Evaluation of Application Possibilities for Packaging Technologies in Canonical Workflows Postprint

Authors: Thomas, Jejkal, Sabrine, Chelbi, Andreas, Pfeil, Peter, Wittenburg, Thomas, Jejkal

Date: 2022-11-28T00:00:00+00:00

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

Full Text

Preamble

RESEARCH PAPER
Title: Evaluation of Application Possibilities for Packaging Technologies in Canonical Workflows
Authors: Thomas Jejkal[1]†, Sabrine Chelbi[1], Andreas Pfeil[1] & Peter Wittenburg[2]
Affiliations:
[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 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 the Canonical Workflow Framework for Research (CWFR), "packages" are relevant in two different directions. In data science, workflows are generally executed on a set of files that have been aggregated for specific purposes, such as 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 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 in providing inputs and outputs for workflow steps in a machine-readable way, as well as in representing a workflow and all its artifacts in a self-describing and self-contained manner.

---

1. INTRODUCTION

In the position paper *Canonical Workflow Framework for Research (CWFR)* [**?**], 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 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—for example, single properties or FAIR DOs referring to data—to a new CWFR-DO that also contains all outputs of previous steps. This results in a complete view of the workflow states at each step and also allows the workflow to be resumed from any step, reusing previously obtained results.

In this paper, we 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 this information without in-depth knowledge about the context in which the package was created. Thus, packages can have different important roles in canonical workflows—for example, 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 of a selection of packaging technologies and evaluate their applicability for canonical workflows. There are plenty of ways to create packages or comparable structures available that can be, for the sake of clarity, grouped into two categories:

File-based approaches allow aggregating resources—typically files—in a defined structure and enable the inclusion of metadata to provide further information about the aggregation itself or its contents.

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

Based on this categorization, we selected a few packaging technologies on which we would like to take a closer look in the following sections. On the one hand, this selection was influenced by our own experience. On the other hand, we want to focus on packaging technologies that are independent from a specific platform or scientific domain they are focusing 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 of the package' s content. Examples of packaging formats are Frictionless Data [**?**] and BagIt [**?**].

Frictionless Data initially started as a packaging format for tabular information—for example, fiscal data—allowing the provision of information to help

understand the structure of contained tabular data, such as in CSV format. Nowadays, Frictionless Data offers several specifications, which are in principle JSON schemas that can be combined and used to describe custom packages—for instance, containing data resources. This has opened Frictionless Data for additional fields like machine learning, where it is used as an internal data format for the well-known Kaggle platform [**?**].

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—for example, BagIt version. The payload is placed in a dedicated subdirectory, and a manifest file contains checksums for all payload elements in the bag. For providing additional descriptive metadata, BagIt offers the inclusion of 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 [**?**] on an approach for 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 as semantically opaque—that is, 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) [**?**], there is another candidate in the group of file-based approaches. It relies on schema.org [**?**] for providing metadata that allows understanding and reusing 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 relative to the metadata file or added by reference, which allows for 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 [**?**], which extend the schema.org vocabulary with 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) [**?**]. It allows modeling rich hierarchies of collections. PCDM is based on RDF, supporting three different elements: Collection,

Object, and File, with each having different kinds of associable information. It supports widely adopted vocabularies and standards like Dublin Core Terms and OAI-ORE [**?**]. PCDM is implemented for Fedora-based repositories like Hydra [**?**]. It is also mentioned in the RDA recommendation of the Research Data Collections Working Group [**?**], 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—for example, 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 restricting a collection to members of a specific type, its size, or its fixity. Furthermore, licensing and model information can be associated with collections. For member items, their type and ontology can be defined, allowing the provision of semantic information. Implementing the API specification allowed us to gain some experience and 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 that has been aggregated for specific purposes, such as 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 [Figure 1: see original paper] presents a workflow that strongly deals with exchanging data structures between workflow steps. This workflow realizes a processing chain of digitized medieval manuscripts, applying 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 [**?**]. 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 [**?**] is created locally and added to the data structure now called *Manuscript + Features (local)*. In the next step, all local data are ingested into a research data repository, resulting in *Manuscript + Features (Repository)*, which now offers Web-resolvable URLs to refer to all elements—for example, digitized manuscript pages, descriptive metadata, and features in PAGE XML. In a final step, annotations following

the Web Annotation Data Model [**?**] are created from PAGE XML and ingested into 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, which are written directly to the WAP Server.

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

1. A machine should be able to decide whether a connected input can be used by a certain canonical step.
2. 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.
3. For data that 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 examine the inputs and outputs of workflow steps, starting with the first step: feature extraction. The easiest way to provide aggregations of data is using packaging formats. Preparing a local directory containing files in a certain structure is a natural approach. 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.

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 relative 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.

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 validation of the referenced XML document. These properties, together with its easy creation, make 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 and outputs exist. In the context of this paper, we investigate the applicability of hierarchical collections—that is, the Collection API specification [**?**] 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 [**?**] and offers a machine-readable interface. In Figure 2 [Figure 2: see original paper], 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 the 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 that 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 that may classify the referenced content as metadata in PAGE XML format.
- **Manuscript Page:** Finally, the collection contains one or more Manuscript Page nodes. Figure 2 only shows one exemplary node. Their local identifiers represent the name of the manuscript page, and they have a data type assigned, classifying them to hold, for example, TIFF images.

A data structure following this model can be automatically created by the workflow step using the machine-readable interface. Specific properties—for example, `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, or not 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 have presented two ways to provide inputs and outputs to workflow steps: Frictionless Data packages, which allow providing local files including metadata for further description, and Collections, which support Web-resolvable data, typing, and a machine-readable interface. In the following, we tackle the challenge of how to represent a canonical workflow—including inputs and outputs as well as all workflow steps and their relationships—in a machine-readable way. Within the scope of this paper, we focus on RO-Crate packages as they are flexible and 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—for example, datasets, contextual information, or workflows. Listing 3 shows the root identified by id `./` and an element describing the `ro-crate-metadata.json` file.

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—for example, its PID—and specifying its inputs and outputs.

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.

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

Finally, we include input and output elements referring to other ids, again by human-readable names for readability. Listing 6 shows the definition of both the input and the output. In contrast to the output, the input is defined as a file named `manuscript.zip`. Thus, it will be shipped with the RO-Crate, allowing remote workflow execution. Alternatively, the input can also be provided as a Web-resolvable URL or PID using the `url` property—for example, for using a hierarchical collection as input. The output is only identified by an id, which is due to the fact that the output does not yet exist. 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 an input identifier for a subsequent step. There are many other options for adding contextual and descriptive information to the RO-Crate depending on its envisioned use and particular requirements.

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, in representing different kinds of inputs and outputs of canonical steps; on the other hand, they enable representing a workflow in a self-describing and self-contained way—for example, 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. While 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 must be taken—for example, 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 significant advantage that an RO-Crate can be used as both a machine-readable and 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 that has been used—for example, for feature extraction. This allows for quickly identifying affected results if there was an issue with a certain software version.

Among other things, the challenge of 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 that may be produced by workflow steps. Especially the question of how to include a primitive value—for example, `TRUE` to state that an ethical review has been applied successfully—requires additional investigation. In such cases, a more flexible solution—for instance, 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 to 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 contributors 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 proofreading 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/IEEE-CS 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 worked on the topic of Grid Computing

within the framework of the German Grid Initiative D-Grid, mainly focusing on distributed computing and Open Grid Service Architecture Data Access and Integration (OGSA-DAI). From 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' s 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 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 and has been working as Technical Director at the Max Planck Institute for Psycholinguistics for many years, acting as a 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 institute' s 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 harmonization and standardization are 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

Note: Figure translations are in progress. See original paper for figures.