Managing Discipline-Specific Metadata Within an Integrated Research Data Management System

Marius Politze[a], Sarah Bensberg and Matthias S. Müller
IT Center, RWTH Aachen University, Templergraben 55, Aachen, Germany

Keywords: Semantic Web, Linked Data, Knowlegde Graph, Service Oriented Architecture, Web Service, Data Repository.

Abstract: Our university intends to improve the central IT-support for management of research data. A core demand is supporting FAIR guiding principles. In order to make research data findable for future research projects, an application for the creation and storage of structured meta data for research data was developed. The created meta data repository enable creating, maintaining and querying research data based on discipline-specific properties. Since large number of meta data standards exist for different scientific domains, technologies from the areas of Linked Data and Semantic Web are used to process and store meta data. This work describes the requirements, the design and the implementation a the meta data application that can be integrated into existing research work flows and gives an overview of technical backgrounds used for creating the meta data repository.

1 INTRODUCTION

Within a research data management system we intend to create IT services allowing researchers to store, retrieve and work with research data. Since research data is fundamental for scientific knowledge, one challenge therefore is the long-term storage and availability for re-use within future research projects. Only preserving the raw data is fairly enough to make it available to future researchers. In order to endorse good data management the FAIR Guiding Principles to support discoverability of scientific data (Wilkinson et al., 2016). These principles place requirements towards both, the actual research data and describing meta data to allow researchers and their peers the retrieval and meaningful interpretation of re-used data.

It is therefore our goal as a university IT-service provider to foster the creation, consolidation and usability of IT services, resources and infrastructure to provide additional value to researchers across scientific disciplines. There is a tension between discipline specific use cases show how tailored, decentralized IT-systems can support scientific processes (Kirsten, Kiel, Wagner, Rühle, and Löffler, 2017; Curdt et al., 2016) and approaches focusing on generic and centralized support of research process not accounting for discipline-specific needs (Van Garderen, 2010; Kraft et al., 2016). Discipline-specific services often provide higher value to the researcher, however generic systems offer higher scalability. To combine both scalability and individualization we established a distributed infrastructure that offers services from various providers to build a generic system that can be integrated into discipline-specific research processes (Politze, Decker, and Eifert, 2017).

While working with research data, each data set can be processed in different environments that define its visibility and accessibility to researchers and their peers. The domain model formalizes that research data is processed in and transferred between different domains of access from *personal* via *group* to *persistent* (Klar and Enke, 2013). Other models depict research data management as a life cycle with data passing through different phases of *Collection* and *Analysis* to *Preservation* and *Re-Use* (“Research Data Lifecycle”, 2012). The goal of a research data management system thus is to provide tools for the researchers to allow transitions between phases (Schmitz and Politze, 2018). Both models share the assumption that data is used more actively within initial phases. Thus, more knowledge about data sets is available: When research data is produced meta information like authors, associated research projects or used instruments are easily available. As the data

[a] https://orcid.org/0000-0003-3175-0659

Politze, M., Bensberg, S. and Müller, M.
Managing Discipline-Specific Metadata Within an Integrated Research Data Management System.
DOI: 10.5220/0007725002530260
In *Proceedings of the 21st International Conference on Enterprise Information Systems (ICEIS 2019)*, pages 253-260
ISBN: 978-989-758-372-8

passes through the life cycle, this implicit knowledge is often not transferred and is then lost for peers re-using the data. In accordance to the FAIR guiding principles, it is our goal to retain this information. Meta data associated to data sets can provide immediate value if used by peers within a research group. Additionally, meta data allows data sets to cross domain boundaries to long-term storage and persistence and forms the basis for re-use of data in future contexts.

Bibliographic information like authorship, descriptions or licensing are widely standardized by meta data schemas like the *DataCite Metadata Schema* (DataCite Metadata Working Group, 2017). Looking at the diverse disciplines of research at the university, shows that each of these disciplines poses its own discipline-specific requirements towards meta data. Recording discipline-specific meta data within an integrated system used across disciplines thus demands a high flexibility from data models. Models provided by linked data applications give exactly this flexibility by describing digital (and also real world) objects using triples of the form $(subject, predicate, object)$.

Based on a previously developed linked data model (Politze and Decker, 2016), our goal thus is to provide an integrated application allowing researchers of various disciplines to describe their research data using discipline-specific meta data schemas and trace their data sets along the research data life cycle (RDLC). The application should integrate into the existing research data management system and research processes within research groups. Considering the data stored in this kind of repository, we therefore aim at building a queryable knowledge graph, as proposed by Galkin et al. (Galkin, Auer, Vidal, and Scerri, 2017) or Decker (Decker, 2017), to make meta data accessible and interoperable and thus make research data findable within and across organization boundaries.

2 SOFTWARE ARCHITECTURE

The application and work flows are designed to fit into an existing decentralized IT-service landscape. Even though services are provided by different organizations within the university, they share a common understanding and nomenclature in terms of the supported business processes. This is achieved by introducing abstraction layers (see Fig. 1). Individual services are merged to consistent minimal valuable processes that are in turn used by different applications. The connection and integration of the systems

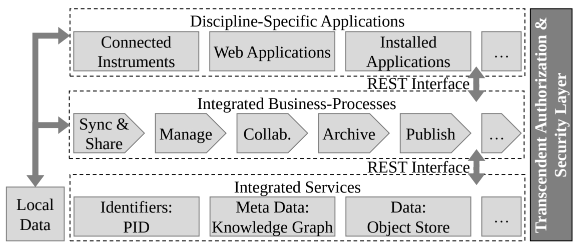


Figure 1: Overview of the integrated system landscape supporting research processes.

is governed by a transcendent layer providing means for security and authorization.

2.1 Data Model

The Resource Description Format (RDF) data model defines triples of the form $(subject, predicate, object)$ that form graphs of connected entities. In order to identify data sets within the graph they need an Internationalized Resource Identifiers (IRI) compatible identifier, that allows the flexibility to follow the data along the RDLC. While several ways to create such Persistent Identifier (PID) exist, the application uses the EPIC service that in turn is based on the Handle system (Kálmán, Kurzawe, and Schwardmann, 2012). EPIC allows creation of PIDs for data set as soon as they are created. Additional key-value-pairs that are part of the PIDs' meta data allow further tracing the state of data sets. Following this idea, every data set is identified by the URI of the PID connecting data and meta data. Data sets thus are nodes in a graph of meta data forming a knowledge graph of linked resources (Bizer, Heath, and Berners-Lee, 2009). RDF offers several serialization formats like Turtle (Beckett, Berners-Lee, Prud'hommeaux, and Carothers, 2014), RDF/XML (Gandon and Schreiber, 2014) and JSON-LD (Sporny, Longley, Kellogg, Lanthaler, and Lindström, 2014) that are well suitable for conveying machine readable information. It is unlikely that researchers author meta data in these formats due to their complexity. To provide researchers with a more comprehensive interface, matching discipline-specific requirements and restricted the expressiveness, for each discipline this is done by gradually identifying the following entities[1]:

Vocabularies provide sets of terms and their relationships. Relations are often hierarchical but can also present other structures. Terms refer to digital or real

[1]Unfortunately these entities are ambivalently used throughout literature and contexts. Within this paper these definitions are followed.

world objects or any other type of concept, including but not limited to concepts like numbers, texts or dates.

Schemas are sets of properties that make up the meta data and their relationships. A property is an attribute that describes a certain detail of an entity. The range of each these properties is a vocabulary.

Standards additionally define a set of required attributes from one or more schemas in order to fulfill the standard.

(Application) Profiles select properties from multiple schemas and combine them to discipline-specific templates. Profiles may further narrow the range of properties or define default values used in provided interfaces. Often profiles are used to extend standards with additional properties.

Meta Data (Sets) then are instances of a profile describing a real world object or a digital data set.

By creating application profiles based on common schemas and standards, meta data remain compatible and form a consistent graph. There exist several languages to define requirements towards RDF graphs like the SHACL (Knublauch and Kontokostas, 2017) or SHEX (Prud'hommeaux, Boneva, Labra Gayo, and Kellogg, 2017) and these are in principle feasible for building profiles. Without loss of generality, a first approach for profiles are described in RDF. RDF already supplies general properties like `label` and `range`, but additional properties need to be defined to support consistency or increase usability of generated interfaces, for example: `position` defines the order in which properties appear, `calculatedValue` defines a template for a default value.

As an illustrative and simplified example, the code below shows such a profile that combines properties from two meta data schemas, Dublin Core (ISO, 2017) and the Core Scientific Metadata Model (Matthews and Fisher, 2013):

```
dc:creator
  a owl:AnnotationProperty ;
  md:calculatedValue "{ME}" ;
  md:position 1;
  rdfs:label "Lab Technician"@en ;
  rdfs:range rdfs:Literal .
dc:title
  a owl:AnnotationProperty ;
  md:position 2;
  rdfs:label "Description"@en .
dc:subject
  a owl:AnnotationProperty ;
  rdfs:range <http://udcdata.info/029653> ;
  md:position 3;
  rdfs:label "Subject Area"@en .
:solute
  rdfs:subPropertyOf
↪  csmd:sampletype_molecularFormula ;
  a owl:AnnotationPropery ;
  md:position 4;
  rdfs:label "Solute"@en .
:solvent
  rdfs:subPropertyOf
↪  csmd:sampletype_molecularFormula ;
  a owl:AnnotationPropery ;
  md:position 5;
  rdfs:label "Solvent"@en .
```

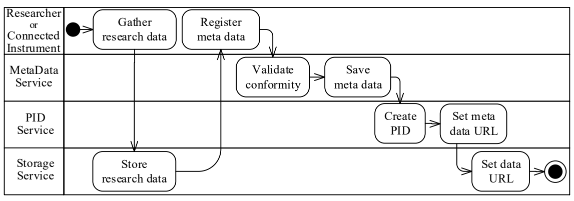


Figure 2: Integration of meta data management in a research data management process.

The code defines meta data properties e.g. `creator`. Some attributes of a property are defined or overridden in the profile: `calculatedValue`, `label` and `range`. The `label` for the property is overridden to be `Author` instead of `Creator`. The range is defined as `Literal`, meaning any kind of plain text. The `calculatedValue` is converted at application run time to the name of the user as a default value for the property. In the case of `subject` the range is defined by referencing a fixed sub set of terms from the universal decimal classification vocabulary. `solute` and `solvent` finally define two discipline-specific properties used to describe conditions chemical experiment.

To satisfy requirements of researchers for discipline-specific support, profiles need to be carefully crafted for each discipline to select adequate meta data properties. Researchers therefore are to be guided by librarians and information scientists before using the application. On the long run this allows the creation of a consistent ontology the super set of all profiles, schemas and vocabularies.

Each data set is assigned a PID allowing resolution using the handle system. As such, PIDs can be transferred between applications and are then enriched using application specific capabilities. A process, as shown in Fig. 2, is established within the integrated research data management system. This process allows distributed handling of data sets. Applications only take a specific role like handling data or meta data. Applications store their states within the PIDs' meta data by which it becomes accessible in the process. Allowing to trace data sets across different phases of the RDLC.

2.2 Functional Requirements

The application will be prototyped as an HTML5 web application. While this provides an easy-to-use way for researchers to manually register meta data, future use-cases should instead use a RESTful API that allows automation and integration into digital research work flows. Nevertheless, the prototype will use AJAX-Requests and thus the same API intended for future applications. Depending on the affiliation of the researcher, different profiles can be selected. These are used to create a user interface (UI) and to validate the input provided by the researcher. Additionally, to the entered meta data, the application also records the users' affiliation and selected profile. The API and prototype therefore need to support the following use cases:

F1: Retrieve Available Profiles: The application should present the user a list of the available profiles. The available profiles depend on the affiliation of the user.

F2: Save New Meta Data Set: The application should validate meta data according to a profile and save them to the data store. Meta data is converted from a simplified JSON serialization to RDF. Data sets are associated with a PID that identifies the data set.

F3: Metadata Visibility: A level of visibility for meta data allows users to make meta data publicly findable, to keep it for a research group or just privately for themselves.

F4: Show All Own Meta Data Sets: All meta data sets created by the user can be retrieved by the application.

F5: Edit Stored Meta Data Set: User who created a meta data set can also edit the meta data. If the meta data set is visible all researchers within the research group should be able to edit the meta data.

F6: Query Stored Meta Data Sets: Researchers can search the data store to find data sets according to a query. Queries should allow searching within meta data properties and account for hierarchical relationships defined by vocabularies. As in F2, the API should be based on a comprehensive JSON serialization that is translated to conform the RDF data store.

F7: Suggestions for Vocabulary Ranges: While searching and editing meta data users may require suggestions for ranges defined by a vocabulary. The application needs to give human readable suggestions according to ranges defined by profiles.

F8: Render Meta Data form based on Schema: The prototype should provide the functionality to create an input form from a profile as an easy-to-use way to store meta data.

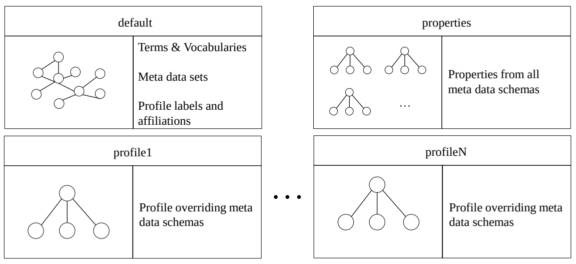

default
Terms & Vocabularies
Meta data sets
Profile labels and affiliations
properties
Properties from all meta data schemas
...
profile1
Profile overriding meta data schemas
...
profileN
Profile overriding meta data schemas

Figure 3: Different contexts encapsulate meta data and definition of profiles.

2.3 Non-functional Requirements

The prototype should ensure that it can be integrated within the RDLC and that meta data sets can be transferred data across domains:

N1: Internationalization: By default all terminology should be presented in English. However, the application should allow switching the language and support multi language profiles and meta data sets.

N2: Compatibility to DCAT: The Data Catalog Vocabulary (DCAT) (Maali and Erickson, 2014) is (according to the definition above) a meta data standard to allow interoperability of published data catalogs. Meta data sets registered by the application should be able to be mapped to DCAT to allow future integration with data repositories.

N3: Dublin Core as Cross Discipline Standard: Well established cross discipline standards should be adhered. For example DCAT and Data Cite reuse fields from the Dublin Core meta data schema (ISO, 2017). Assuring that meta data is compatible to these standards allows seamless integration with applications in other domains of the RDLC.

2.4 Database Model

For the database management system Virtuoso is used as it allows to efficiently store and retrieve RDF triples. Virtuoso offers a SPARQL ("SPARQL 1.1 Overview", 2013) endpoint to query or manipulate the stored triples. Within one database instance Virtuoso supports storing multiple graphs. This property of Virtuoso is used to save different profiles as shown in Fig. 3.

Queries concerning meta data are run against the `default` graph holding information about meta data, definitions of terms, vocabularies and labels of defined profiles. Additionally, the default graph holds information about the association of profiles with affiliated research groups.

A `properties` graph contains the definition of all properties from schemas and therefore forms a super set of all defined profiles. An explicit record of all properties, labels and potential ranges defined by any of the profiles is necessary to allow more efficient computation of suggestions needed for query interfaces.

Every profile is an additional graph. This gives the flexibility to re-use properties from the `properties` graph but also allows narrowing down ranges or overriding of properties for discipline-specific applications. All profiles are associated with a URI identifying their current version. The set of profiles therefore forms a vocabulary in the `default` graph. Likewise, also affiliations are defined as a vocabulary to define the ownership of profiles and meta data.

3 IMPLEMENTATION

To allow integration within already existing digital research work flows, the main focus of the application lies on the implementation of a RESTful API. This API provides a discipline specific interface for the researchers and translates calls to RDF and SPARQL accordingly.

3.1 RESTful API

To create new meta data, the API endpoint `Create` accepts the meta data in JSON-LD as well as a simplified JSON serialization. In the simplified format all triples within the meta data are assumed to have the data set as subject. If no PID is specified as an URL for the data set a new PID is created and used. Within the simplified JSON serialization triples are therefore reduced to pairs of $(predicate, object)$. Rather than using the URIs defining the predicates, substitution of properties by the labels defined in the profile allows submitting meta data sets as simple JSON key-value-pairs. Basing on the profile in the example above the endpoint accepts a document of the form:

```
{
    "Description": "Solving salt in water",
    "Lab Technician": "John Doe",
    "Subject Area": "http://udcdata.info/030042",
    "Solute" : "NaCl",
    "Solvent" : "H2O"
}
```

If the provided values exactly match with the URI of an object within the range of the property, this object is used. Otherwise, the back end will then retrieve the most likely properties based on similarity according to the labels by querying the graph of the profile using a SPARQL query:

```
SELECT ?s WHERE {
    GRAPH <profileN> {
        ?s rdf:label ?label .
        FILTER REGEX(STR(?label), "Value", "i") .
    }
}
```

In order to retrieve a list of all available schemas, the `GetAllProfiles` endpoint retrieves a list of profiles available for the affiliation of the user. The `GetProfile` endpoint then retrieves a single profile with full information about all used properties.

The API endpoint `GetAll` allows to retrieve all stored meta data sets. The method works uses an additional properties to assess the affiliation and visibility of the meta data. To achieve this behaviour the back end translates the request into a SPARQL query:

```
SELECT ?s ?title ?author WHERE {
    GRAPH <default> {
        ?s dc:title ?title .
        ?s dc:creator ?author .
        ?s md:hasOwner ?owner .
        ?s md:publishMetadata ?visibility .
        ?s md:hasAffiliation ?affiliation .
        FILTER (
            ?owner = "#U#" ||
            ?visibility = md:metadataIsPublic ||
            ?visibility = md:metadataIsProtected &&
            ?affiliation = #A#
        )
    }
}
```

Where the placeholders `#U#` and `#A#` are replaced with identifications of the user and the affiliations. The `GetAll` endpoint then provides URIs of all described data sets visible for the user. Retrieval of a single meta data set identified by its URI is then performed by the `Get` endpoint.

Especially for vocabulary ranges, it is necessary to get valid terms for a property. The `GetRange` endpoint therefore allows to retrieve those terms matching a query. Especially for hierarchies vocabularies sometimes use different ways to describe the relationship between the terms. In order to resolve hierarchies `GetRange` uses the following strategies:

RDF Schema Class Hierarchies: Given a class C all other classes $?c$ that transitively satisfy the condition `?c rdfs:subClassOf C` are within the range.

RDF class instantiations or DCMI Abstract Model members: Given a class C all terms $?t$ that satisfy one of the conditions `?t rdf:type C` or `?t dcam:memberOf C` are within range.

Simple Knowledge Management Organization System (SKOS) Instances (Miles and Bechhofer, 2009): Given an `skos:Concept` C all other terms $?t$ transitively satisfying the the condition `?t skos:narrower C` are within range.

In order to transitively resolve matching terms, the SPARQL query additionally needs the `TRANSITIVE` option. In the case of SKOS the query with placeholders for the given Concept `#C#` and a query `#Q#` looks as follows:

```
SELECT DISTINCT ?subject STR(?label) AS ?label WHERE{
    ?subject skos:prefLabel ?label
    {
        SELECT ?subject ?y WHERE {
            GRAPH <default> {
                ?subject skos:broader ?y
            }
        }
    }
    OPTION(TRANSITIVE, t_in(?y), t_out(?subject))
    FILTER(?y = #C#) .
    FILTER regex(STR(?label), "#Q#", "i")
}
```

After storing meta data sets in the repository, researchers furthermore are able to retrieve data sets based on the provided meta data. The `Find` endpoint queries both, meta data sets created by the user and public meta data sets according to their visibility and retrieves a list of matching meta data sets according to a query. This query can either be a full text search or conform to a query syntax that allows targeting single meta data fields. The placeholder `#Q#` creates such a full text query:

```
SELECT ?s ?title ?author WHERE {
    GRAPH <default> {
        ?s dc:title ?title .
        ?s dc:creator ?author .
        ?s ?p ?o .
        FILTER REGEX(STR(?o), "#Q#", "i") .
    }
}
```

A little more complex queries can be performed using a JSON query language supplying labels and respective values in the form

```
{
    "property1": "...",
    "property2": "..."
}
```

The algorithm for matching properties that was already discussed for the `Create` endpoint is used to select properties from the profile and a dynamic query is created to target the specified properties. At the moment the `Find` endpoint only allows logical conjunction of properties. More advanced queries need an extension of the simplified query language or require the user to formulate them directly in SPARQL.

To build an interface for the user that allows building this kind of queries, the `GetProperties` endpoint retrieves a list of properties from the `properties` graph according to a query. Since properties can have multiple labels within different profiles, all labels are retrieved, however, the label provided by the meta data schema is highlighted.

Table 1: RDF ranges mapped to HTML5 field types.

RDF Range	HTML5 Type
`rdfs:Literal`	`text`
`xml:dateTime`	`date`
`md:metadataVisibility`	`radio`
none	`text`
other	`select`

The requirements towards internationalization result in an optional parameter to supply the language for all endpoints discussed above. Since RDF supports internationalization, the provided language, for example `en` or `de`, can then be directly passed to the data base within a sparql statement. RDF allows the specification of labels with language classifiers:

```
dc:title
  a owl:AnnotationProperty ;
  rdfs:label "Title"@en, "Titel"@de .
```

When accesing labels from a SPARQL query a filter can be supplied to access only labels of a specific language:

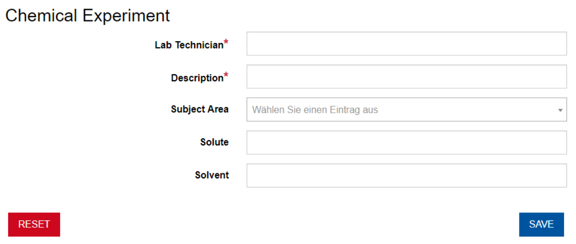
```
SELECT STR(?label)
WHERE {
    ?s rdfs:label ?label .
    FILTER (LANG(?label) = "en")
}
```

3.2 HTML5 Prototype

With the endpoints as a back end the prototype is required to build a presentation layer that can be easily used by researchers to manually register data sets to the system. Depending on `label` and `range` of the properties selected in the profile, the UI should present different input fields. For presentation within an web application Table 1 defines the mapping between `range` and HTML5 fields `type`.

Especially the fields of the type `select` require dedicated attention. These fields present the researcher suggestions based on the range of the property. Each term is therefore identified by its URI that can be used to link the term definition within the meta data set. Instead of displaying the URI to users its label should be used. Researchers need to be able to submit a query to find a desired term based on its label. This is done using the `GetRange` endpoint. A fully rendered meta data form is shown in Fig. 4.

The prototype also features a basic search interface allowing researchers to retrieve stored meta data sets. The interface lets users pick from a list of available properties and therefore allows building a query that can be processed by the `find` endpoint. The resulting meta data sets and their URIs are then displayed to the users.



Chemical Experiment

Lab Technician*	
Description*	
Subject Area	Wahlen Sie einen Eintrag aus
Solute	
Solvent	

Figure 4: Screenshot of genreated UI for a meta data schema.

Table 2: Block coverage by unit tests.

Area	Coverage	
SPARQL	95.92%	47/49
RDFWrapperSchema	100.00%	68/68
RDFWrapperMetadata	91.32%	442/484
API	85.50%	171/200
MetadataSchema	94.44%	34/36
Metadata	92.65%	189/204
Total	91.35%	951/1041

4 FIRST EVALUATION RESULTS

The back end application has been tested using an extensive set of automated white box tests. The test cases have been developed for various external and internal methods and cover both error and normal cases. Table 2 shows that with 91.35% block coverage the current testing methodology achieves satisfying results.

The biggest draw back of the current application is the necessity of defining application profiles in RDF. While RDF powers the flexibility of the application it has shown to be a quite complex task to find adequate meta data schemas and use them to build the necessary profiles. Currently this is done by a joint team of the university library and the IT Center forming a service unit to support the researchers. To further elaborate the application it is necessary to at least lower the threshold of building profiles based on available shemas without in depth knowledge of the underlying semantic data model.

To allow interoperability with other repositories it is possible to map the registered meta data to the data model of DCAT. DCAT therefore defines three main classes: *Catalog*, *Dataset*, *Distribution* and *CatalogRecord* that can be mapped as follows:

Catalog: The application defines multiple catalogs. Each researcher manages a private catalog. By associating each meta data set with a affiliation a catalog for each organization is created. Additionally, there is the public catalog that contains all meta data sets that are publicly visible. A meta data set can be part of multiple catalogs.

CatalogRecord: For each meta data set some properties like the affiliation and user are automatically assessed. This information maps to catalog records.

Dataset: All properties provided by the researcher form the data set. This includes minimal information like author and title but also other discipline-specific properties defined in the profile.

Distribution: The PID used as an identifier allows resolving the research data. The integrated character of the system allows gathering necessary information. If the data is accessible this can be retrieved from additional properties of the PID.

5 CONCLUSION

The presented application is a building block for a continuous support of the RDLC. Central leverage point for the integration into the research work flow is the usage of PID identifiers throughout the research process. PIDs and meta data are therefore not only used as a virtual reference for the data set, but are also used to store references to the data set and directly query and retrieve them using standardized and open protocols within the integrated research data management system at our university.

By setting a minimal requirement towards the profiles, the application fosters FAIR guiding principles: Centrally collecting meta data and making public meta data sets queryable by local and external researchers using simplified JSON and complex, standardized SPARQL endpoints makes the data sets findable and meta data accessible. All data sets are additionally assigned a PID to clearly identify them and to be able to reference data sets throughout the research process. Using the linked data model based on RDF also allows sharing and distributing profiles and meta data schemas according to the FAIR guiding principles.

The application for creating and querying meta data was successfully launched with several pilot users at our university and is slowly introduced to other research groups. It currently fulfills the discussed requirements. Discipline-specific profiles that base on meta data schemas are formulated in RDF and are then made available to the researchers via the RESTful API and user interface. Researchers can make use of these easy-to-use interfaces to integrate the registration of meta data sets at early phases within the RDLC. We have shown that our approach to save and manage discipline-specific meta data within an discipline-agnostic repository and database can be successfully implemented as central and scalalabe service at our university.

REFERENCES

Beckett, D., Berners-Lee, T., Prud'hommeaux, E., & Carothers, G. (2014). RDF 1.1 Turtle. W3C. Retrieved June 10, 2018, from https://www.w3.org/TR/turtle/

Bizer, C., Heath, T., & Berners-Lee, T. (2009). Linked Data - The Story So Far. *I*nternational Journal on Semantic Web and Information Systems, 5(3), 1–22. doi:10.4018/jswis.2009081901

Curdt, C., Hoffmeister, D., Jekel, C., Udelhoven, K., Waldhoff, G., & Bareth, G. (2016). Implementation of a centralized data management system for the CRC Transregio 32 'Patterns in Soil-Vegetation-Atmosphere-Systems'. In C. Curdt & C. Wilmes (Eds.), *P*roceedings of the 2nd Data Management Workshop (pp. 27–33). Kölner Geographische Arbeiten. doi:10.5880/TR32DB.KGA90.6

DataCite MetadataWorking Group. (2017). DataCite Metadata Schema Documentation for the Publication and Citation of Research Data v4.1. doi:10.5438/0014

Decker, S. (2017). Rethinking access to Scientific Knowledge: Knowledge Graphs. Retrieved February 3, 2018, from https://www.linkedin.com/pulse/rethinking-scientific-knowledge-graphs-stefan-decker/

Galkin, M., Auer, S., Vidal, M.-E., & Scerri, S. (2017). Enterprise Knowledge Graphs: A Semantic Approach for Knowledge Management in the Next Generation of Enterprise Information Systems. In *P*roceedings of the 19th International Conference on Enterprise Information Systems (pp. 88–98). doi:10.5220/0006325200880098

Gandon, F., & Schreiber, G. (2014). RDF 1.1 XML Syntax. W3C. Retrieved June 10, 2018, from http://www.w3.org/TR/rdf-syntax-grammar/

ISO. (2017). Information and documentation - The Dublin Core metadata element set. Geneva, Switzerland: ISO.

Kálmán, T., Kurzawe, D., & Schwardmann, U. (2012). European Persistent Identifier Consortium - PIDs für die Wissenschaft. In R. Altenhöner & C. Oellers (Eds.), *L*angzeitarchivierung von Forschungsdaten (pp. 151–164). Berlin, Germany: Scivero Verl.

Kirsten, T., Kiel, A., Wagner, J., Rühle, M., & Löffler, M. (2017). Selecting, Packaging, and Granting Access for Sharing Study Data. In M. Eibl & M. Gaedke (Eds.), INFORMATIK 2017: Digitale Kulturen (pp. 1381–1392). GI Edition Lecture Notes in Informatics Proceedings (LNI). doi:10.18420/in2017_138

Klar, J., & Enke, H. (2013). Projekt RADIESCHEN: Rahmenbedingungen einer disziplinübergreifenden Forschungsdateninfrastruktur, Report "Organisation und Struktur". doi:10.2312/RADIESCHEN_005

Shapes Constraint Language (SHACL). (2017). W3C. Retrieved June 10, 2018, from https://www.w3.org/TR/shacl/

Kraft, A., Razum, M., Potthoff, J., Porzel, A., Engel, T., Lange, F., . . . Furtado, F. (2016). The RADAR Project - A Service for Research Data Archival and Publication. *I*SPRS International Journal of Geo-Information, 5(3), 28. doi:10.3390/ijgi5030028

Data Catalog Vocabulary (DCAT). (2014). W3C. Retrieved June 10, 2018, from http://www.w3.org/TR/vocabdcat/

Matthews, B., & Fisher, S. (2013). CSMD: the Core Scientific Metadata Model. Retrieved June 10, 2018, from http://icatproject-contrib.github.io/CSMD/csmd-4.0.html

SKOS Simple Knowledge Organization System Reference. (2009). W3C. Retrieved June 10, 2018, from https://www.w3.org/TR/skos-reference/

Politze, M., & Decker, B. (2016). Ontology Based Semantic Data Management for Pandisciplinary Research Projects. In C. Curdt & C. Wilmes (Eds.), *P*roceedings of the 2nd Data Management Workshop. Kölner Geographische Arbeiten. doi:10.5880/TR32DB.KGA96.10

Politze, M., Decker, B., & Eifert, T. (2017). pSTAIX - A Process-Aware Architecture to Support Research Processes. In M. Eibl & M. Gaedke (Eds.), INFORMATIK 2017: *D*igitale Kulturen (pp. 1369–1380). GI Edition Lecture Notes in Informatics Proceedings (LNI). doi:10.18420/in2017_137

Shape Expressions Language 2.0. (2017). Retrieved from http://shex.io/shex-semantics/

Research Data Lifecycle. (2012). Retrieved February 13, 2019, from https://www.ukdataservice.ac.uk/manage-data/lifecycle

Schmitz, D., & Politze, M. (2018). Forschungsdaten managen – Bausteine für eine dezentrale, forschungsnahe Unterstützung. o-bib. Das offene Bibliotheksjournal, 5(3), 76–91. doi:10.5282/o-bib/2018H3S76-91

SPARQL 1.1 Overview. (2013). W3C. Retrieved June 10, 2018, from https://www.w3.org/TR/sparql11-overview/

Sporny, M., Longley, D., Kellogg, G., Lanthaler, M., & Lindström, N. (2014). W3C. Retrieved June 10, 2018, from https://www.w3.org/TR/json-ld/

Van Garderen, P. (2010). Archivematica: Using microservices and open-source software to deliver a comprehensive digital curation solution. In A. Rauber (Ed.), *P*roceedings of the 7th International Conference on Preservation of Digital Objects (pp. 145–149). Books@ocg.at. Vienna, Austria: Österreichische Computer Gesellschaft.

Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J. J., Appleton, G., Axton, M., Baak, A., Mons, B. (2016). The FAIR Guiding Principles for scientific data management and stewardship. *S*cientific data, 3. doi:10.1038/sdata.2016.18