Objects, types, collections and operations in DOIP

GEDE Workshop on Digital Objects

Ulrich Schwardmann

Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen (GWDG)

Am Fassberg, 37077 Göttingen
ulrich.schwardmann [at] gwdg.de

26 September 2018, Bruxelles

ePIC
Persistent Identifiers for eResearch

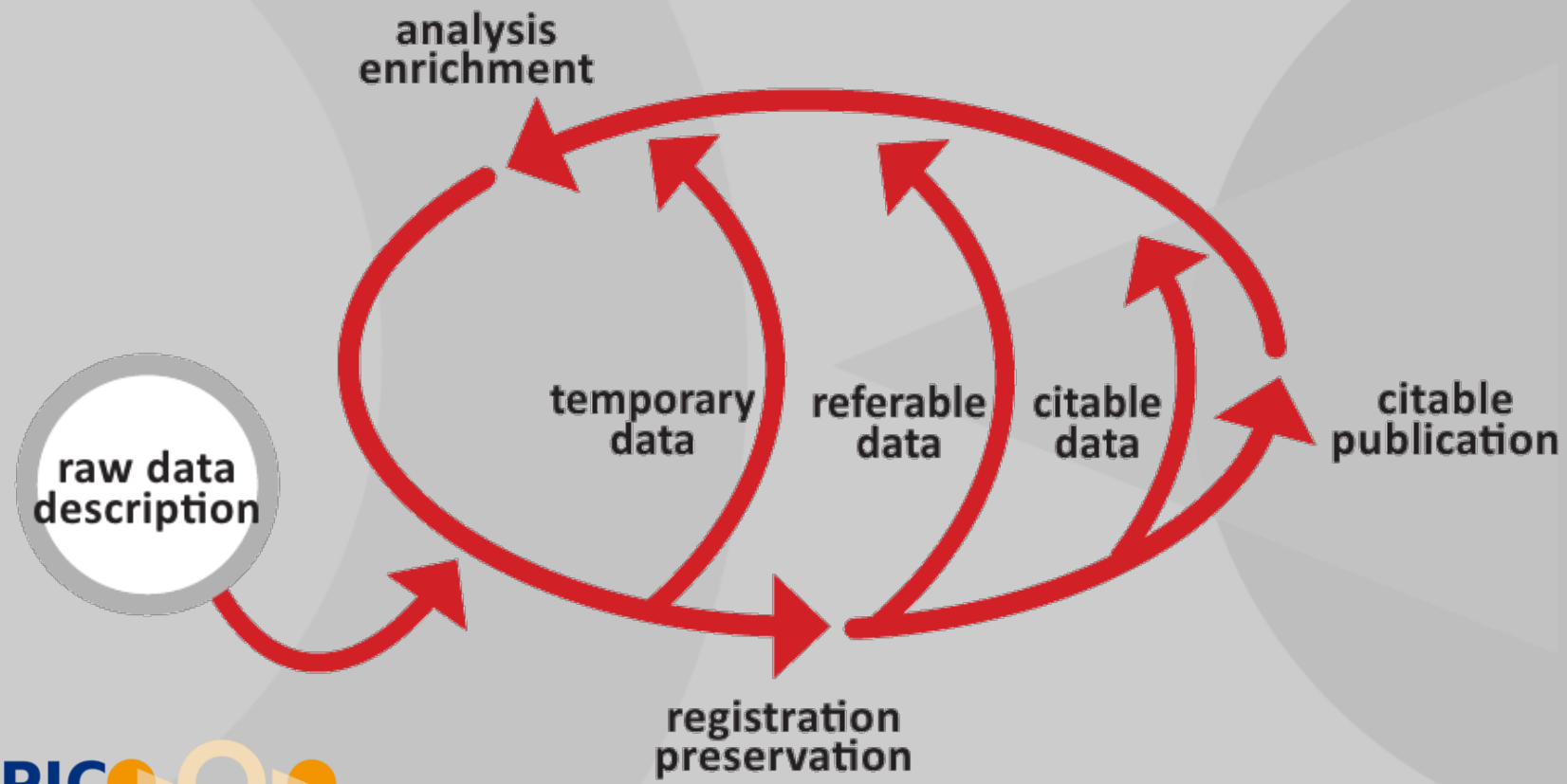
Content

GWDG, ePIC and DONA

- GWDG
  - is computer center of the University of Göttingen
  - and competence center for the Max-Planck-Society
- ePIC
  - is a network of currently eight strong scientific service providers
  - that signed a contract to **ensure a reliable PID infrastructure** for research
- DONA
  - is a Swiss foundation hosting an international consortium
  - that governs the Handle structure at the top level
  - GWDG is DONA MPA for ePIC

Research Data Life Cycle

- *The Scientific Supply Chain/Cycle:*
  - inputs are sensors, simulations, public data ...
  - products are publications and data
- sharing data needs reliable references across domains



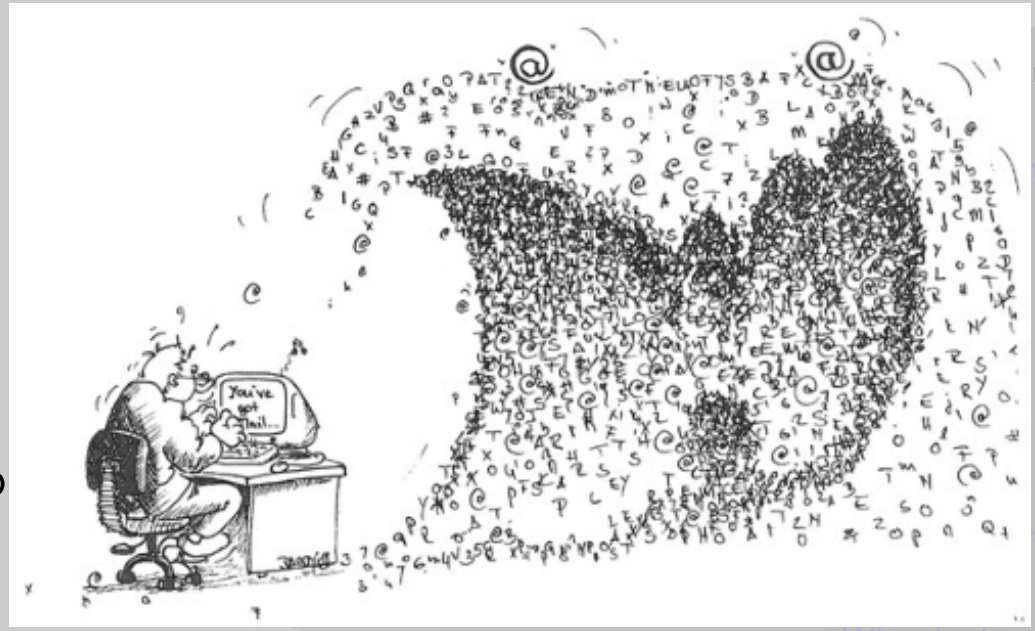
ePIC
Persistent Identifiers for eResearch

What is the problem?

Dynamics in the Data Domain

- Data heterogeneity hampers data exchange and reuse already now.

- about 80% of the time of data experts is wasted with data wrangling (i.e. making data ready for analytics),
  - findings in relevant data analytics projects:
  - RDA EU 2013 Survey: 75%
  - M. Brodie MIT S.: 80%
  - CrowdFlower 2017 S.: 79%



- In industry the phenomena are essentially the same
  - BD/AI Summit 2018: 60% of industrial data projects fail
- All will become even worse with IoT and new sensors

ePIC
Persistent Identifiers for eResearch

What is the proposal?

Abstractions in the Data Domain

- the mayor obstacle for automation: **Heterogeneity and Complexity of Data**
- Abstraction
  - is a way to hide heterogeneity and complexity
- Virtualisation
  - provides a layer of abstraction between data and application
  - in our case the reference becomes a placeholder for data
- Encapsulation
  - provides a layer of abstraction between inner heterogeneity and complexity and outer simplification
  - in our case the reference becomes the broker for information about inner complexity

Abstractions in the Data Domain

Classical abstraction in Computer Science:

pointer

- as reference to avoid complexity of operations (synchronisation, ...)

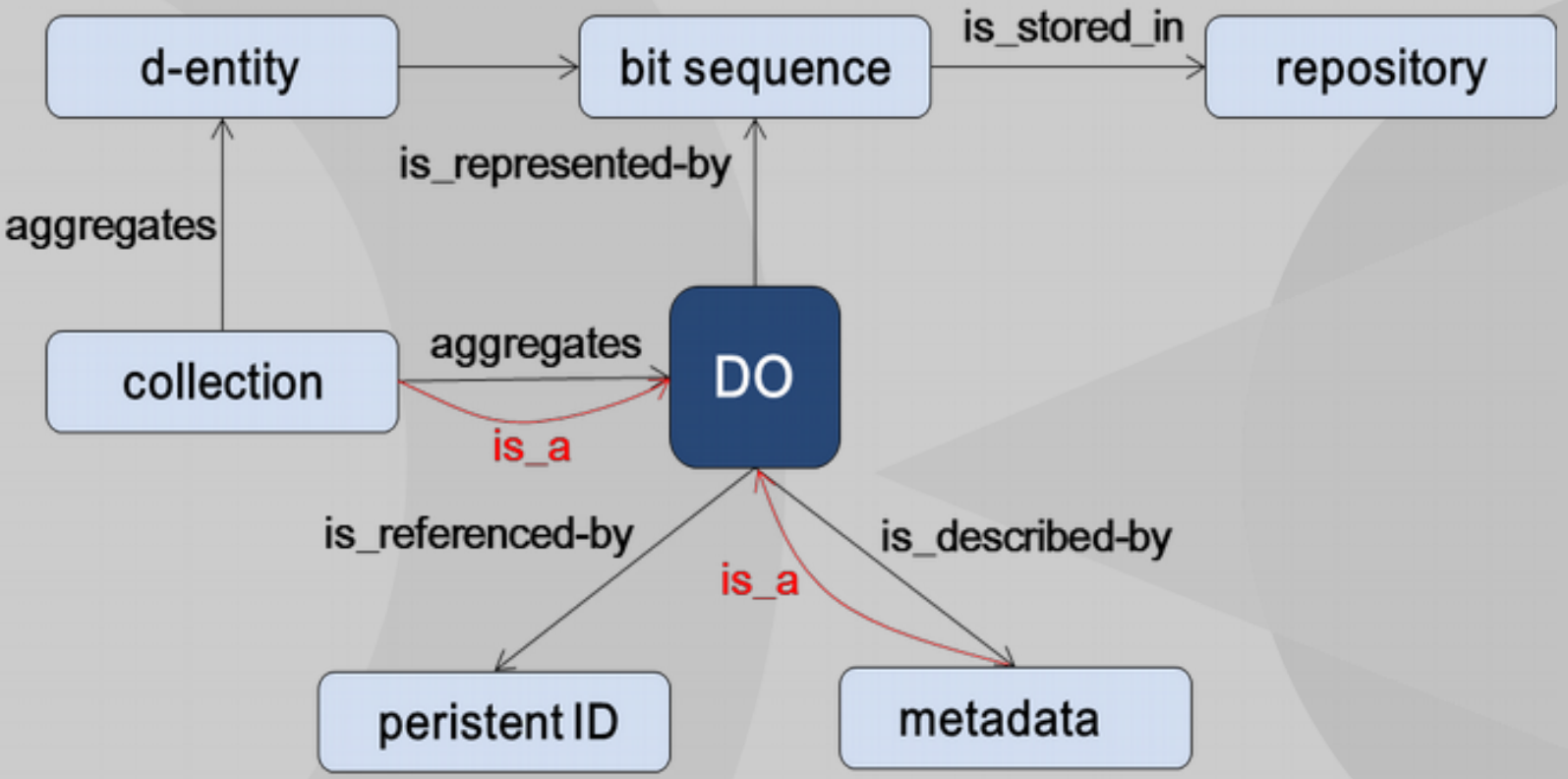
Abstraction for cross domain data management:

enhanced pointer

- as reference that provide
  - understandible description
  - reliability of global resolution
- again in order to simplify and automate operations

Digital Objects

The Core Data Model of the
RDA Data Foundation & Technology Working Group



ePIC
Persistent Identifiers for eResearch

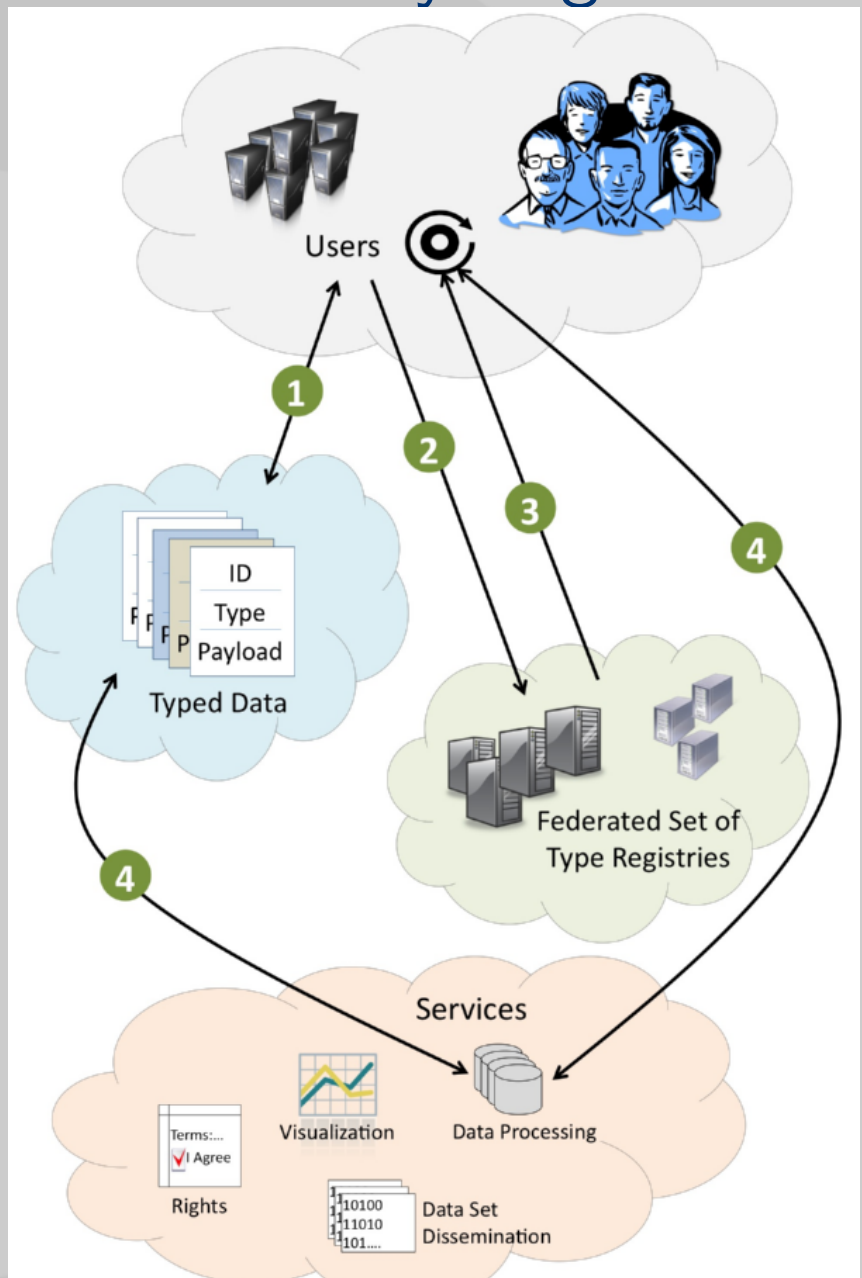
Reusability

- needs knowledge about basic properties of data
  - **Metadata** is often unavailable, not connected to data or not interpretable
- Registration:
  bind metadata and data with PID to a digital object
- For reuse provide as much of this knowledge before access to the data
- **PID Information Types**
  - are additional metadata, stored in the PID database
  - similar to Mime Types, but much more flexible
  - Examples are *checksum* , *mime type* , *reference information, versioning (relative and absolute), embargo time, expiration date, add. metadata location, basic Dublin Core, access restrictions and methods, data and table column formats, collection description, ...*

Reusability

- needs knowledge about basic properties of data
  - **Metadata** is often unavailable, not connected to data or not interpretable
- Registration:
  bind metadata and data with PID to a digital object
- For reuse provide as much of this knowledge before access to the data
- **PID Information Types**
  - are additional metadata, stored in the PID database
  - similar to Mime Types, but much more flexible
  - Examples are *checksum , mime type , reference information, versioning (relative and absolute), embargo time, expiration date, add. metadata location, basic Dublin Core, access restrictions and methods, data and table column formats, collection description, ...*

Interoperability by Registration of Types



RDA working group on
Data Type Registries

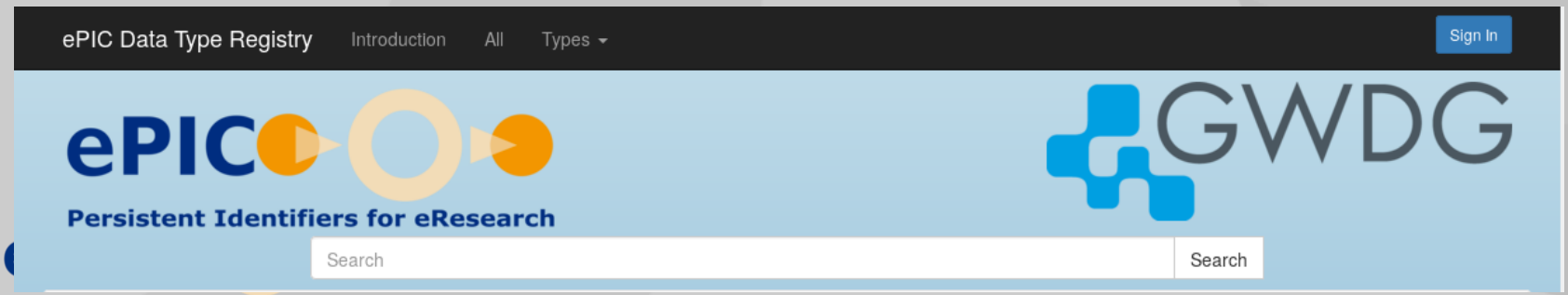
- approach to provide *type definitions*
- a PID for each definition
- defines the type structure, its use and semantics
- CORDRA as DTR service
- typical use cases:
  - with given PID find a type and ask for its use at DTR (see left)
  - ask at DTR for types with given semantics and find via PIDs according data

The ePIC Data Type Registry

- Features
  - Definition of PID Information Types
  - hierarchical types and automated schema extraction
  - Access via REST API, Browser
- based on CORDRA software
- GWDG is provider on behalf of ePIC
- Who can use the service?
  - public, authorization needed only for type definition
- Overview: `http://dtr.pidconsortium.eu/`

Policies for a PID InfoType life cycle:

- *in preparation* (21.T11148),
  - `http://dtr-test.pidconsortium.eu/`
- *candidate, approved, deprecated* (21.11104)
  - `http://dtr-pit.pidconsortium.eu/`

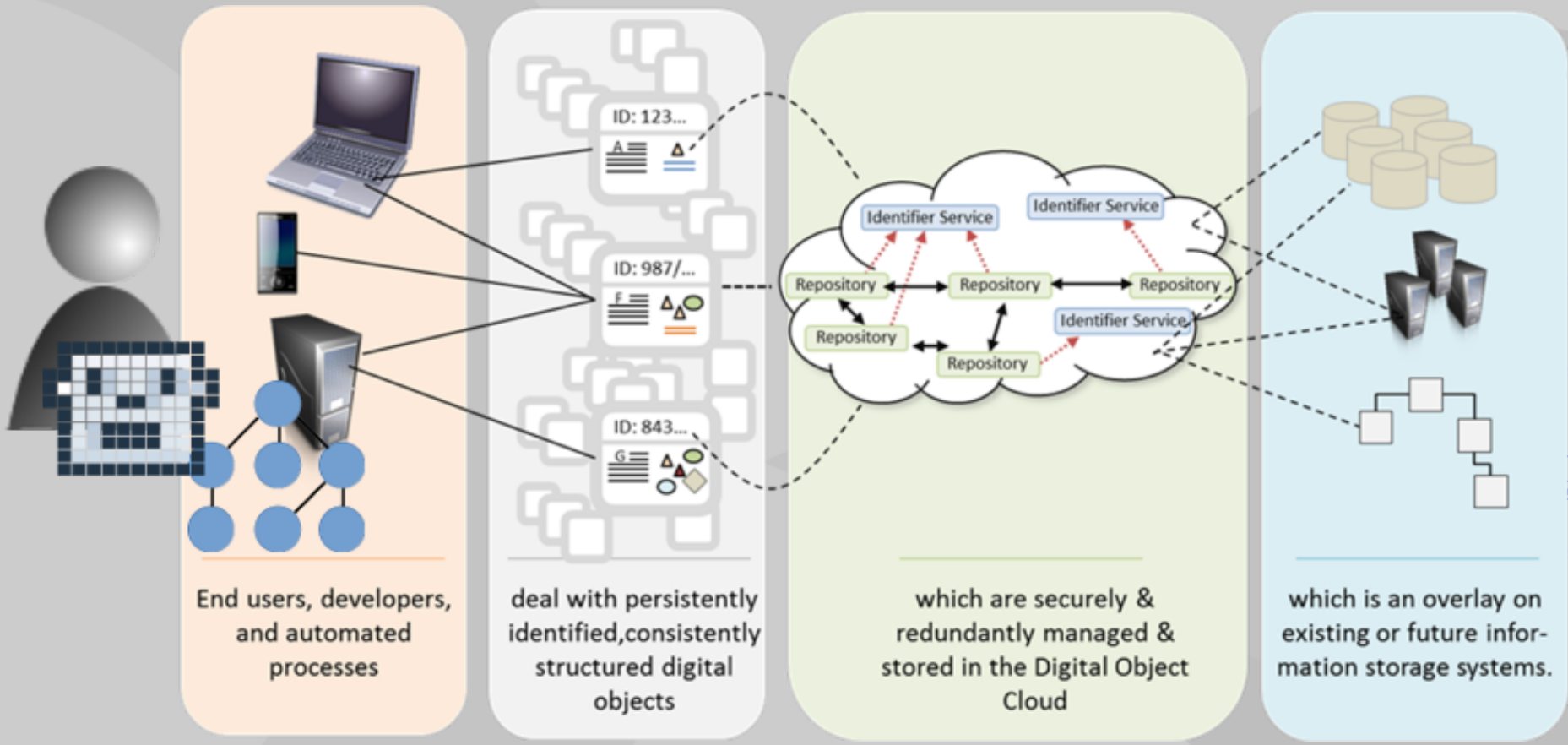


What are the advantages?

The Digital Object Cloud

Encapsulated Complexity for the Users View of the DO Cloud

Objects, types, collections and operations in DOIP



[1] Global Digital Object Cloud, Larry Lannom, 2016

Types vs. Linked Data

- An Example of a type: `isPreviousVersionOf`
  - Such a type is stored as key-value pair in the PID (*pid-do1*) of a digital object
  - as key-value pair consisting of the *type* and the PID of the previous version (*pid-do2*)

This gives a triple:

- *pid-do1 type pid-do2*
- Digital-Object-1 `isPreviousVersionOf` Digital-Object-2

Thus one has a relation:

subject predicate object

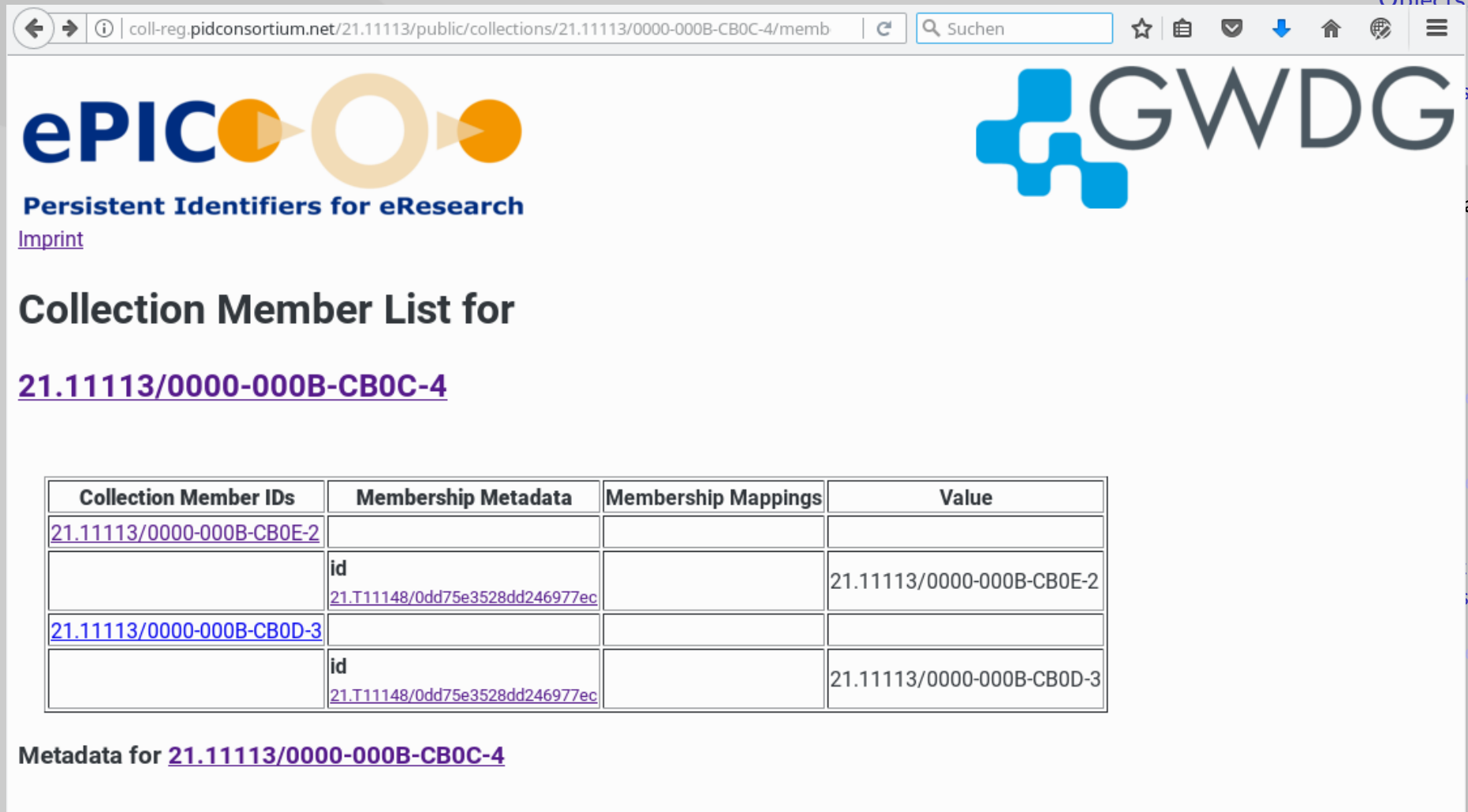
with types as predicates.

- Types can be represented by PIDs again (DTR)

Pointers enable Recursion

- **Collections** in the RDA sense are PIDs pointing to a list of PIDs
  - and additional metadata to enable services
  - this is a **recursive** definition: members can be collections
- the RDA outcome is a concrete REST API to manage collections
- collections are ubiquous also in data management:
- collections are a very general way to organize objects hierarchically
- often repositories have an implicit hierarchical structure

A Collection Repository



What is the next step?

data driven research relies on **methods using data**

and **data management relies on operations on data**

- it is therefore even more important to have
  - **reliable references to operations**
  - and the **exakt description of operations**
- Technology for cross domain operations: **web services**
  - which are given by ressources (not operations) and methods (operations in operations)
- WSDL/RSDL tries to give descriptions for web services
- a possible approach could be
  - use a **PID to reference the location** of a web service
  - additionally use a **PID Info type to refer to the WSDL/RSDL**
- But the expressiveness of WSDL/RSDL is very limited
  - there is often no WSDL/RSDL at all necessary for REST
  - the operations are **only described by API descriptions**

data driven research relies on **methods using data**

and **data management relies on operations on data**

- it is therefore even more important to have
  - **reliable references to operations**
  - and the **exakt description of operations**
- Technology for cross domain operations: **web services**
  - which are given by ressources (not operations) and methods (operations in operations)
- WSDL/RSDL tries to give descriptions for web services
- a possible approach could be
  - use a **PID to reference the location** of a web service
  - additionally use a **PID Info type to refer to the WSDL/RSDL**
- But the expressiveness of WSDL/RSDL is very limited
  - there is often no WSDL/RSDL at all necessary for REST
  - the operations are **only described by API descriptions**

data driven research relies on **methods using data**

and **data management relies on operations on data**

- it is therefore even more important to have
  - **reliable references to operations**
  - and the **exakt description of operations**
- Technology for cross domain operations: **web services**
  - which are given by ressources (not operations) and methods (operations in operations)
- WSDL/RSDL tries to give descriptions for web services
- a possible approach could be
  - use a **PID to reference the location** of a web service
  - additionally use a **PID Info type to refer to the WSDL/RSDL**
- But the expressiveness of WSDL/RSDL is very limited
  - there is often no WSDL/RSDL at all necessary for REST
  - the operations are **only described by API descriptions**

data driven research relies on **methods using data**

and **data management relies on operations on data**

- it is therefore even more important to have
  - **reliable references to operations**
  - and the **exakt description of operations**
- Technology for cross domain operations: **web services**
  - which are given by ressources (not operations) and methods (operations in operations)
- WSDL/RSDL tries to give descriptions for web services
- a possible approach could be
  - use a **PID to reference the location** of a web service
  - additionally use a **PID Info type to refer to the WSDL/RSDL**
- But the expressiveness of WSDL/RSDL is very limited
  - there is often no WSDL/RSDL at all necessary for REST
  - the operations are **only described by API descriptions**

Try to make data operations simpler

Can we try to describe data operations similar to mathematical functions

$$f : X \to Y,\ x \mapsto f(x)$$

where f is the function name, X and Y are domain (source S) and codomain (target T) of data and metadata (incl. AAI)?

- Lets have a look at the definitions in the DOIP draft:
- *operation/function name*
  - `operationId`: is f , the identifier of the operation
- *data*
  - `targetId` (S): Id of the source DO
  - `input/output` (S,T): arbitrary I/O streams.
- *metadata*
  - `requestId` (S,T): the (unique) identifier of the request
  - `attributes` (S,T): optional array of JSON properties
  - `clientId` (S): the identifier of the client (AAI).
  - `authentication` (S): optional AAI JSON (sub) object
  - `status` (T): status identifier

Try to make data operations simpler

Can we try to describe data operations similar to mathematical functions

$$f : X \to Y,\ x \mapsto f(x)$$

where f is the function name, X and Y are domain (source S) and codomain (target T) of data and metadata (incl. AAI)?

- Lets have a look at the definitions in the DOIP draft:
- *operation/function name*
  - `operationId`: is f , the identifier of the operation
- *data*
  - `targetId` (S): Id of the source DO
  - `input/output` (S,T): arbitrary I/O streams.
- *metadata*
  - `requestId` (S,T): the (unique) identifier of the request
  - `attributes` (S,T): optional array of JSON properties
  - `clientId` (S): the identifier of the client (AAI).
  - `authentication` (S): optional AAI JSON (sub) object
  - `status` (T): status identifier

Try to make data operations simpler

Can we try to describe data operations similar to mathematical functions

$$f : X \to Y,\ x \mapsto f(x)$$

where f is the function name, X and Y are domain (source S) and codomain (target T) of data and metadata (incl. AAI)?

- Lets have a look at the definitions in the DOIP draft:
- *operation/function name*
  - `operationId`: is f , the identifier of the operation
- *data*
  - `targetIds` (S,T): Ids of the source and target DOs
  - `input/output` (S,T): *arbitrary I/O streams?*
- *metadata*
  - `requestId` (S,T): the (unique) identifier of the request
  - `attributes` (S,T): optional array of JSON properties
  - `clientId` (S): the identifier of the client (AAI).
  - `authentication` (S): optional AAI JSON (sub) object
  - `status` (T): status identifier

Many Thanks

Questions ???

Contact at ePIC:

- support [at] pidconsortium.eu

Contact at GWDG:

- **Ulrich Schwardmann**
  T: 0551 201-1542, E: ulrich.schwardmann [at] gwdg.de