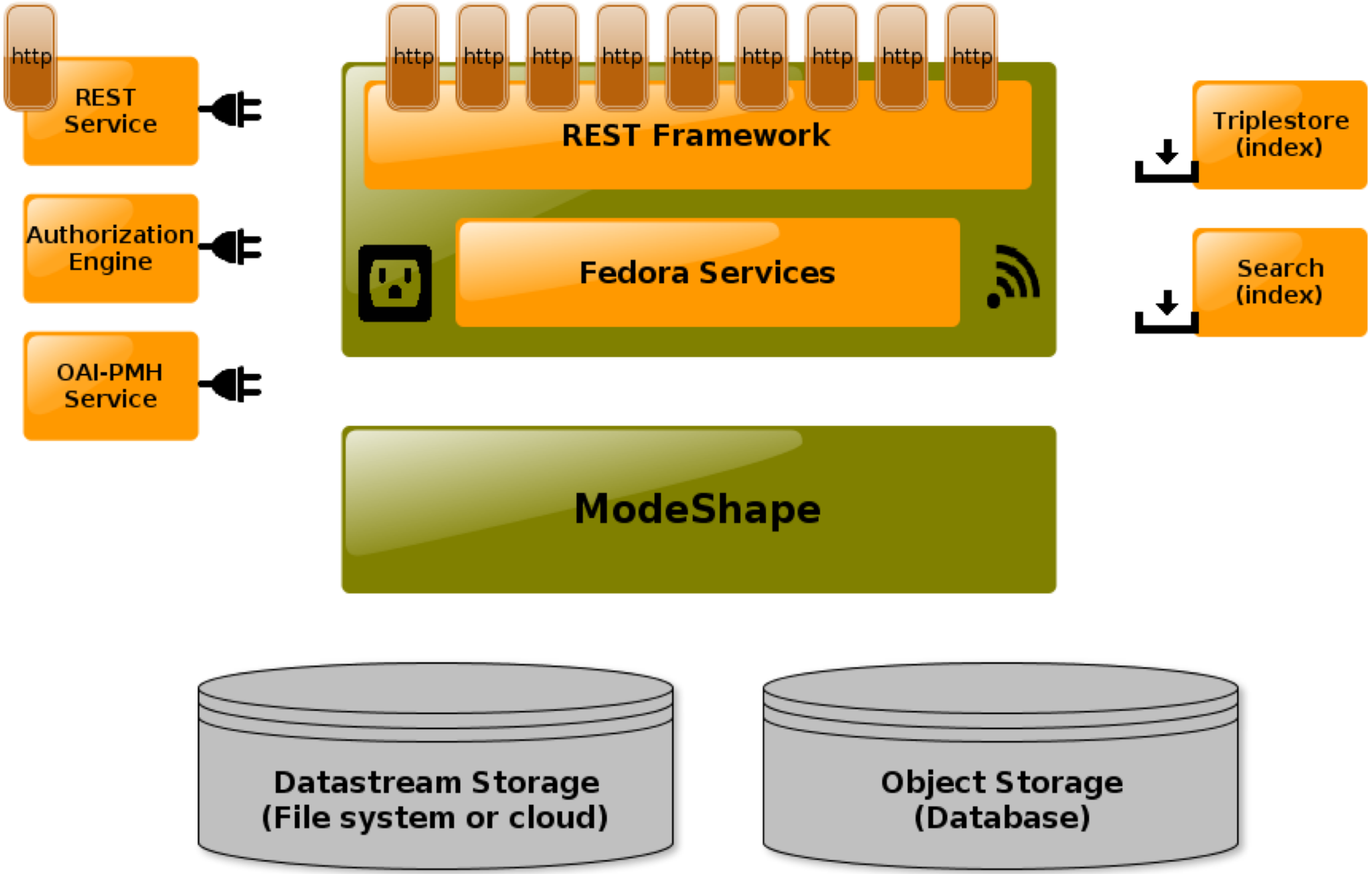
But...

- New standards evolve in metadata management and object modeling
  - Semantic web
  - Ontologies (e.g. PREMIS, MODS)
- Technology stack
  - is getting deprecated
  - lacks functionality
  - should support new/emerging standards
  - performance improvements necessary

Advantages of Fedora 4

- continued community support
- even more focus core functionality
  - modularizing/externalizing
  - using reliable technologies for core concepts:
    - Memento (Versioning)
    - ACL (Permissions)
- native support for Linked Data
  - Fedora 4 is a LDP server, i.e. implementing the W3C rules on handling web resources (https://www.w3.org/TR/ldp/)
  - Metadata not only as literals, but as resources

F4 Architecture



Fedora 5

- Fedora 5 has been published at the end of 2018
- primarily an alignment of the API and a reference implementation
  - Resource Management (Linked Data Platform)
  - Resource Versioning (Memento)
  - Resource Authorization (Web Access Controls)
  - Notifications (Activity Streams)
  - Binary Resource Fixity (HTTP headers)
- Adoptions by frameworks
  - Samvera/Hyrax/Avalon
  - Islandora

Onwards perspectives:

- Migration Path analysis: https://wiki.duraspace.org/display/FF/Designing+a+Migration+Path
- Roadmap: https://wiki.duraspace.org/display/FF/Roadmap

Data model, Matterhorn RDF

New data model:
From Matterhorn METS to Matterhorn RDF

- Matterhorn METS (XML):
  - based on METS, EAD, PREMIS (v2)
  - is today used in > 25 institutions in Switzerland, Germany, France.
  - will continue as a standard for SIPs/DIPs

- Matterhorn RDF
  - Fedora 4ff: A new RDF-based data model is possible
  - Rework the Matterhorn METS Profile in view of the possibilities offered by Linked Data and Open Linked Data
  - Resource can be uniquely identified using a URI
  - Relationships between resources can be qualified
  - The use of external resources and knowledge sources for cataloguing is greatly simplified (eg. Wikidata, GND, VIAF).
  - Make use of existing resources to improve the precision of archival description

Matterhorn RDF

- The Matterhorn RDF data model is again a joint development by the State Archives Canton of Wallis and docuteam. It is based both on the ICA standards for descriptive metadata (now ISAD/ISAAR/ISDF, in the future Records in Context (RiC)) and the OAIS information model.
- The class model of Matterhorn RDF is mainly based on the PREMIS 3 ontology and the library standard RDA (but not the FRBR data model). The PREMIS classes are enriched with attributes from other ontologies to be able to model the required details.
- Most important ontologies:
  - PREMIS 3
  - RDA (Ressource Description and Access)
  - DC, DC Terms, CIDOC, SKOS
- All informations are online:
  - https://wiki.docuteam.ch/doku.php?id=docuteam:matterhornrdf
  - https://matterhorn.tools

Matterhorn RDF: Improved contextualization

Contextualization of documents

- **What?** → Content (ISAD)
- **Who?** → Actors (ISAAR)
- **How?** → Process of creation (ISDF)

Contextualization goes further

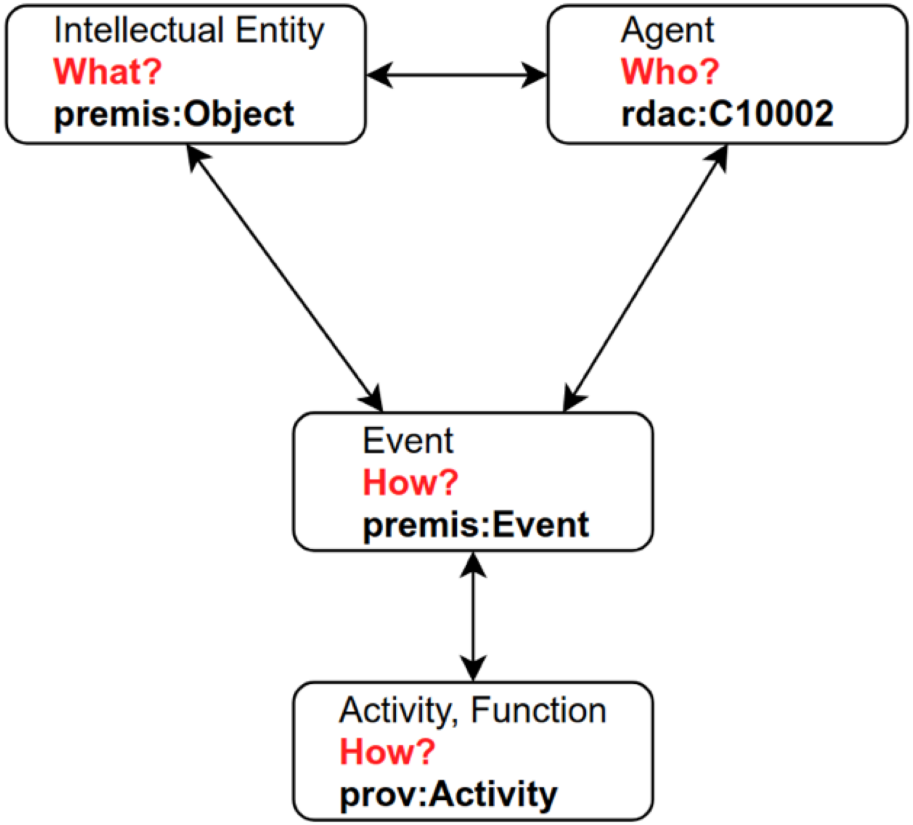
- Preservation description information
- Representation Information

The goal is a model that contains **both the content and the technical contextualization** of a document or a record.

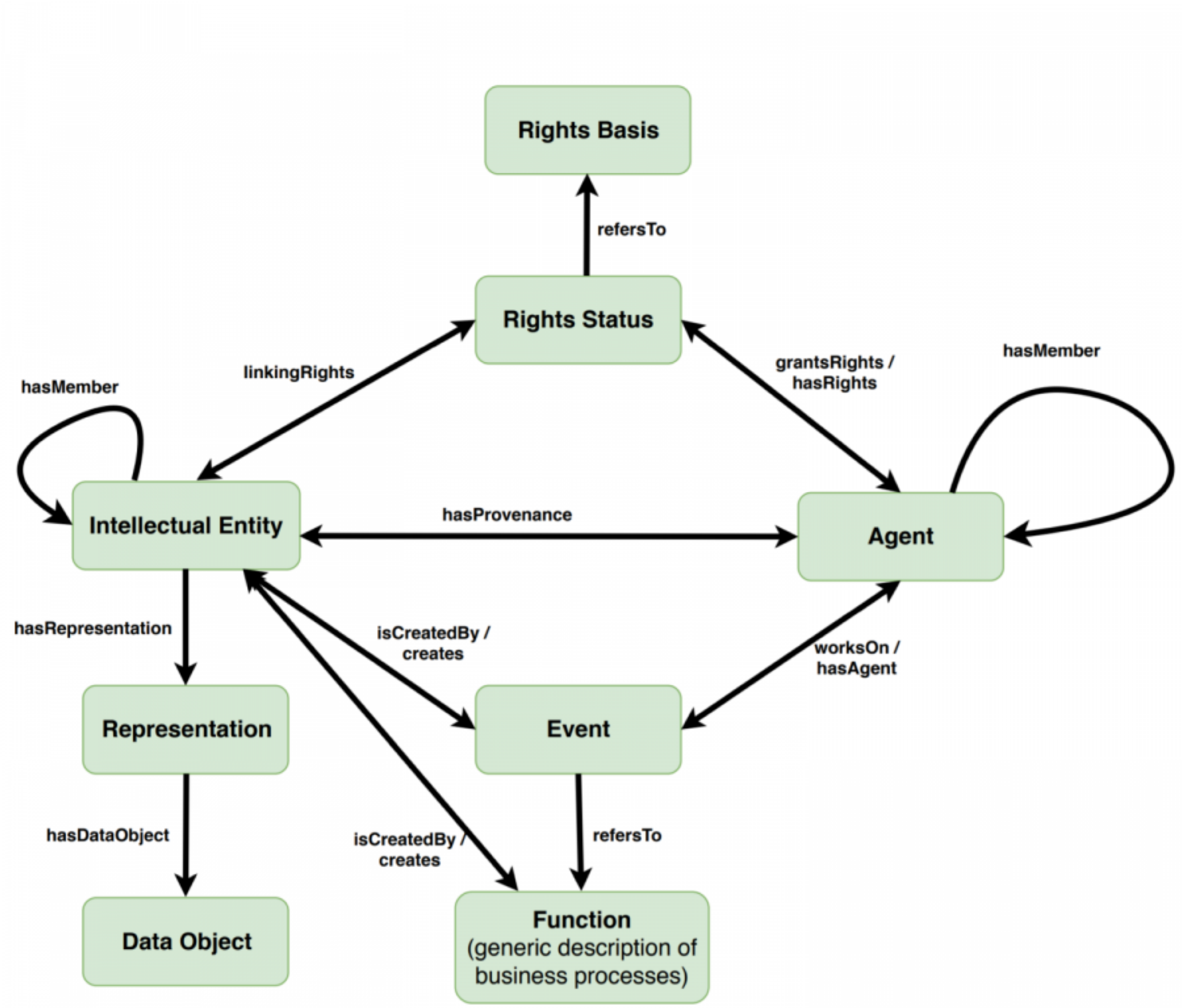
Matterhorn RDF: core concepts

Very similar to the PREMIS 3-model with three core classes

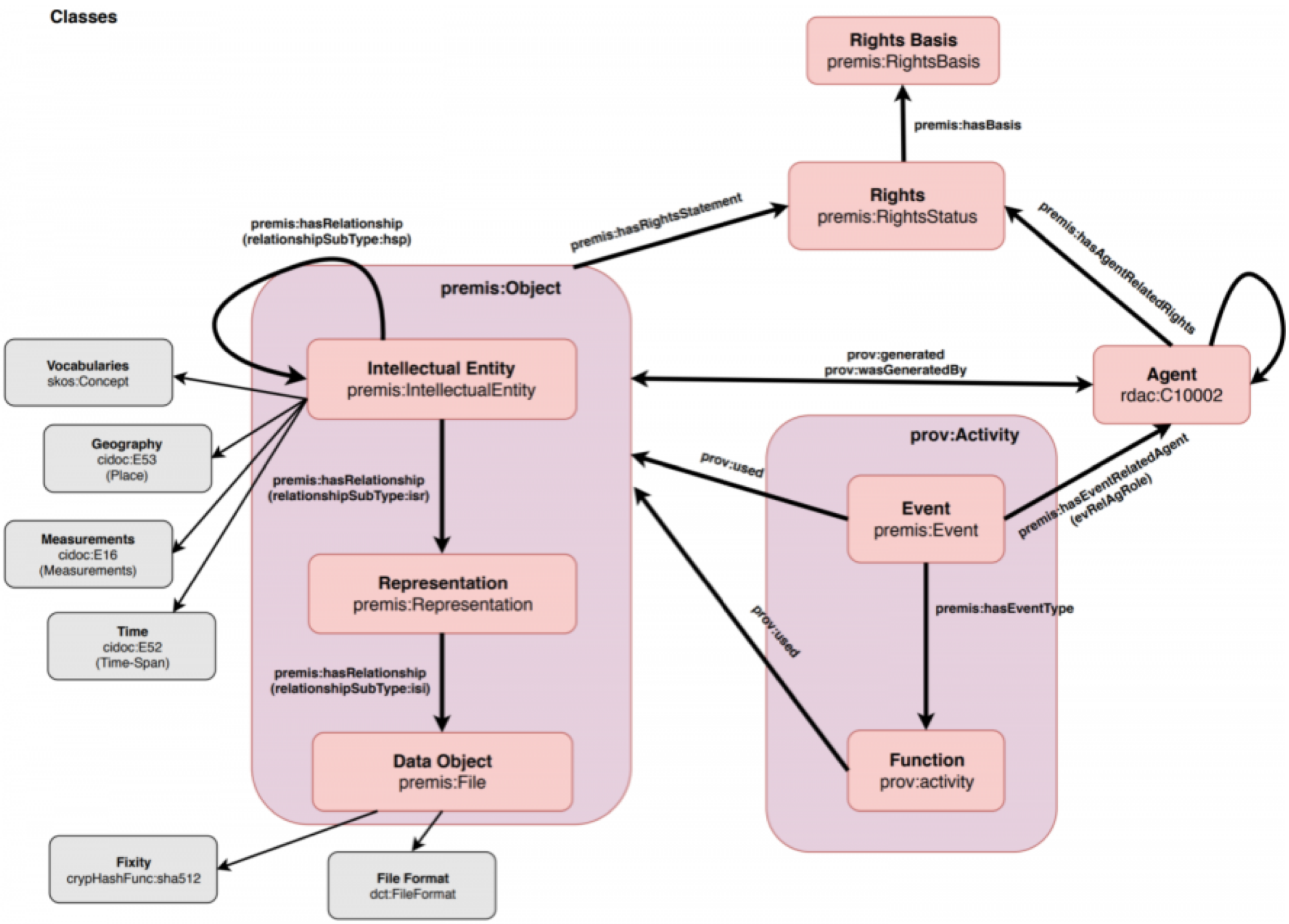
- Intellectual Entities (Records):
  - premis:object from PREMIS 3 ontology
- Agents:
  - rdac:C10002 from RDA ontology
- Functions and Events:
  - prov:Activity from PROV ontology of the W3C



Matterhorn RDF: Conceptual Model



Matterhorn RDF: Class model



SHACL instead of an Ontology

- No ontology for Matterhorn RDF. Maintenance would cost too much time + effort.
- Formalization is done in Shapes Constraint Language (SHACL)
- For each property, restrictions regarding value ranges, minimum or maximum occurence and data types are to be formulated.
- Statements made in triples can be validated

```
sh:property [
    sh:path dc:title ;
    rdfs:label "Title"@en ;
    rdfs:label "Titel"@de ;
    rdfs:label "Titre"@fr ;
    rdfs:comment "ISAD 1.2; RDA 'has
title'" ;
    # owl:sameAs rico:title ;
    sh:datatype xsd:string ;
    sh:minCount 1 ;
    sh:maxCount 1 ;
    sh:nodeKind sh:Literal ;
] ;
```

RiC/Records in Context and Matterhorn RDF

- ICA «Expert Group for Archival Description» (EGAD) works on «Records in Context» since 2012 to replace ISAD / ISAAR / ISDF

- **EGAD's non-generic approach:** Developing an RDF standard specific to archives, with gateways to library and museum standards

- **The Matterhorn RDF Data Model's generic approach:** Based upon RDF existing and consensual international standards, allowing to model Records in Context.

- In contrast to EGAD, the Matterhorn RDF Data Model is based on existing ontologies. It follows the best practices propagated by the W3C: **«It is best practice to use or extend an existing vocabulary before creating a new vocabulary.»**

Fedora 6, Oxford Common File Layout

Designing a Migration Path Survey Results

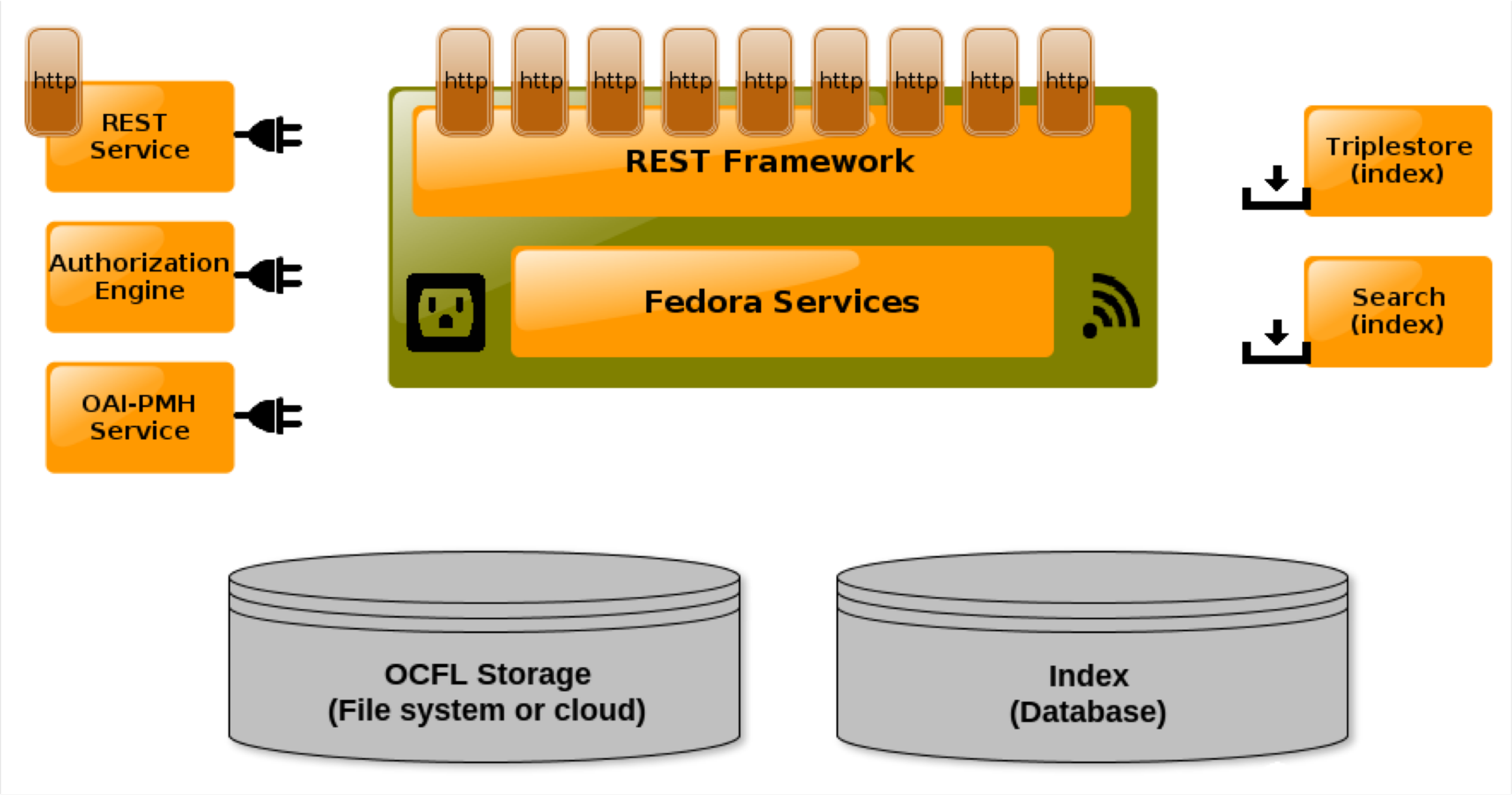
- **Goal of the study**: “assess the barriers to migrating to a supported version of Fedora”
- Almost **80%** of the participants are still **using Fedora 3**
  - current community for Fedora 4/5 is rather small
- **File System Layout** and **Preservation Capabilities** are **among the top 3 of Fedora 3 features**
- **Flexible Content Model** is the **top rated Fedora 3 feature**

Fedora 6 Pilot

- What we do
  - Weekly Tech Calls
  - Testing migration utilities for F3 → F6
  - Testing Fedora 6 setup an infrastructure
  - Test our tools against Fedora 6

- Goals
  - Running **Fedora 6 prototype** by the end of 2019
  - Start with migrations to Fedora 6 **soon after release**

F6 Architecture



Why we wait for Fedora 6

- Storage Structure similar to Fedora 3
  - Open Standard (repository independent)
  - same backup strategies can be used
  - easier migration

- Fedora objects will be self-describing outside of the context of the software
  - meaning: metadata and data will be stored together

- Rebuild from disk possible (again)

Why we wait for Fedora 6

docuteam

- Synchronous Search Service included
  - provides basic search functionality over repository
  - full text search will still need an external index
- Implements same the Fedora API as Fedora 5
- Support for OpenJDK 11

Fedora 3 → Fedora 6

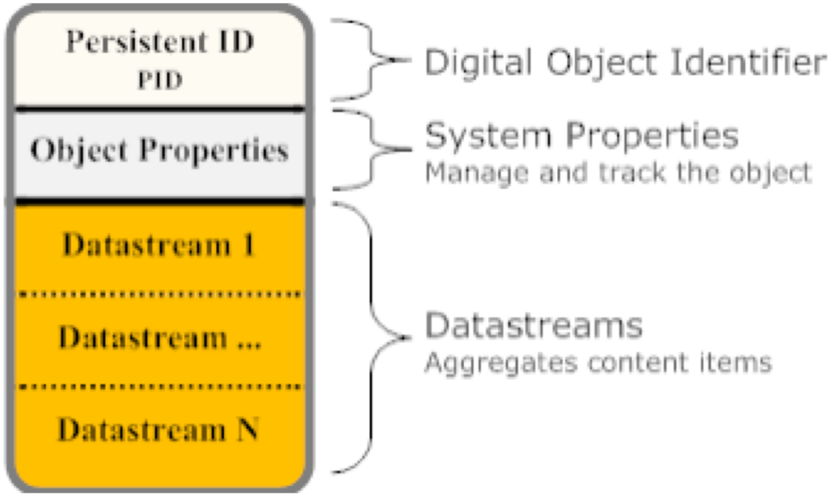
docuteam

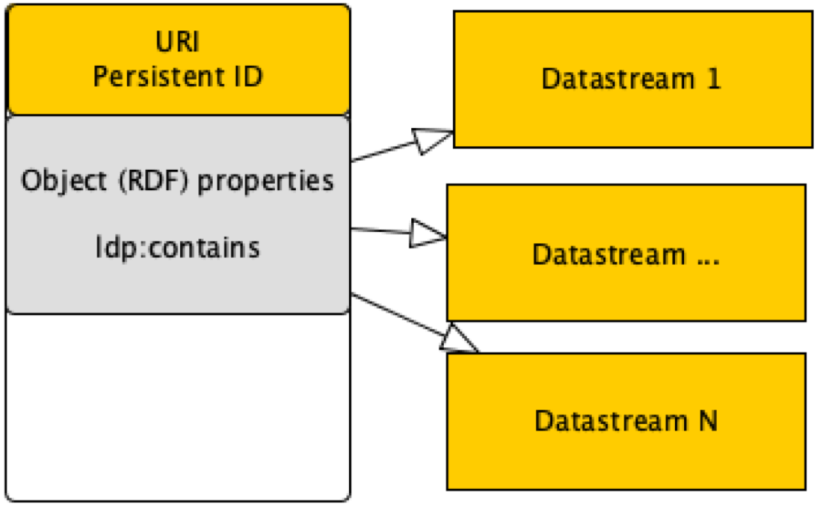
Fedora 3:

- Object
- Object Properties
- Datastream
- Datastream Properties

Fedora 6:

- RDF Ressource
- RDF properties
- Binary Ressource
- RDF properties of Binary Ressource





Oxford Common File Layout (OCFL)

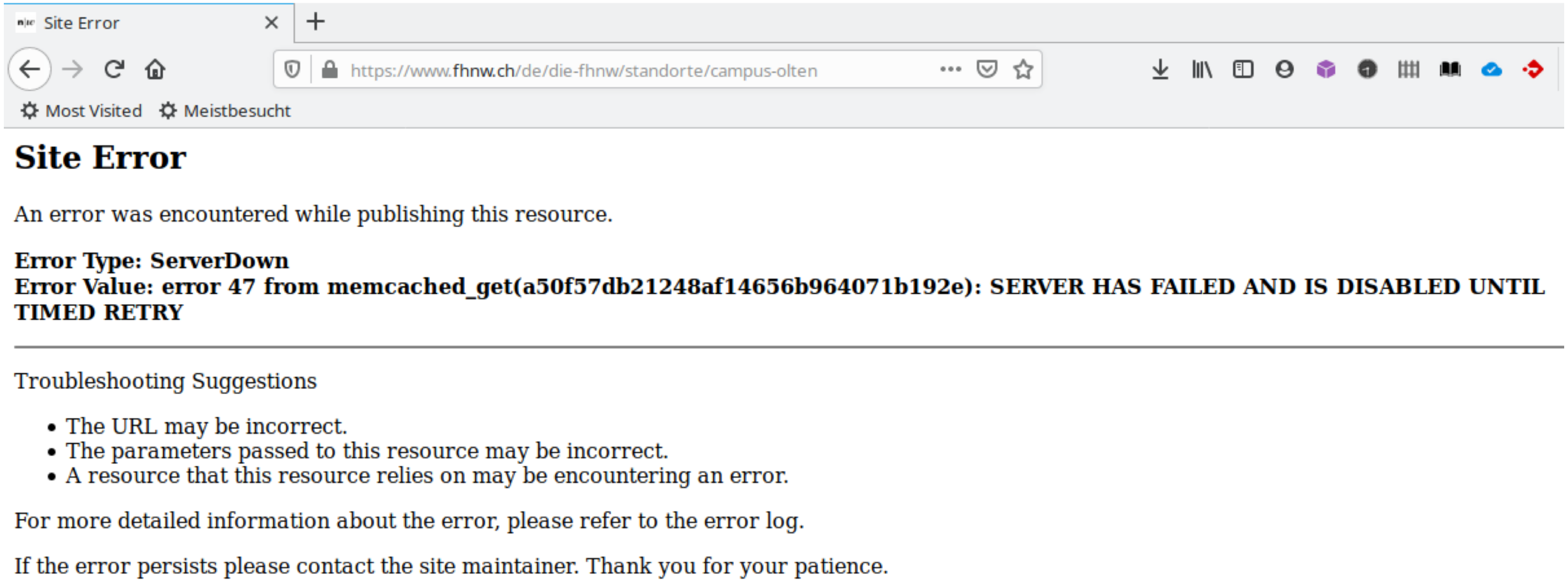
"This Oxford Common File Layout (OCFL) specification describes an application-independent approach to the storage of digital information in a structured, transparent, and predictable manner. It is designed to promote long-term object management best practices within digital repositories."

Source: https://ocfl.io/, 30.10.2019.

- Website: https://ocfl.io/
- Latest specification draft: https://ocfl.io/draft/spec/

Why OCFL?

docuteam



Site Error

An error was encountered while publishing this resource.

Error Type: ServerDown
Error Value: error 47 from memcached_get(a50f57db21248af14656b964071b192e): SERVER HAS FAILED AND IS DISABLED UNTIL TIMED RETRY

Troubleshooting Suggestions

- The URL may be incorrect.
- The parameters passed to this resource may be incorrect.
- A resource that this resource relies on may be encountering an error.

For more detailed information about the error, please refer to the error log.

If the error persists please contact the site maintainer. Thank you for your patience.

Why OCFL?

- **Completeness**, so that a repository can be rebuilt from the files it stores
- **Parsability**, both by humans and machines, to ensure content can be understood in the absence of original software
- **Robustness** against errors, corruption, and migration between storage technologies
- **Versioning**, so repositories can make changes to objects allowing its history to persist
- **Storage diversity**, to ensure content can be stored on diverse storage infrastructures including cloud object stores

Source: https://ocfl.io/, 30.10.2019.

File Layout of an OCFL Storage Root

```
ocfl  Storage root
 ├─ '0=ocfl_1.0'  Root conformance declaration
 ├─ ocfl_1.0.txt Human-readable text of the OCFL specification; optional
 ├─ ocfl_layout.json Description of storage hierarchy layout; optional
 ├─ it
 |    ├─ 2h
 |    |    └── s9
 |    |          └── [ojbect_root]
 |    ├─ 9a
 |    |    └── n2
 |    |          └── [object_root]
 |    └─ 0c
 |        └── f7
 |              └── …
 └── …
```

File layout of an OCFL object without binary (example: it-01:1)

```
it-01%3a1 Object root
├─ '0=ocfl_object_1.0' Object conformance declaration
├─ inventory.json Inventory
├─ inventory.json.sha512 Inventory digest
├─ v1 Version
│   ├─ inventory.json
│   ├─ inventory.json.sha512
│   └─ content
│       └─ object.ttl RDF data
├─ v2
│   └─ ...
├─ v...
└─ head Mutable version
```

File layout of an OCFL object with binary (example: it-01:2)

```
it-01%3a2 Object root
 ├─ '0=ocfl_object_1.0' Object conformance declaration
 ├─ inventory.json Inventory
 ├─ inventory.json.sha512 Inventory digest
 ├─ v1 Version
 │   ├─ inventory.json
 │   ├─ inventory.json.sha512
 │   └─ content
 │       ├─ object.ttl RDF data
 │       └─ ORIGINAL Binary file
 ├─ v2
 │   └─ ...
 ├─ v...
 └─ head Mutable version
```

Serialization of RDF data in an OCFL object

Content of `object.ttl`:

```
<info:fedora/it-01:1>
        <info:fedora/fedora-system:def/model#createdDate>
                "2017-08-
24T10:50:59.705Z"^^<http://www.w3.org/2001/XMLSchema#dateTime> ;
        <info:fedora/fedora-system:def/model#label>
                "01-SimpleTXT_1KB" ;
        <info:fedora/fedora-system:def/model#ownerId>
                "" ;
        <info:fedora/fedora-system:def/model#state>
                "Active" ;
        <info:fedora/fedora-system:def/view#lastModifiedDate>
                "2017-08-
24T10:50:59.705Z"^^<http://www.w3.org/2001/XMLSchema#dateTime> .
```