RESEARCH PAPER

Evaluation of Application Possibilities for Packaging Technologies in Canonical Workflows

Thomas Jejkal[1†], Sabrine Chelbi[1], Andreas Pfeil[1] & Peter Wittenburg[2]

[1]Karlsruhe Institute of Technology, Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Germany

[2]Max Planck Computing and Data Facility, Gießenbachstraße 2, 85748 Garching, Germany

Keywords: Canonical Workflow Framework for Research; Packaging technologies; Research data collections; Packaging formats

Citation: Jejkal, T., et al.: Evaluation of application application possibilities for packaging technologies in canonical workflows. Data Intelligence 4(2), 372-385 (2022). doi: 10.1162/dint_a_00137

Received: September 17, 2021; Revised: November 26, 2021; Accepted: February 5, 2022

ABSTRACT

In Canonical Workflow Framework for Research (CWFR) "packages" are relevant in two different directions. In data science, workflows are in general being executed on a set of files which have been aggregated for specific purposes, such as for training a model in deep learning. We call this type of "package" a data collection and its aggregation and metadata description is motivated by research interests. The other type of "packages" relevant for CWFR are supposed to represent workflows in a self-describing and self-contained way for later execution. In this paper, we will review different packaging technologies and investigate their usability in the context of CWFR. For this purpose, we draw on an exemplary use case and show how packaging technologies can support its realization. We conclude that packaging technologies of different flavors help on providing inputs and outputs for workflow steps in a machine-readable way, as well as on representing a workflow and all its artifacts in a self-describing and self-contained way.

1. INTRODUCTION

In the position paper Canonical Workflow Framework for Research (CWFR) [1] the idea was presented, to increase the efficiency and reproducibility of research by using workflows of parameterizable, reusable, canonical steps to describe and execute recurring patterns in research. At execution time, the state of each step is preserved in so called CWFR State Digital Objects (CWFR-DO) which are supposed to be FAIR

† Corresponding author: Thomas Jejkal (Email: thomas.jejkal@kit.edu; ORCID: 0000-0003-2804-688X).

Digital Objects (FAIR DOs) strongly based on persistent identification and typing in order to realize the well known FAIR principles in a machine-actionable way. Each workflow step may add outputs, e.g., single properties or FAIR DOs referring to data, to a new CWFR-DO also containing all outputs of previous steps. This results in a complete view on the workflow states at each step and also allows to resume the workflow from each step re-using previously obtained results.

In this paper we like to discuss the potential role of packaging technologies in CWFR. By packaging technology we mean a technical solution for aggregating information, e.g., data and metadata, in a possibly self-describing way allowing third parties to extract and reuse these information without in-depth knowledge about the context in which the package was created. Thus, packages can have different important roles in canonical workflows, e.g., to describe aggregations of input or output data or to describe a fully orchestrated workflow including all contextual information necessary for its execution. This description can serve as an execution template and can also be enriched by provenance information at execution time. In the following chapter, existing packaging technologies will be evaluated with a focus on their feasibility to fill at least one of the aforementioned roles. Afterwards, possible approaches for applying adequate packaging technologies to CWFR are presented by means of a basic example workflow. Finally, conclusions and an outlook to future work will be given.

2. STATE OF THE ART

In this chapter we give an overview about a selection of packaging technologies and we evaluate their applicability for canonical workflows. There are plenty ways of creating packages or comparable structures available that can be, for the sake of clarity, grouped in two categories:

File-based approaches allow to aggregate resources, typically files, in a defined structure and allow to include metadata for providing further information about the aggregation itself or its contents.

Hierarchical collections have similar characteristics than file-based approached but also offer interfaces for accessing their content remotely.

Based on this categorization, we selected a few packaging technologies on which we like to take a closer look in the following sections. On the one hand, this selection was influence by our own experience. On the other hand, we want to focus on packaging technologies which are independent from a specific platform or a scientific domain they are focussing on.

2.1 File-based Approaches

Packaging formats are the most basic form of file-based packages. They typically reflect local file structures enriched by small amounts of metadata. The metadata is stored in files as part of the package and gives a basic description about the content of the package. Examples for packaging formats are Frictionless Data [2] and BagIt [3].

Frictionless Data initially started as a packaging format for tabular information, e.g., fiscal data, allowing to provide information helping to understand the structure of contained tabular data, e.g., in CSV format. Nowadays, Frictionless Data offers a couple of specifications, which are in principle JSON schemas that can be combined and used to describe custom packages, e.g., containing data resources. This has opened Frictionless Data for additional fields like machine learning where it is used as internal data format for the well-known Kaggle platform [4].

BagIt is specified as a generic packaging format. It was primarily designed to package hierarchical file structures. In the first place, a bag is a local directory containing bag-related key-value metadata encoding information, e.g., BagIt version. The payload is placed in a dedicated sub-directory and a manifest file contains checksums for all payload elements in the bag. For providing additional, descriptive metadata, BagIt offers to include tag files. Tag files are treated the same way as payload, but there is no specific location prescribed where they must be located in the bag, which makes it hard to find tag files containing specific metadata required for a certain purpose. To address this issue, the RDA Working Group on Research Data Interoperability published a recommendation document [5] on an approach on how to create self-describing bags. Thus, the main difference between BagIt and Frictionless Data is, that for BagIt the payload of a bag is supposed to be treated semantically opaque, i.e., the consumer should handle the payload as uninterpreted octets. This makes it hard for third party consumers to identify a specific payload element for a certain purpose.

With Research Object Crate (RO-Crate) [6] there is another candidate in the group of file-based approaches. It relies on schema.org [7] for providing metadata allowing to understand and reuse the content of an RO-Crate but can be flexibly extended to support additional schemata. An RO-Crate is supposed to be self-describing and self-contained. The main elements of an RO-Crate are a JSON-LD metadata file and optionally payload files located relatively to the metadata file or added by reference, which allows an easy implementation on top of existing platforms. An RO-Crate may contain a Website representing its content in a human-readable form. The specification adopts certain schema.org elements to describe and contextualize research data. In addition, RO-Crate adopts the Bioschemas profiles [8], which extends the schema.org vocabulary by additional types to describe for example computational workflows, genes or samples.

2.2 Hierarchical Collections

In contrast to file-based packages we consider hierarchical collections as a dynamic representation of different kinds of resource references in a structured manner.

One generic approach for representing collections is described by the Portland Common Data Model (PCDM) [9]. It allows to model rich hierarchies of collections. PCDM is based on RDF supporting three different elements: Collection, Object and File, with each of them having different kinds of associable information. It supports widely adopted vocabularies and standards like Dublin Core Terms and OAI-ORE [10]. PCDM is implemented for Fedora-based repositories like Hydra [11]. It is also mentioned in the RDA recommendation of the Research Data Collections Working Group [12] pointing out its lack of a generic, machine-readable interface.

In order to fill this gap, the Research Data Collection WG created a generic API specification incorporating other, FAIR DO-related recommendations of the RDA, e.g., persistent identifier support and data typing. From the model perspective, the proposed API is based on two elements: Collections and Members, whereas a collection can also be a member of another collection. Additional properties allow to restrict a collection to members of a specific type, its size or its fixature. Furthermore, licensing and model information can be associated with collections. For member items, their type and ontology can be defined allowing to provide semantic information. Implementing the API specification allowed us to gain some experience and to apply the recommendation to scientific use cases.

3. PROBLEM FORMULATION

In data science, the input of a workflow step is often a set of files, which has been aggregated for specific purposes, such as for training a model in deep learning or processing a set of raw data of a specific kind. The same applies to outputs, which are supposed to be reused in subsequent workflow steps. Figure 1 presents a workflow, which strongly deals with exchanging data structures between workflow steps. This workflow realizes a processing chain of digitized, medieval manuscripts applying an automatic layout analysis, the ingest into a research data repository and the transformation of layout features into annotations, which are supposed to be modified in a manual process by humanities scientists using standardized interfaces at the end of the workflow.

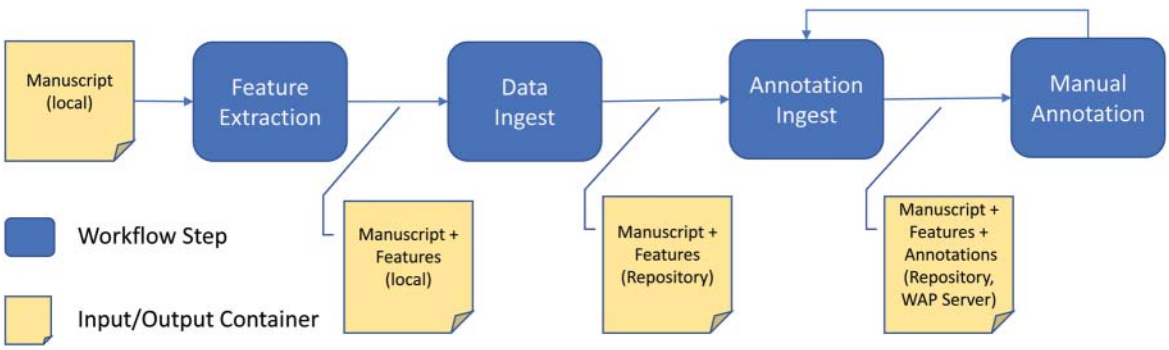

Figure 1. Sample workflow using packages for data exchange between workflow steps.

Figure 1 shows a data processing workflow starting with a local data structure *Manuscript (local)* aggregating all digitized pages of a medieval manuscript as well as descriptive metadata about the entire manuscript in TEI XML format [13]. In the first workflow step, feature extraction is applied to all image files in *Manuscript (local)* using a feature extraction pipeline of several tools. An additional file containing all extracted features in PAGE XML format [14] is created locally and added to the data structure called now *Manuscript + Features (local)*. In the next step, all local data are ingested in a research data repository resulting in *Manuscript + Features (Repository)* now offering Web-resolvable URLs to refer to all elements, e.g., digitized manuscript pages, descriptive metadata and features in PAGE XML. In a final step, annotations following the Web Annotation Data Model [15] are created from PAGE XML and ingested in a Web

Application Protocol Server. Afterwards, references to all annotations are also part of the data structure now called *Manuscript + Features + Annotations (Repository, WAP Server)*. At the end of the workflow, users may add additional annotations or modify existing annotations manually, that are written directly to the WAP Server.

Assuming that this workflow is implemented using canonical steps, which can be reused in different contexts by minor customization, puts some requirements on the aforementioned data structures. These are:

- A machine should be able to decide, whether a connected input can be used by a certain canonical step.
- Elements of a given data structure should be accessible in a machine-readable way and should be remotely resolvable, where possible, also during external workflow execution.
- For data, which is not remotely resolvable, a machine-readable workflow representation with the possibility to include data and software should be supported.

4. APPLYING PACKAGING TECHNOLOGIES FOR DATA AND WORKFLOW EXCHANGE

To meet the requirements described in the previous section, we first want to look at the inputs and outputs of workflow steps starting with the first step, the feature extraction. The easiest way of providing aggregations of data is using packaging formats. Preparing a local directory containing files in a certain structure is something natural. By including additional metadata, the content of a package can be sufficiently described. Due to the assumption that all payload is opaque in BagIt-based packages, we take a closer look at Frictionless Data.

The minimum data package contains a single file *datapackage.json* with metadata about the package itself and its content. Listing 1 shows a metadata file with minimum entries following the Data Package specification of Frictionless Data. Besides these basic human readable properties it is also recommended to add a globally unique identifier to each package in order to be able to refer to and update the package, if required.

```
{
  "name" : "manuscript_1",
  "title" : "Digitized Manuscript 1",
  # Recommended: Add locally or globally unique identifier
  # "id" : "b03ec84-77fd-4270-813b-0c698943f7ce"
  "licenses" : [{
  "name": "ODC-PDDL-1.0",
  "path": "http://opendatacommons.org/licenses/pddl/",
  "title": "ODC Public Domain Dedication and License v1.0"
  }],
  # list of the data resources in this data package
  "resources": [
    { #... resource info described in Listing 2 ... }
  ]
}
```

Listing 1. Minimum content of datapackage.json.

For referencing data, the *resources* element is used. In the minimal case, a resource element contains a *path* property pointing to a file stored relatively to *datapackage.json* or to a URL. In addition, further metadata describing the file format, its media type or even a validation schema can be provided. As a result, the *resources* section might look as shown in listing 2.

```
"resources": [
   {
    "name": "page_1r",
    "path": "page1_r.tiff",
    "title": "Page 1, recto",
    "format": "tiff",
    "mediatype": "image/tiff",
    "bytes": 3456454,
    "hash": "sha1:d6605ede08f4a56aab089f2b8a6447b56739761a",
    },
    {
    "name": "manuscript_metadata",
    "path": "manuscript_metadata.xml",
    "title": "Manuscript Metadata in TEI XML Format",
    "format": "xml",
    "mediatype": "application/tei+xml",
    "bytes": 234542,
    "hash": "sha1:c4632982043b874fdd5486e5115ee60b671f2fb4",
    "schema": "https://tei-c.org/[...]/xsd/tei_lite.xsd"
    }
 ]
```

Listing 2. Resources section of datapackage.json.

Properties like *mediatype*, *bytes*, or *hash* allow the selection of appropriate readers and basic validation. The *schema* property as part of the manuscript metadata resource even allows a validation of the referenced XML document. These properties together with its easy creation makes Frictionless Data well suited for use cases where local files are required as inputs for canonical steps. If this is not required, other options to represent inputs as well as outputs exist. In the context of this paper we will investigate the applicability of hierarchical collections, i.e., the Collection API specification [16] recommended by the RDA Research Data Collections WG, for this purpose. The specification aims to support the concept of FAIR DOs while providing a domain agnostic representation of hierarchical collections. It supports data types as recommended by the RDA Data Type Registries WG [17] and offers a machine-readable interface. In Figure 2 we modelled the data structure *Manuscript + Features (Repository)*, which is the output of the data ingest step shown in Figure 1.

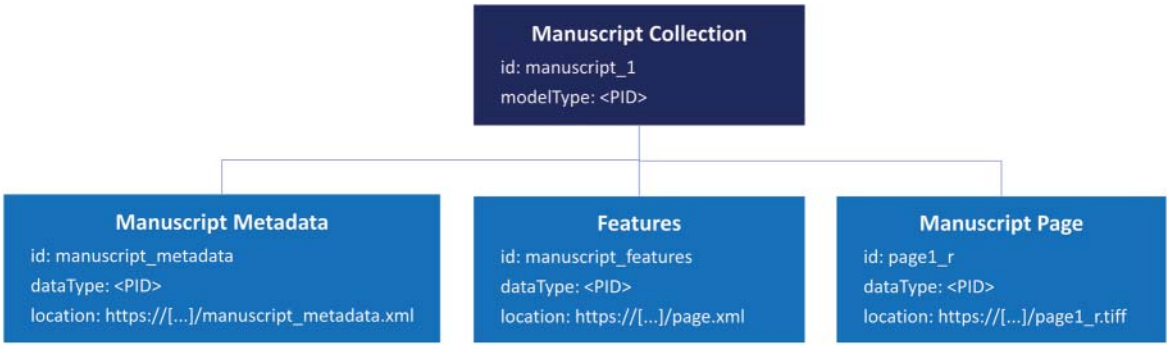

Figure 2. Manuscript + Features (Repository) data structure modelled as collection.

We are using four different elements:

Manuscript Collection: This element serves as collection root aggregating data and metadata of a single manuscript. It has a (local) identifier representing the name of the manuscript, and a property *modelType*, which can be a value from a controlled vocabulary or a PID to uniquely identify the collection model.

Manuscript Metadata: This node is a member item of *Manuscript Collection*. It has a (local) identifier classifying the node as manuscript metadata in a human-readable way, and it has a data type assigned which may classify the referenced content as metadata in TEI XML format.

Features: This member item refers to the extracted features from all manuscript pages obtained during feature extraction. It has a local identifier and a data type assigned which may classify the referenced content as metadata in PAGE XML format.

Manuscript Page: Finally, the collection contain one or more *Manuscript Page* nodes. Figure 2 only shows one exemplary node. Their local identifiers are representing the name of the manuscript page and they have a data type assigned, classifying them to hold e.g., TIFF images.

A data structure following this model can be automatically created by the workflow step using the machine-readable interface. Specific properties, e.g., *modelType* or *dataType*, are assigned using corresponding data type PIDs. Properties like *id* or *location* are filled in during the data ingest. For reasons of reusability it can be decided, whether PIDs are assigned to the entire collection, to member items, to both of them or none at all. In any case, the collection is Web-resolvable if the service is publicly available. One big advantage is, that the collection can be reused by any other workflow step supporting the collections' *modelType*.

By now, we presented two ways on how to provide inputs and outputs to workflow steps: Frictionless Data packages allowing to provide local files including metadata for further description, and Collections supporting Web-resolvable data, typing and a machine-readable interface. In the following we tackle the challenge on how to represent a canonical workflow including input and outputs as well as all workflow

steps and their relationships in a machine-readable way. Within the scope of this paper we like to focus on RO-Crate packages as they are flexible as well as easy to implement.

The central element of an RO-Crate is a metadata file *ro-crate-metadata.json* containing JSON-LD metadata following the schema.org vocabulary, optionally extended by the Bioschemas profile. This file contains a graph of elements, e.g., datasets, contextual information or workflows. Listing 3 shows the root identified by id ./ and an element describing the *ro-crate-metadata.json* file.

```
{ "@context": "https://w3id.org/ro/crate/1.1/context",
  "@graph": [
 {
    "@type": "CreativeWork",
    "@id": "ro-crate-metadata.json",
    "conformsTo": {"@id": "https://w3id.org/ro/crate/1.1"},
    "about": {"@id": "./"}
 },
 {
    "@id": "./",
    "@type": "Dataset",
    "name": "Sample Workflow",
    "description": "This RO-Crate describes an example image processing
       ↪ workflow."
    "hasPart": [
        [...]
      ]
 }
 ]
}
```

Listing 3. Root element description in ro-crate-metadata.json.

Now, we want to describe our workflow, which is done by listing all workflow steps using the *hasParts* element. In our example we have four steps. For reasons of readability we identify them by their human-readable names. The result is presented in listing 4. For each identifier we need a section describing the corresponding step. This section may contain all details we are able to provide, but at least information identifying the canonical step we use, e.g., its PID, and specifying its inputs and outputs.

```
[...]
 "hasPart": [
    { "@id": "#FeatureExtraction" }
    { "@id": "#DataIngest" }
    { "@id": "#AnnotationIngest" }
    { "@id": "#ManualAnnotation" }
 ]
```

Listing 4. References to workflow parts in ro-crate-metadata.json.

Listing 5 shows the section with id *FeatureExtraction*. We define *FeatureExtraction* as an element of type *ComputationalWorkflow*, which is a term from the Bioschemas profile specified by the *conformsTo* property. We also provide a property *url*, which is a PID pointing to a canonical step. However, it is also possible to refer to a software package also described in the same RO-Crate.

```
{
   "@id": "#FeatureExtraction",
   "@type": ["ComputationalWorkflow"],
   "conformsTo": {"@id": "https://bioschemas.org/profiles/
      ↪ ComputationalWorkflow/0.5-DRAFT-2020_07_21/"},
   "name": "Feature extraction",
   "input": [
     { "@id": "manuscript.zip"}
   ],
   "output": [
     { "@id": "#Manuscript_Features_local"}
   ],
   "url": "https://hdl.handle.net/21.T23412/44d1f3241ae31758cf10",
   "version": "1.0.0"
}
```

Listing 5. Description of each workflow step in ro-crate-metadata.jsonfloat.

Finally, we include *input* and *output* elements referring to other ids, again by human-readable names for reasons of readability. Listing 6 shows the definition of both: the input and the output. In contrast to the output, the input is defined as file named *manuscript.zip*. Thus it will be shipped with the RO-Crate allowing a remote workflow execution. Alternatively, the input can also be provided as Web-resolvable URL or PID using the *url* property, e.g., for using a hierarchical collection as input. The output is only identified by an id, which is due to the fact, that the output not yet exists. Finally, input and output contain *additionalType* and *format* properties which can be used to give additional, machine-readable information about the element type and format, preferably as types defined in a Data Type Registry or by a value from a controlled vocabulary.

The remaining steps of the workflow can be defined in the same way. Links between steps via their output can be easily achieved by using an output identifier as input identifier for a subsequent step. There are plenty other options for adding contextual and descriptive information to the RO-Crate depending on its envisioned use and particular requirements.

```
{
   "@id": "manuscript.zip",
   "@type": ["File", "FormalParameter"],
   "conformsTo": {"@id": "https://bioschemas.org/profiles/FormalParameter
      ↪ /0.1-DRAFT-2020_07_21/"},
   "name": "Manuscript in Frictionless Data Format ",
   "additionalType": {"@id": "..."},
   "format": {"@id": "..."}
    },
   {
   "@id": "#Manuscript_Features_local",
   "@type": "FormalParameter",
   "conformsTo": {"@id": "https://bioschemas.org/profiles/FormalParameter
      ↪ /0.1-DRAFT-2020_07_21/"},
   "name": "Manuscript with PAGE XML in Frictionless Data Format",
   "additionalType": {"@id": "..."},
   "format": {"@id": "..."}
}
```

Listing 6. Input and output elements in ro-crate-metadata.json referring to Frictionless Data package.

5. DISCUSSION AND CONCLUSIONS

Summarizing, we can successfully apply packaging technologies to the presented workflow. They help on the one hand for representing different kinds of inputs and outputs of canonical steps, on the other hand to represent a workflow in a self-describing and self-contained way, e.g., for external execution. For providing inputs and outputs we presented the choice between Frictionless Data and the Collection API depending on boundary conditions and requirements. Where Frictionless Data is easy to use, the Collection API offers Web-resolvable and citable representation of data and metadata.

To make this possible for Frictionless Data packages as well, some steps have to be taken, e.g., to make them available as FAIR DOs. If this is not required, data can also be provided together with a reusable workflow representation. For this purpose we considered RO-Crate, which offers many possibilities for describing canonical workflows in JSON-LD for later execution. It uses the rich vocabulary of schema.org extended by Bioschemas profiles. This offers the big advantage, that an RO-Crate can be used as machine- as well as human-readable description of workflows. For the presented workflow, this is a huge benefit as the transitions between workflow steps can be well defined. Furthermore, the workflow can be described as a whole including information about the software version which has been used, e.g., for feature extraction. This allows to quickly identify affected results if there was an issue with a certain software version.

Among other things, the challenge on how to deal with state information collected during workflow execution remains open. What is certain is that none of the evaluated packaging technologies seems to offer a sufficient degree of flexibility to store all kinds of information which may be produced by workflow steps. Especially the question on how to include a primitive value, e.g., *TRUE* to state that an ethical review

has been applied successfully, requires additional investigation. In such cases, a more flexible solution, e.g., using CWFR State Digital Objects, might help but requires further investigation.

However, the application of packaging technologies is not limited to one use case but can be beneficial for canonical workflows in general. Agreeing on a selection of self-describing packaging formats for data exchange could improve the reusability of data between canonical steps on the one hand, and it could extend the applicability of canonical steps to data from other sources on the other hand. The same applies for using packaging technologies to describe workflows in a machine-readable and self-contained way. Here, too, packaging technologies can help to further concretize the idea of canonical workflows and bring it closer to implementation.

ACKNOWLEDGEMENTS

This research has been supported by the research program 'Engineering Digital Futures' of the Helmholtz Association of German Research Centers and the Helmholtz Metadata Collaboration Platform (HMC). We thank all contributers from the Research Data Alliance, the FAIR DO Forum, and HMC as well as potential users for fruitful discussions and feedback.

AUTHOR CONTRIBUTIONS

S. Chelbi (sabrine.chelbi@kit.edu) contributed to the implementation of the Collection API service and wrote its documentation. A. Pfeil (andreas.pfeil@kit.edu) contributed with substantive discussions on how to apply packaging technologies to real use cases. P. Wittenburg (peter.wittenburg@mpcdf.mpg.de) provided major parts of the abstract and the introduction section. Furthermore, all mentioned co-authors contributed by proof-reading and correcting the paper.

REFERENCES

[1] Hardisty, A., Wittenburg, P. (eds.): Canonical Workflow Framework for Research (CWFR)—position paper—version 2, December 2020. Working paper. Available at: https://osf.io/9e3vc/. Accessed 9 December 2021
[2] Barratt, J., Rono, S., Walsh, P.: Frictionless data and data packages. Available at: https://zenodo.org/record/1301152. Accessed 26 July 2021
[3] Kunze, J.: The BagIt file packaging format (V1.0). Available at: https://datatracker.ietf.org/doc/html/rfc8493. Accessed 26 July 2021
[4] Kaggle Inc.: Kaggle: Your machine learning and data science community. Available at: https://www.kaggle.com/. Accessed 26 July 2021
[5] RDA Research Data Repository Interoperability WG: Research data repository interoperability WG final recommendations. Available at: https://zenodo.org/record/2657354. Accessed 26 July 2021
[6] Carragáin, E.Ó., et al.: A lightweight approach to research object data packaging. Available at: https://zenodo.org/record/3250687. Accessed 26 July 2021
[7] W3C schema.org: Community group. Available at: https://schema.org/. Accessed 26 July 2021

[8] Bioschemas profiles. Available at: https://bioschemas.org/profiles/. Accessed 27 July 2021
[9] Tuyl, S., Boock, M., Zhang, H.: Use of the Hydra/Sufia repository and Portland Common Data Model for research data description, organization, and access. Available at: https://hal.univ-lille.fr/hal-01396642. Accessed 27 July 2021
[10] Lynch, C., et al.: The OAI-ORE effort: Progress, challenges, synergies. In: Proceedings of the 7th ACM/IEEECS Joint Conference on Digital Libraries (JCDL '07), p. 80 (2007)
[11] The University of Hull: Digital repository. Available at: https://hydra.hull.ac.uk. Accessed 8 March 2021
[12] Weigel, T., et al.: RDA Research Data Collections WG recommendations. Available at: https://doi.org/10.15497/RDA00022. Accessed 8 March 2021
[13] Ide, N.M., Véronis, J.: Text encoding initiative: Background and contexts. Kluwer Academic Publishers, Dordrecht (1995)
[14] Pletschacher, S., Antonacopoulos, A.: The PAGE (Page Analysis and Ground-Truth Elements) format framework. In: The 20th International Conference on Pattern Recognition, pp. 257–260 (2010)
[15] Ciccarese, P., Soiland-Reyes, S., Clark, T.: Web annotation as a first-class object. In: IEEE Internet Computing 17(6), pp. 71–75 (2013)
[16] Weigel, T.: RDA collections API. Available at: http://rdacollectionswg.github.io/apidocs/#/. Accessed 4 August 2021
[17] Lannom, L., Broeder, D., Manepalli, G.: RDA data type registries working group output. Available at: https://zenodo.org/record/1406127. Accessed 27 July 2021

AUTHOR BIOGRAPHY

Thomas Jejkal studied Computer Science at Baden-Württemberg Cooperative State University. After his studies he was working on the topic of Grid Computing within the framework of the German Grid Initiative D-Grid mainly focussing on distributed computing and Open Grid Service Architecture Data Access and Integration (OGSA-DAI). From the year 2010 he started working on cross-disciplinary software solutions in the field of research data management. Currently, he coordinates the working package "FAIR Data Commons" of the Helmholtz Metadata Collaboration (HMC) Platform responsible for defining and implementing base services and generic processes to harmonize metadata management in the German Helmholtz Association. He is also active in the Research Data Alliance, where he was co-chair of the Research Data Repository Interoperability Working Group until 2017.
ORCID: 0000-0003-2804-688X

Sabrine Chelbi received her Master's degree in Computer Science at Karlsruhe Institute of Technology (KIT). The topic of her master thesis was the design and evaluation of a versioning method for a data repository. In April 2020, she joined the Helmholtz Metadata Collaboration (HMC) platform and she has made her contribution by developing basic services since then.
ORCID: 0000-0002-4480-6116

Andreas Pfeil completed his Master's degree in Computer Science at Karlsruhe Institute of Technology (KIT). His Master's thesis dealt with distance metrics for and the visualization of unexplored, unstructured metadata in the context of historical manuscripts. Now he is working at KIT within the HMC (Helmholtz Metadata Collaboration) platform on the realization of FAIR Digital Objects in the Helmholtz Association.
ORCID: 0000-0001-6575-1022

Peter Wittenburg has a background in electrical engineering, has been working as Technical Director at the Max Planck Institute for Psycholinguistics for many years and acted as member of the IT Advisory board of the president of the MPS. The Max Planck Institute was from the beginning focusing on digital technologies to understand the functioning of the brain with respect to language processing. The institutes need for getting access to data from other institutes to feed the stochastic engines they applied rather early led him to become an expert in building data/research infrastructures. He was responsible for the technological aspects of three large international and European research infrastructures: DOBES, CLARIN and EUDAT. In this function he understood that data work across silos is highly inefficient and that harmonisation and standardisation is required to improve the situation. This was the reason that he co-founded the Research Data Alliance in 2013 and the FAIR Digital Object Forum in 2019.
ORCID: 0000-0003-3538-0106