PIDs, Types and the Semantic Web

Ulrich Schwardmann

Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen
(GWDG)

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

24 April 2017, Wien

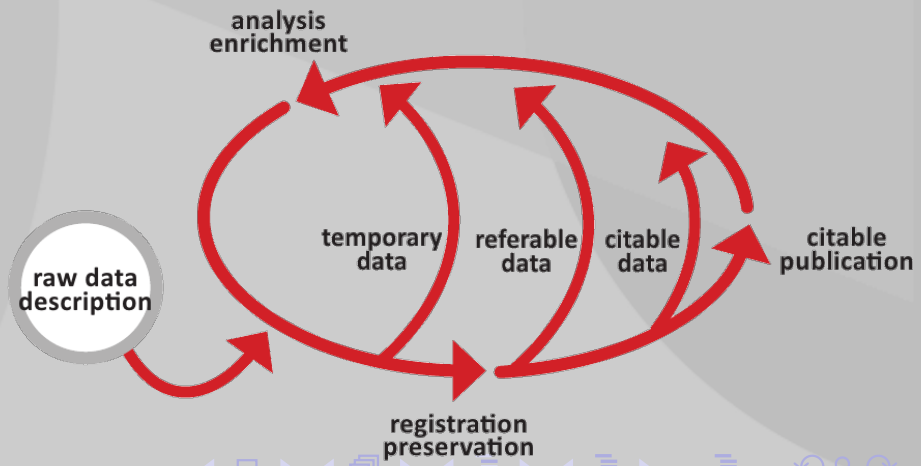
CC BY

GWDG and ePIC

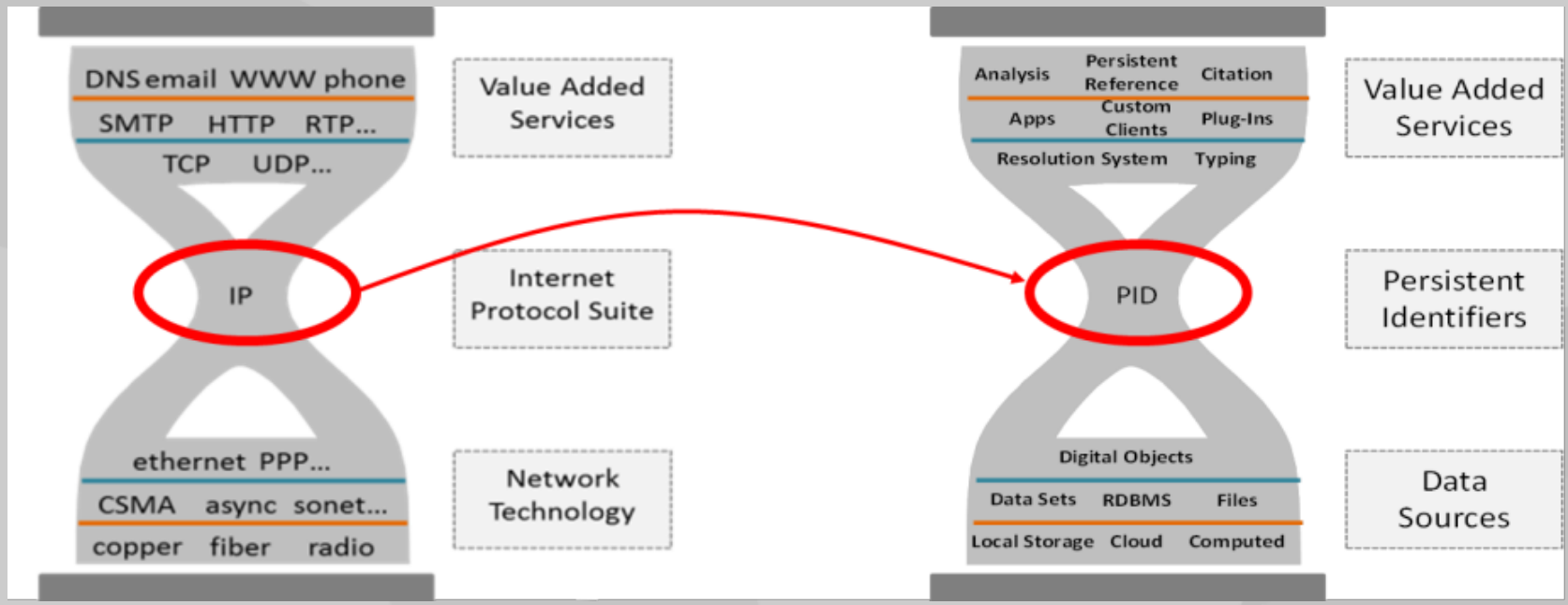
- GWDG is a joint facility of Göttingen University and the Max Planck Society. It functions as the
  - data and IT competence center for the Max Planck Society, and as the
  - data center for the university.

- ePIC Persistent Identifiers for eResearch builds a network of currently six strong scientific service providers signed a contract
  - to ensure a reliable and persistent identifier infrastructure
  - devoted to the needs of the research community at large.

- Mayor focus: the referability of data
  - with finer granularity
  - for sharing during the research process



Persistent Identifier



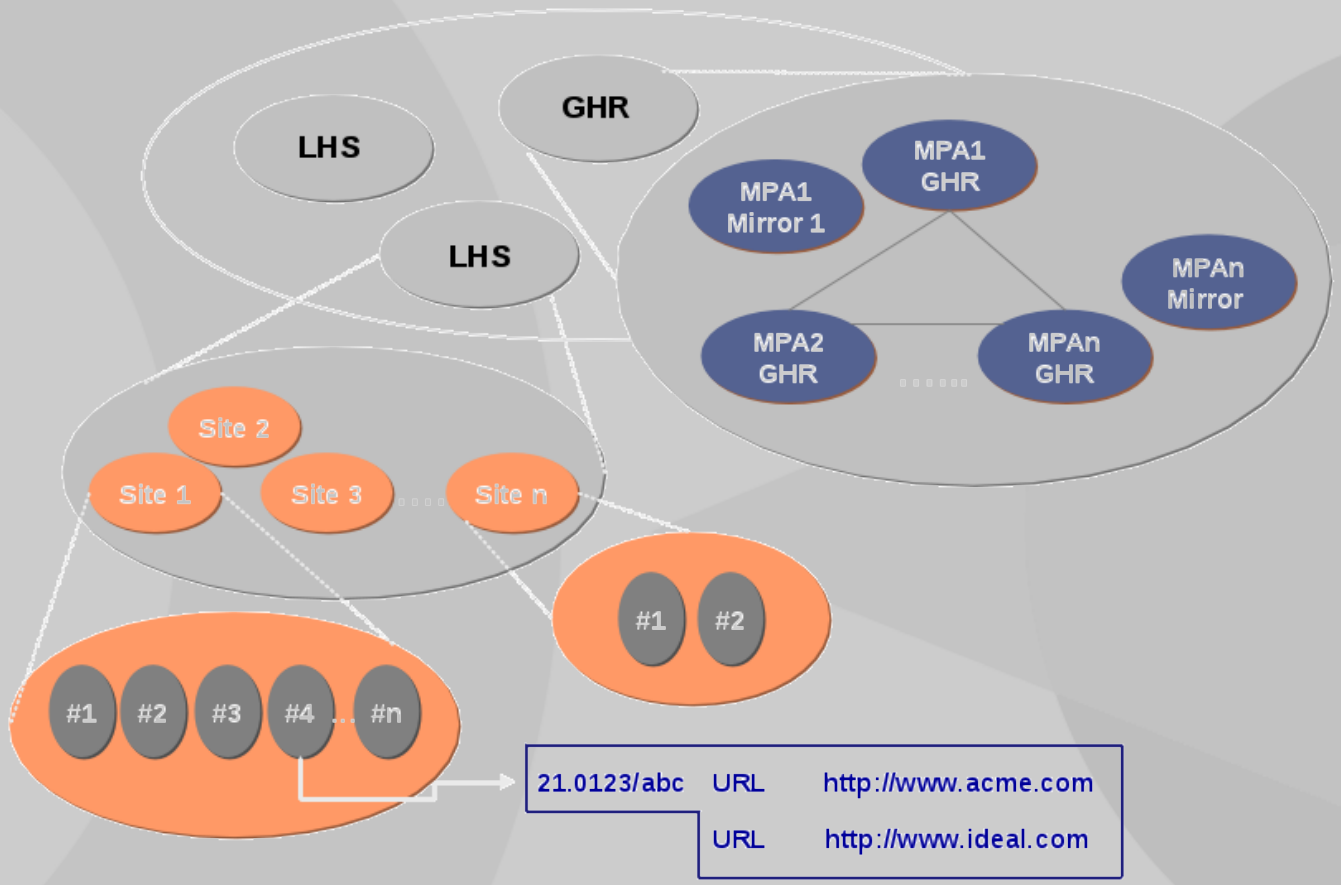
[1]

- In RDA is the PID seen as the central component to mediate the access of *services* to *digital objects*
- Key feature for the central role of PIDs is the **resolution** and the **transparent presentation of metadata** by types
- This concept is also known as *digital object architecture (DOA)*

[1] Larry Lannom,CNRI, first presented at RDA P5

Resolution of Handles

The Multi Primary Administrator GHR of DONA



Policies and Profiles

- Policies of PID services describe the general way, how PIDs are maintained
  - deletion of PIDs
  - patterns for PID suffixes
  - enforce type profiles
- Profiles of PID services describe the alloweed type pattern
  - lists for mandatory, optional and forbidden types

Types and PID Information Types (PIT)

- Types are a kind of metadata for digital objects,
  - rather structural and administrative then descriptive.
  - Types are used as **key**s for given **value**s.
- Types can be stored inside the Persistent Identifier (PID) record.
  - This way one can operate with them independently from the digital object itsself,
  - and similar to **mime types**
    as useful information before processing the data
- Such types are called **PID Information Types (PITs)**
- They are well referenced by a persistent identifier assigned to its definition (registration in a DTR).

CC BY

Types and PID Information Types (PIT)

- Types are a kind of metadata for digital objects,
  - rather structural and administrative then descriptive.
  - Types are used as **key**s for given **value**s.
- Types can be stored inside the Persistent Identifier (PID) record.
  - This way one can operate with them independently from the digital object itsself,
  - and similar to **mime types**
    as useful information before processing the data
- Such types are called **PID Information Types (PITs)**
- They are well referenced by a persistent identifier assigned to its definition (registration in a DTR).

CC BY

Data Type Registries (DTR)

By registration of types in DTRs the properties of types become transparent (standards)

- Currently a couple of such DTRs exist,
  - based on *Cordra*[2], a software from an RDA WG outcome,
- The DTRs use a special vocabular for type specifications.
  - This vocabular is partly extended for different purposes.
- A process of federation of DTRs is ongoing,
  - which also requires a standard for the vocabular
- ePIC provides a DTR at `http://dtr.pidconsortium.eu/`
  - a type life cycle is introduced:
    - **in preparation**, **candidate**, **approved** and **deprecated**

[2] https://www.cordra.org/

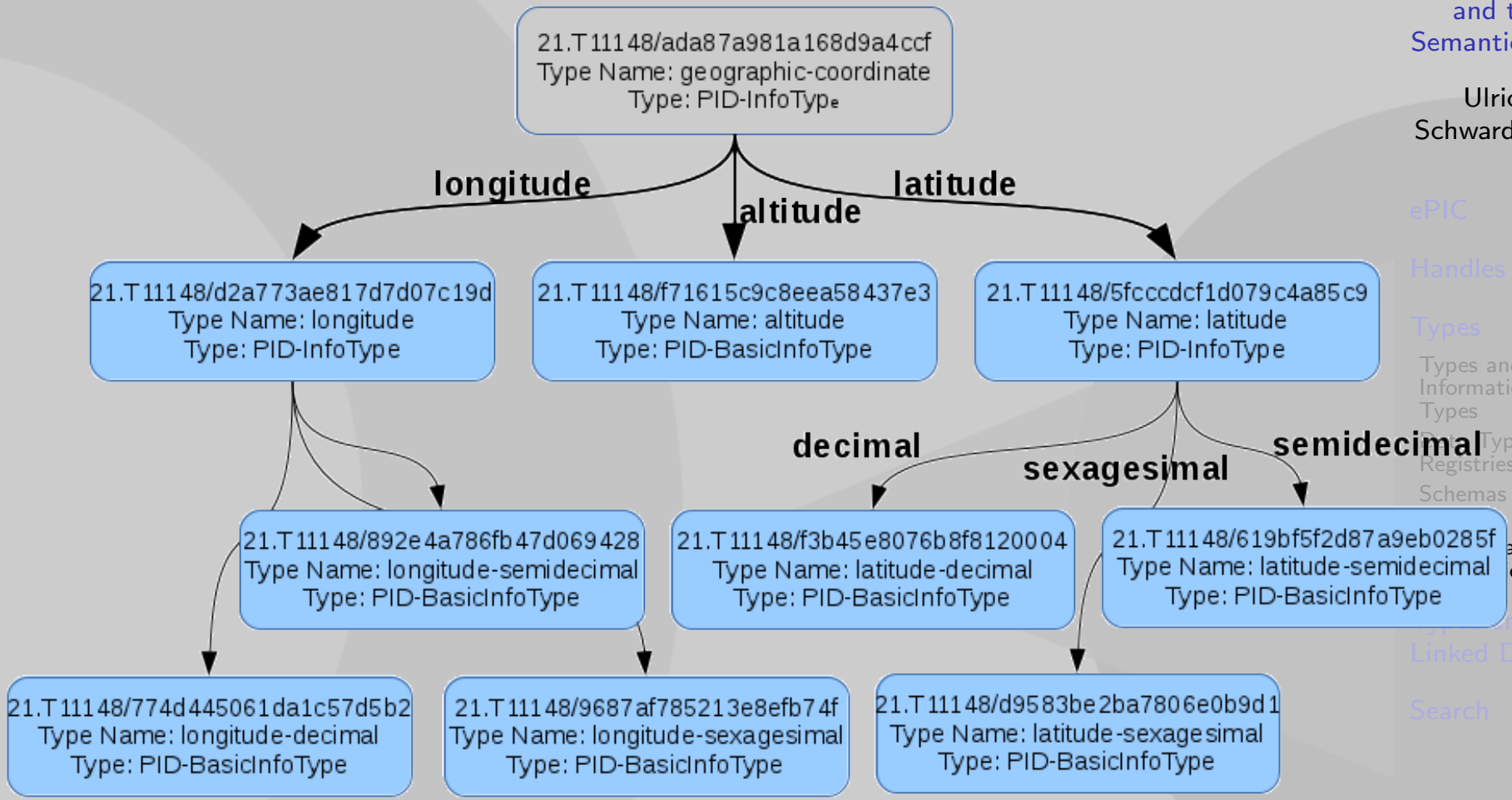
Schemas for Types

- PID Information Types are used to prepare and enable the processing of digital objects.
- This requires high accuracy of the type value,
- at least syntactical correctness.
- Therefore a **schema** for each PIT is needed.
- Manual schema definition is cumbersome and errorprone.
- Automated processes would be of great help.

CC BY

Hierarchies in Metadata

Example: geographic coordinate.



Hierarchical Type Definitions

- makes it easy to define new types
  - concept uses **basic PID info types** and **PID info types**
- in consequence:
  - semantics can be introduced easily by reuse of types under different names
    - `isPreviousVersionOf` and `isNextVersionOf` both refer to `ListOfHandles`
  - definition of complex types is simplified
  - possibility to derive automatically **schemas** for the type values
    - automated server side schema derivation implemented at ePIC DTR

CC BY

Types vs. Linked Data

- An Example of a type: `isPreviousVersionOf` (see: `21.11104/3eaedeaced10be5805d2`).
  - 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 PID of the type (*pid-type*) and the PID of the previous version (*pid-do2*)

This gives a triple:

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

Thus one has a:

subject predicate object

relation with types, given by PID, as predicates.

CC BY

Triples of Types in PID

Type Triple for PID 21.11101/0000-0004-10CA-6

< 21.11101/0000-0004-10CA-6 > < has Handle >	< 21.11101/0000-0004-10CA-6 >
< 21.11101/0000-0004-10CA-6 > < URL >	< http://wwwuser.gwdg.de/~uschwar1/Publikat
< 21.11101/0000-0004-10CA-6 > < 21.T11148/3c6de1b7dd91465d437e : title >	< European Persistent Identifier Consortium -
< 21.11101/0000-0004-10CA-6 > < 21.T11148/1d8764d7dcfda12a864d : creators >	< Tibor Kálmán, Daniel Kurzawe, Ulrich Schwa
< 21.11101/0000-0004-10CA-6 > < 21.T11148/b497a0aad7d4c7179b4f : publisher >	< Scivero >
< 21.11101/0000-0004-10CA-6 > < 21.T11148/508ceef36e27328aebf1 : publicationYear >	< 2012 >
< 21.11101/0000-0004-10CA-6 > < 21.T11148/278d671782892173c3b9 : resourceTypeAttribute >	< Text >
< 21.11101/0000-0004-10CA-6 > < 21.T11148/15dc2832134396bc506e : isContainedIn >	< ["21.11101/0000-0004-0F26-2"] >
< 21.11101/0000-0004-10CA-6 > < 21.T11148/82e2503c49209e987740 : sha-md5-checksum >	< 2042d2e529a97fa5d50ed304a9374b93 >
< 21.11101/0000-0004-10CA-6 > < INST >	< GWDG >
< 21.11101/0000-0004-10CA-6 > < HS_ADMIN >	< [object Object] >

CC BY

Linked Data vs. Types

RDF triples are *post web*:

- object is IRI/URI or literale
- subject and predicate are IRI/URI
  - literales are never subject or predicate
- IRI/URI are fuzzy endpoints with no persistancy
- the trust relation for the semantic of predicates is not explicit

Types are *post DOA (digital object architecture)*:

- object is PID or literale
- subject and predicate are PIDs
  - literales are never subject or predicate
- PIDs are persistent by construction
- DTRs build the trust relation for the semantic of predicates

CC BY

Search in the Handle World

- Handle REST API: only a list of all PIDs of a prefix

```
#> curl -u 'user:pwd' -X GET -H 'Content-Type:
application/json'
'http://hdl.handle.net/api/handles?prefix=21.11101'
```

- ePIC REST API:
  maps **REST GET request string** to **SQL queries**
  - but this provides only a small subset of SQL queries
  - and has no specified SELECT part, matching PIDs are returned

CC BY

ePIC Search

```
#> curl -u 'user:pwd' -X GET -H 'Content-Type:
application/json'
'http://demo.pidconsortium.eu/handles/21.11101/
?21.T11148/b497a0aad7d4c7179b4f=*Scive*
&21.T11148/1d8764d7dcfda12a864d=*Schwardmann*'
0000-0004-0F27-1
0000-0004-0F2E-A
0000-0004-107C-F
0000-0004-10CA-6
```

is internally by the ePIC API translated to

```
SELECT * FROM handles WHERE
(type=21.T11148/b497a0aad7d4c7179b4f AND data LIKE
'Scive') AND (type=21.T11148/1d8764d7dcfda12a864d
AND data LIKE 'Schwardmann')
```


CC BY

Search in the RDF World

SPARQL-Queries:

- describe a subgraph and try to map that subgraph in the knowledge graph
- a query is intentional similar to SQL:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
       ?email
WHERE
  {
    ?person  a          foaf:Person .
    ?person  foaf:name  ?name .
    ?person  foaf:mbox  ?email .
  }
```


CC BY

Translation into the Handle World

use *InfoTypes* as *predicates*:

- given InfoTypes *isPerson*, *hasName* and *hasEmailAddress* for a couple of PIDs:
  - the SPARQL query above returns names and emails of every *PID* with isPerson==True in the query scope (LHS/GHR/whatever)

Question for Handle services:

- provide more search functionality in the REST API or
- implement a SPARQL endpoint?

CC BY

Many Thanks for Your Attention

Contact:

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

Questions ???