ing.grid preprints

SOFTWARE DESCRIPTOR

h5RDMtoolbox - A Python Toolbox for FAIR Data Management around HDF5

Matthias Probst [1]
Balazs Pritz [1]

1. Institute for Thermal Turbomachinery, Karlsruhe Institute of Technology, Karlsruhe.

Date Submitted:
2023-09-26

Keywords:
Data management, HDF5, metadata, data lifecycle, Python, database

Data availability:

Software availability:
Software can be found here

Abstract. Sustainable data management is fundamental to efficient and successful scientific research. The FAIR principles (Findable, Accessible, Interoperable and Reusable) have been proven to be successful guidelines to enable comprehensible analysis, discovery and re-use. Although the topic has recently gained increasing awareness in both academia and industry, the engineering sciences in particular are lagging behind in managing the valuable asset of data. While large collaborations and research facilities have already implemented metadata strategies, smaller research groups and institutes are often missing a common strategy due to heterogeneous and rapidly changing environments as well as missing capacity or expertise. This paper presents an open-source package, called *h5RDMtoolbox*, written in Python helping to quickly implement and maintain FAIR research data management along the entire data lifecycle using HDF5 as the core file format. One of the key features of the toolbox is the flexible, high-level implementation of metadata standards, adaptable to the changing requirements of projects, collaborations and environments, such as experimental or computational setups. Implementation of existing schemas such as EngMeta or the cf-conventions are possible and intended use-cases. Other benefits of the toolbox include a simplified interface to the data and database solutions to query metadata stored in HDF5 files.

1 Introduction

Sustainable data management is fundamental in today's data-driven world for several reasons. The amount of acquired data storage capacity has long ceased to be the limiting factor, while the computing power has increased greatly [1]. However, it is the ability to share data rather than generate it that defines success [2]. Furthermore, interdisciplinary and international collaborations have become essential in scientific research, and the main means of communication is based on digital documents [3]. A bottleneck in data exploration and processing, and therefore the general re-usability, is often the lack of auxiliary data (metadata). As a consequence, much time is spent on recovering missing information. In some cases, this may require to re-conduct simulations and experiments. Effective data management practices hence hold the potential of saving time and money as well as increasing the value of data at the same time.

Introducing a new data management concept, however, is challenging: Different priorities, expectations and existing practices, as well as a lack of expertise or a clear understanding of the benefits, collide and may impede the efforts. Large and interdisciplinary collaborations depend on standards being used efficiently. Small research groups and PhD projects, however, are challenged by heterogeneous file formats, individual software solutions, personal preferences for storage and tools, and established structures [4]. They often do not have the time and resources to develop and implement an overarching data management approach that is fit for purpose and sustainable. The issue calls for flexible and practical metadata concepts that follow the basic requirements of the community, but take into account the needs of the specific project or research topic, too.

Although the implementation of a common management system is beneficial in the long term, both financially [5] and in terms of efficiency, it disrupts structures and requires time, resources and cultural change. In academia, high staff turnover is an additional barrier, making it difficult to establish sustainable solutions. The decay of value develops as projects progress, ultimately finish and contracts expire. Consequently, the value of data will diminish over time. This issue is discussed in more detail in [6], [7]. In addition, a value decay can also be observed with increasing distance from the source of the data. The further away and therefore less involved a potential data user is, the more information may be missing, either due to restricted access or limited personal connections. Ensuring that data is preserved and being interpretable at all times can be achieved by adhering to the so-called FAIR principles, which stand for Findable, Accessible, Interoperable and Reusable and were first introduced in 2016 by [8]. Since their publication, the principles have become the cornerstones of many scientific communities and help to establish a sustainable data management [9]. Structured, highly descriptive information about data, known as metadata, is an integral part of it. Metadata provides context about its creation, purpose, use, processing history and the meaning of datasets. Consequently, it enables data to be discoverable, interoperable and reusable.

In recent decades, there has been a growing awareness of the importance of sustainable data management. As a ”key component of open science” [3], [10] it has now become a requirement for funding and is also enforced by research journals [10]. Associations such as the National Research Data Infrastructure (NFDI) [11] in Germany or the European Open Science Cloud (EOSC) [12] are examples of organizational supporters for this trend. They provide trainings and concepts to support the FAIR, so-called, lifecycle of research data.

This work is a contribution to assist small collaborative groups or communities and doctoral researchers with achieving a FAIR research data lifecycle. The design concept and methodology are outlined in this paper and practical examples are given. Furthermore, the extensive online documentation [13] based on Jupyter Notebooks [14] provide more details and further examples for immediate usage.

2 The research data lifecycle

The different phases data run through, are described in the lifecycle of (research) data. There are several versions to lay out the lifecycle, each emphasizing different aspects depending on the research question or context. In this work, the cycle is broken down into five phases (cf. Figure 1). They serve to outline and contextualize the concept and features of the toolbox.

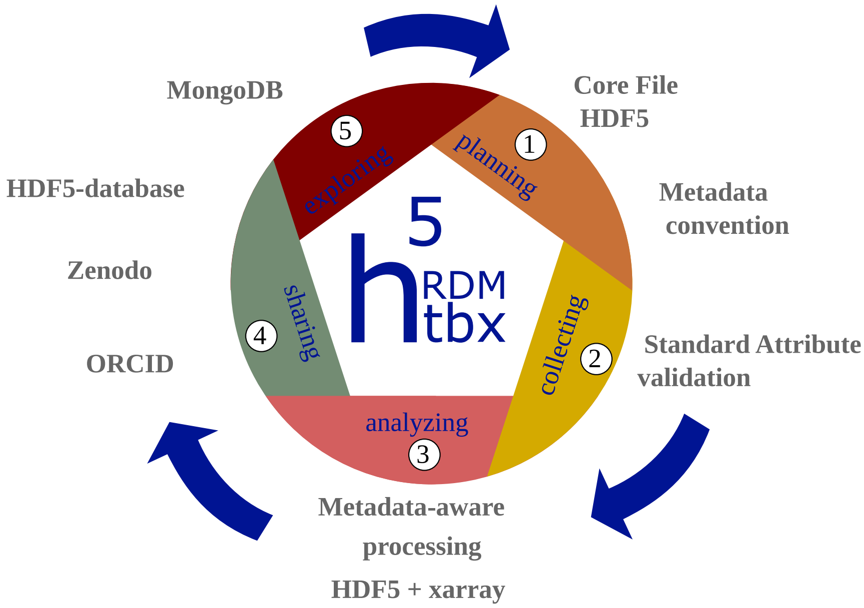

Figure 1: Illustration of the lifecycle of research data. Each phase is supported by the `h5RDMtoolbox`. It starts by selecting a file format and a metadata concept (1) and performing quality assurance measures during the selection and processing phase (2). Data is analyzed effectively for scientific output in the next step (3). After publication, the availability of the data should be ensured (4). (Meta)data quality finally is defined by its findability and consequently its re-usable (5) for additional analysis at later time. The respective tools and solutions provided by the toolbox and explained in this word are indicated by keywords around the lifecycle.

In the planning phase (1), a data management plan is written, stating concepts to handle data during and after the project. This includes the identification of relevant data to be collected, as well as a metadata strategy (e.g.selecting existing, adjust or define standards). The plan describes not only the concept and measures of handling data during, but also after the project (data archiving and publication). One important aspect is the agreement on one or multiple file formats. It has a significant impact on the realization of a FAIR data cycle as a whole. Many properties must be considered, such as public availability, software requirements, licenses, interoperability, handling, community acceptance and maturity of the file format. The data structure given by the collected data, e.g. the computational or experimental layout and properties used, will also influence the choice. The source output files can be very heterogeneous and may range from simple text-based files to structured or proprietary formats and need to be harmonized.

In the second phase, data is collected (2), e.g. experiments are carried out or simulations are conducted. Relevant metadata about the data is collected, including secondary information such as used instruments and software version. The collection phase further involves the mapping and conversion into the final file format in order to be compliant with a selected standard.

In the analysis phase (3), the data are studied and used for the first time for the purpose of scientific output. Data processing, such as statistical analysis or the calculation of derived variables and visualization, are the means to present the results of the research.

In the fourth phase, data is preserved and shared (4). Especially the metadata quality should be checked, and further information added, which is needed for the deposit in suitable data repositories. The datasets need to be assigned with persistent identifiers such as Digital Object Identifiers (DOIs) for example, licensing information and more. Depending on the kind of data and its confidentiality, data kept locally or is made publicly available with or without access restrictions.

The final phase (5) re-uses the data after publication. This is generally at a later point in time to generate new scientific insights through further analysis or to provide input for the planning of a new project.

3 Methodology and Concept of the toolbox

The primary objective of the toolbox is to provide comprehensive support to small collaborative groups or communities and doctoral researchers throughout the lifecycle of their data. A key aspect is the flexible implementation of metadata standards without imposing too much complexity but meeting the FAIR principles. The toolbox achieves this through three design principles:

1. **Relevant programming language:** The programming language has important influence on the usability and acceptance of this toolbox and data handling in general. Python is selected for this reason. It is the most popular and widely used language in the scientific community. The high relevance of Python in the field allows the toolbox to address as many users as possible.

2. **One core file format:** The Hierarchical Data Format (HDF5) [15] is chosen as the core and general purpose file format. It is suitable for most scientific and engineering data sources and allows metadata to be stored with the raw data, making it a self-explanatory data store. The file format is open-source, well-supported by the HDF5 Group [15] and has a proven track record in many disciplines. The choice of a single file format around which to build a management toolbox is therefore by no means a limitation. With user-friendliness and acceptance in mind, the toolbox requires a high-level interface to HDF5, extending the commonly used Python package *h5py* [16].

3. **Flexible Metadata Standardization:** The ability to store metadata alongside raw data requires its standardization (convention) to achieve discoverability. The definition of standard attributes, as introduced by the toolbox, is simple and flexible. The convention can be written into the file itself and provides feedback to the user about the correctness of the file with respect to the metadata description.

The chosen core file format HDF5 is widely used in science and management concepts exists. However, they are often very specialized for a specific application, e.g. [17]–[19] or [20], or only focus on partial aspects of the lifecycle, like database solutions [1], [21]. The presented toolbox therefore aims at practicality and a general approach in order to be applicable to a wide range of applications. Conversely, the management concepts cited should, in principle, be transferable to the approach presented here.

3.1 Planning

Aiming to simplify workflows and file handling, the concept presented is based on a single file format, namely the Hierarchical Data Format (HDF5). The choice of file format is not limiting in any way: HDF5 finds application in numerous scientific disciplines. It allows efficient storing of large multidimensional datasets together with metadata independent of the storage media, programming environment or operating system. The hierarchical structure of group and dataset objects (cf. Figure 2) resembles most engineering data, such as numerical experimental studies, for instance. Attributes (key-value pairs) are means to store metadata and can be assigned to each object. The HDF5 file format is therefore regarded as self-explanatory. It will fit most engineering data and proofs to be suitable to harmonize heterogeneous source data. An in-depth presentation of the file format can be found in [22]. A review of other file formats is beyond the scope of this work and literature should be referred to, for example [7], [23], [24].

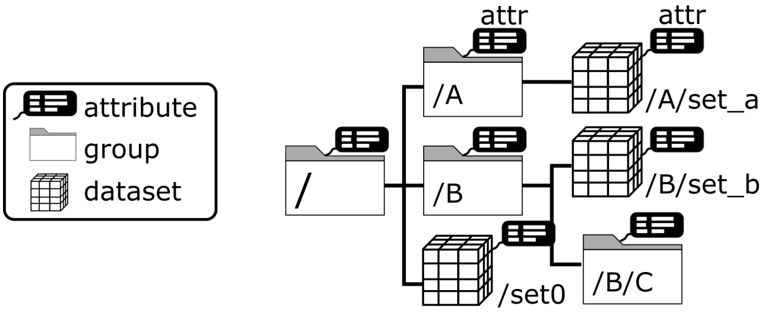

Figure 2: Illustration of the hierarchical structure of an HDF file. The internal file structure is organized like a file storage system, where folders are represented by the HDF group objects and files by HDF dataset objects. Both objects can be associated with attributes, which provide the metadata in order to make the objects interpretable.

Despite all the advantages of the file format, the organization of data management around HDF5 is left to the user [25]. This means that the choice of attribute names and values is not regulated by any standard. Findability, effective re-usability and automatic analysis, however, are dependent on standardization [26].

It is imperative to establish a standardized interface that includes a rigorous validation process to ensure accurate linkage of metadata to datasets. It is also important to give priority to ease-of-use and practicability, acknowledging that sustainable data management involves certain obligations, but does not overburden individual researchers. Achieving this balance is critical because researchers often prioritize their scientific output over long-term preservation, reusability and adherence to standards. This oversight can hinder the potential for future re-evaluation of data and the exploration of new research questions.

The package *h5RDMtoolbox* introduces so-called ”standard attributes” to regulate the obligatory or optional usage for datasets or groups. These objects can simply be defined in a YAML file. A collection of standard attributes is called ”convention” and is defined by a governance (stakeholders, a community, a project or a research group). It is then published to all collaborators. Currently, the Python package supports publication of conventions in Zenodo repositories, which will give them persistent identifiers, in this case a DOI. The layout of a convention YAML file is shown in Figure 3 and an example of such a convention can be found here: [27]. First, the name of the convention and contact information of its creator is given at the beginning. Then, the standard attributes are listed with their properties. In the example in Figure 3 ”comment”, ”contact_id” and ”data_type” are defined. Controlled via ”target_method” and ”default_value”, the attribute ”comment” is optional during dataset creation and the ”contact_id” is obligatory during the file creation. The default value for the ”data_type” is not given, which makes it an optional attribute by default. The values of all defined standard attributes will get checked by a validator. The creator of the convention may choose between built-in validators (e.g. ”$str”, ”$orcid”) or define own validators, like ”$dataSourceType” in the given example.

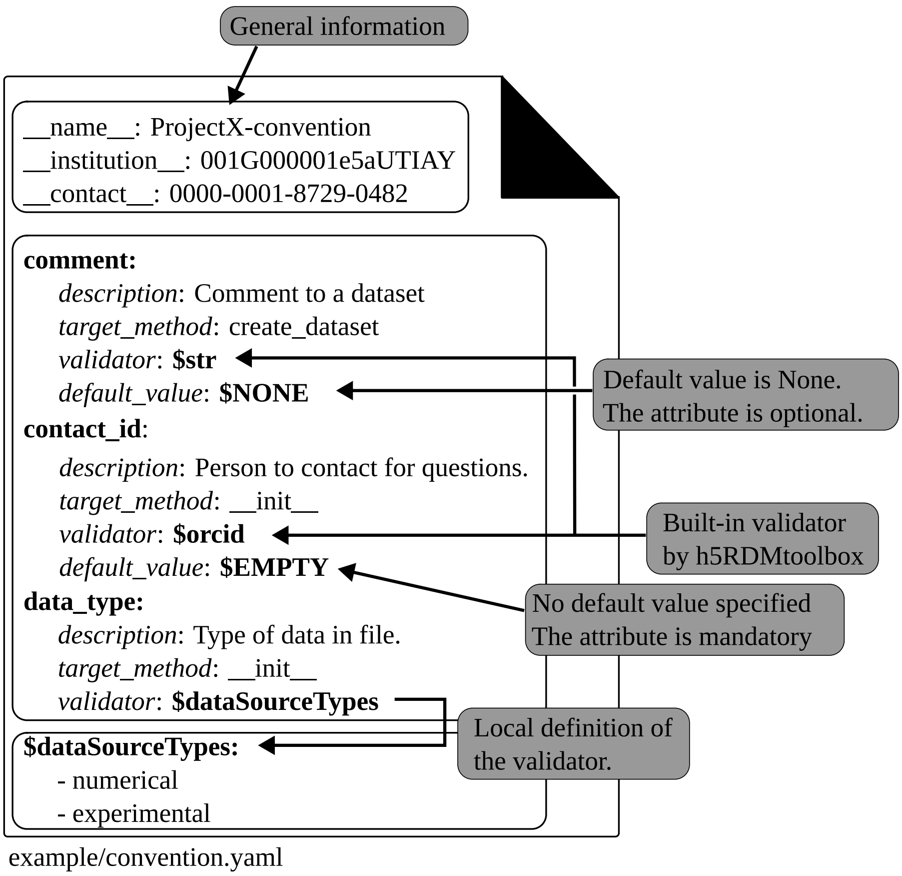

Figure 3: Example of a convention YAML file. First, general information is provided before the standard attributes and user-defined validators are documented.

The code snippet below shows, how a convention is read in for the first time and how it is enabled to be effected during following interaction with HDF5 files. Concrete and more complex examples can be found in the documentation [13].

```
import h5rdmtoolbox as h5tbx
# read the convention from file:
cv = h5tbx.conventions.from_yaml(my_convention.yaml)
# enable the convention
h5tbx.use(cv)
```

3.2 Collecting

In the collection phase, data is written to files. Depending on the use case and the origin of the data, files created by other software may need to be converted to HDF5. This step benefits greatly from the previously defined standard attributes and their validators, as shown in Figure 4. The convention is downloaded from the shared repository based on the DOI (get) and guides the user or software to generate the final HDF5 file (convert). The validators that are associated with the respective standard attributes will provide feedback to the conversion process by raising an error in the case of invalid metadata (reject). As use-cases and specific metadata may have been overlooked during design, missing or incorrect standard attributes should be reported to stakeholders at this stage. If the conversion is successful, the file can be used in subsequent lifecycle phases (analyzing, ...).

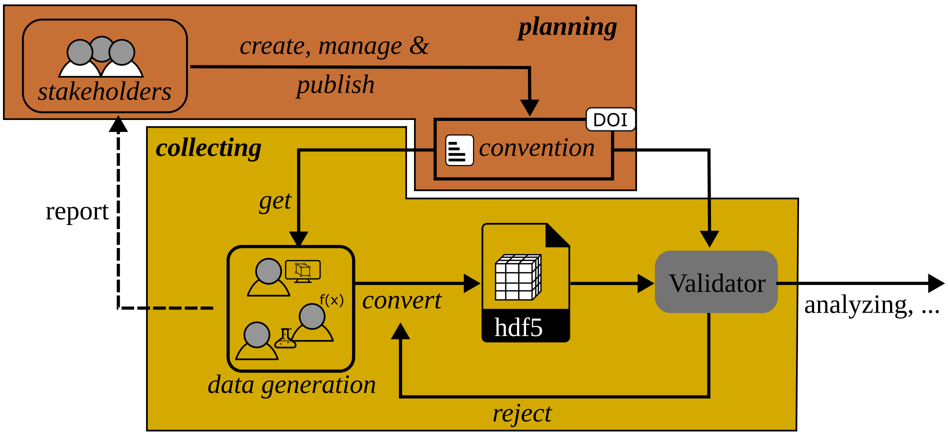

Figure 4: Workflow of collecting and converting the source data. The convention validates the created HDF5 files and serves as a feedback loop to the file creators or the software developers writing the conversion scripts. Only validated files can be further processed or published.

3.3 Analyzing

As pointed out previously, one reason, why HDF5 is selected is because it fits the multidimensionality of many scientific and engineering data. This means, that data is a function of other datasets, e.g. spatial and/or temporal coordinates. Fluid data is one such popular example. The data model of HDF5 allows associating the dimensions of a dataset array with other datasets in the file. Figure 5 illustrates this for the example of a three-dimensional dataset called ”data” with its dimensions ”x”, ”y” and ”time”.

In Python, a common interface is provided by the package *h5py*. It returns datasets as *numpy* arrays, which contain the raw values only [28]. However, this does not resemble the full information stored in the file, as the associated metadata is missing. The user must request the attribute data separately. The *h5RDMtoolbox* therefore uses the package *xarray* [29] instead. It allows associating data to the dimensions (called ”coordinates” in xarray syntax) of the array and supports usage of attributes, too. It hence resembles the HDF5 data model very closely. Using the package *xarray* to provide richly described data has three main benefits, which are also illustrated in Figure 5:

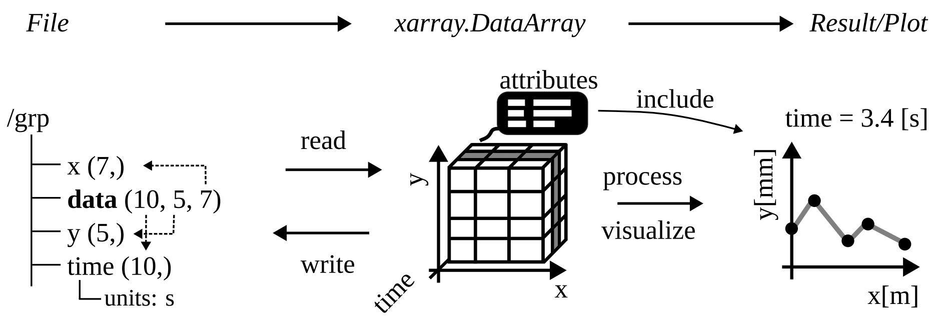

Figure 5: The h5RDMtoolbx makes use of the *xarray* features. Instead of *numpy* arrays, *xarray*.DataArray objects are returned, which allow carrying the dimension references and attributes and results in comprehensive data processing and visualization.

1. The *h5RDMtoolbox* manages the construction of the return type and therefore only one line of code is needed to obtain the *xarray* object.
2. The attributes become available automatically to the user when data is read from the file. This makes the array and its processing comprehensible and less error-prone.
3. Further processing can make use of the beneficial features of the *xarray* package, like data selection based on the dimensions/coordinate names and plotting features, which include metadata.

The code below demonstrates the workflow as illustrated in Figure 5. A subset of the dataset "data" is selected based on the coordinates. The return value is a *xarray.DataArray* on which the rolling mean is computed. The result is finally plotted on the screen. With only a few lines of code, the user obtains quick insight into the dataset while maintaining comprehensibility.

```
import h5rdmtoolbox as h5tbx
with h5tbx.File(filename) as h5:
    # select and read selected data and store in variable:
    d = h5['data'].sel(x=4.3, y=0.2, method='nearest')

# process (compute rolling mean over time with window size 3):
drm = d.rolling(time=3).mean()

# visualize the result:
drm.plot()
```

Although associating metadata with raw data is clearly beneficial, the file content can quickly get overwhelming. Third party software exist in the form of graphical user interfaces, however, this breaks the workflow. A better solution is to directly display the HDF5 file content. For this reason, *h5RDMtoolbox* allows printing the file content in a comprehensive way. This is especially useful when working with interactive Notebooks. The representation style is adopted from the one *xarray* uses. Figure 6 shows both, the interactive representations of *h5RDMtoolbox* and *xarray*.

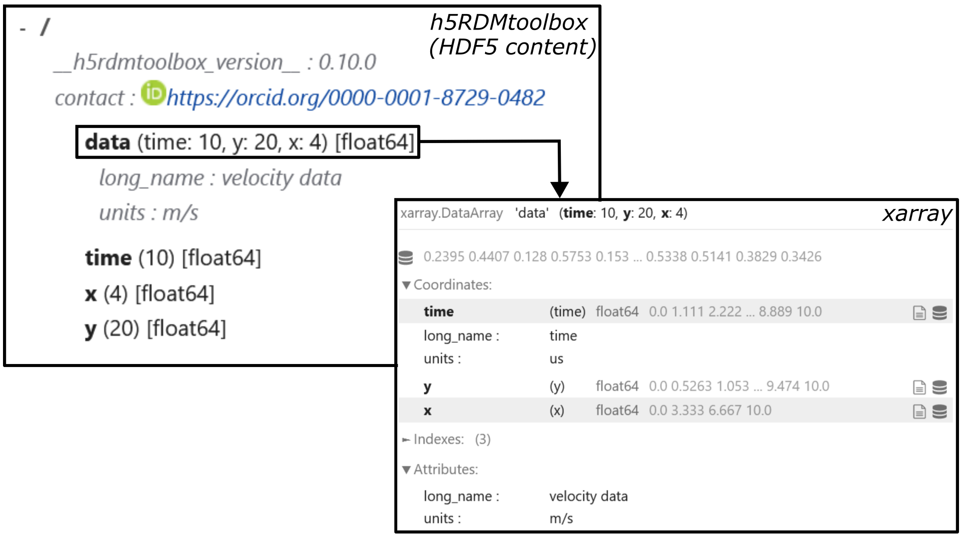

Figure 6: Screenshot of the HDF5 file content representation in a Jupyter Notebook through *h5RDMtoolbox* (upper left) and data array representation by package *xarray* (lower right). The content can be explored by navigating through the groups, datasets and attributes respectively.

3.4 Sharing and re-using

How data is shared depends on the scope and restrictions of the work. Most use-cases will, at least for some time, store data locally for internal use or later upload to a data repository. The toolbox supports two ways of sharing and re-using by means of databases:

1. Using HDF5 as a database inside a file system.
2. Mapping HDF5 to the NoSQL database MongoDB [30].

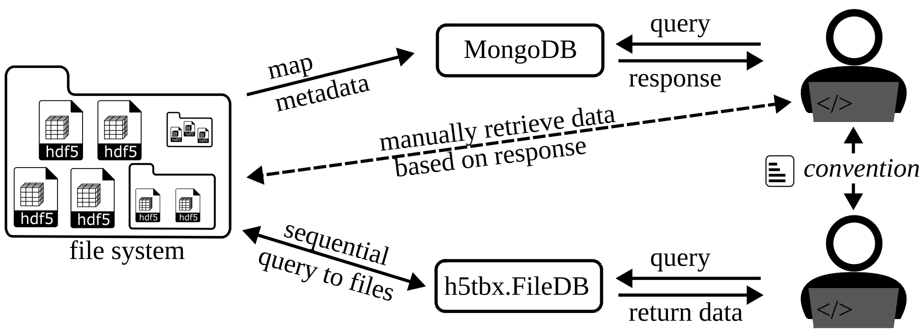

Figure 7: Workflow of the two provided database solutions: The metadata of the HDF5 files can be mapped to a MongoDB database and then filtered. The response includes the filename with which further processing can follow. The other option does not require a database infrastructure but filters the HDF5 files sequentially and returns the data directly.

Figure 7 shows the workflows for both options. The first approach treats an HDF5 file itself as a database and multiple files as multiple databases respectively. The consequences are twofold: Firstly, no third party database needs to be used and set up. Secondly, because files are opened and searched sequentially, the performance of finding data is not compatible with a dedicated system. However, if there are only a few files or the search is within a single file, the inefficiency is outweighing. In addition, this concept, as implemented in the toolbox, requires no prior operations on the data and only takes a minimum number of lines for the user.

The second approach extracts the file information and all metadata of each HDF5 object (datasets and groups) and writes it into the MongoDB. While this is the most time-consuming part, the query itself becomes highly efficient.

As Figure 7 indicates, the shared convention is also a valuable document at this stage in the lifecycle, because it tells the user which metadata has been regulated and is therefore appropriate to add in a query.

4 Documentation

The *h5RDMtoolbox* is versioned via a GitHub repository and can be installed using the python package installer (pip). The current version is v0.10.0 and extensive documentation automatically created and published [13]. The documentation website is generated based on Jupyter Notebooks. On the one hand, this results in practical documentation, showing code and explanations together. On the other hand, it allows users to reuse the code from the documentation for immediate application by simply copying the code snippets. As Jupyter Notebooks become more popular [14], [31], the option to download the full Notebooks will be another efficient option for most users who are new to the toolbox.

5 Conclusion and Outlook

The Python package *h5RDMtoolbox* has been introduced to assist researchers in handling HDF5 files in the FAIR way throughout the research data lifecycle. The tools include

- high-level standardization of metadata,
- user-friendly and metadata-aware HDF5 interface and
- HDF5 database solutions.

The target audience are mainly smaller projects, PhD research and collaborations. Flexibility, ease-of-use and the ability to quickly implement and adapt conventions according to the research question is a requirement of these users and strikes a balance between achieving high (meta)data quality and additional work. These features, together with the toolbox's design features such as the high-level interface using *xarray* and the database solutions around HDF5, help to increase the acceptance of data management. Furthermore, it shifts the invested workload from managing incomplete datasets to answering the scientific question.

The package and data management approach presented here differs from other existing solutions in its generality and ease of use. The metadata standardization is designed using simple YAML text files and therefore does not require source code editing or deep programming knowledge. It builds on existing and established solutions such as HDF5 for the file format or *xarray* for the dataset interface.

The next steps are to implement features that will further simplify data and metadata handling: This includes the implementation of a graphical user interface for the database concepts. Query fields could be generated directly from the convention.

Currently, the integration of existing schemas such as EngMeta [32] or the cf-conventions [26] requires manual translation into the YAML file format. Future developments may provide the corresponding reader functions.

Although automated unit tests are implemented, additional testing through practical application to various problems and scientific disciplines needs to follow. This will improve the code and allow it to be adapted to the needs of users. Current use cases investigate fluid problems, such as computational fluid dynamics simulations and particle image velocity measurements. Lessons learned from these areas will be incorporated into future publications, while further examples and guidelines will be continuously added to the online documentation [13].

6 Acknowledgements

The software was developed in-house without any external funding. The authors would like to thank all users, who have been testing the toolbox so far and provided helpful feedback.

7 Roles and contributions

Matthias Probst: Conceptualization, Writing, Software Development – original draft

Balazs Pritz: Project administration, Formal Analysis, Writing - review & editing

References

[1] Y. Wang, Y. Su, and G. Agrawal, “Supporting a Light-Weight Data Management Layer over HDF5,” in *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, IEEE, 2013, pp. 335–342, ISBN: 978-0-7695-4996-5. DOI: 10.1109/CCGrid.2013.9.

[2] J. Georgieva, V. Gancheva, and M. Goranova, “Scientific data formats,” in *AIC*, vol. 9, 2009, pp. 19–24.

[3] E. National Academies of Sciences, Medicine, *et al.*, “Open science by design: Realizing a vision for 21st century research,” 2018.

[4] F. De Carlo, D. Gürsoy, F. Marone, *et al.*, “Scientific data exchange: a schema for HDF5-based storage of raw and analyzed data,” *Journal of synchrotron radiation*, vol. 21, no. 6, pp. 1224–1230, 2014.

[5] European Commission and Directorate-General for Research and Innovation, *Cost-benefit analysis for FAIR research data : cost of not having FAIR research data*. Publications Office, 2019. DOI: doi/10.2777/02999.

[6] W. K. Michener, “Meta-information concepts for ecological data management,” *Ecological informatics*, vol. 1, no. 1, pp. 3–7, 2006.

[7] N. Preuss, G. Staudter, M. Weber, R. Anderl, and P. F. Pelz, "Methods and technologies for research-and metadata management in collaborative experimental research," in *Applied Mechanics and Materials*, Trans Tech Publ, vol. 885, 2018, pp. 170–183.

[8] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, *et al.*, "The FAIR Guiding Principles for scientific data management and stewardship," *Scientific data*, vol. 3, no. 1, pp. 1–9, 2016.

[9] A. Jacobsen, R. de Miranda Azevedo, N. Juty, *et al.*, "FAIR Principles: Interpretations and Implementation Considerations," *Data Intelligence*, vol. 2, no. 1-2, pp. 10–29, Jan. 2020, ISSN: 2641-435X. DOI: 10.1162/dint_r_00024.

[10] L. M. Schriml, M. Chuvochina, N. Davies, *et al.*, "COVID-19 pandemic reveals the peril of ignoring metadata standards," *Scientific Data*, vol. 7, no. 1, p. 188, 2020, ISSN: 2052-4463. DOI: 10.1038/s41597-020-0524-5. [Online]. Available: https://www.nature.com/articles/s41597-020-0524-5.

[11] N. Hartl, E. Wössner, and Y. Sure-Vetter, "Nationale Forschungsdateninfrastruktur (NFDI)," *Informatik Spektrum*, vol. 44, no. 5, pp. 370–373, 2021.

[12] A. V. d. Almeida, M. M. Borges, and L. Roque, "The European Open Science Cloud: A New Challenge for Europe," in *Proceedings of the 5th International Conference on Technological Ecosystems for Enhancing Multiculturality*, ser. TEEM 2017, Cádiz, Spain: Association for Computing Machinery, 2017, ISBN: 9781450353861. DOI: 10.1145/3144826.3145382.

[13] Probst, Matthias, *HDF5 Research Data Management Toolbox (Documentation)*. [Online]. Available: https://h5rdmtoolbox.readthedocs.io/en/latest/, (accessed: 26.09.2023).

[14] J. M. Perkel, "Why Jupyter is data scientists' computational notebook of choice," *Nature*, vol. 563, no. 7732, pp. 145–147, 2018.

[15] The HDF Group, *Hierarchical Data Format, version 5*. [Online]. Available: https://www.hdfgroup.org/HDF5/, (accessed: 25.09.2023).

[16] A. Collette, *Python and HDF5*. O'Reilly, 2013.

[17] Y. Meller and A. Liberzon, "Particle Data Management Software for 3DParticle Tracking Velocimetry and Related Applications–The Flowtracks Package," 2016.

[18] "HDF5eis: A storage and input/output solution for big multidimensional time series data from environmental sensors, author=White, Malcolm CA and Zhang, Zhendong and Bai, Tong and Qiu, Hongrui and Chang, Hilary and Nakata, Nori," *Geophysics*, vol. 88, no. 3, F29–F38, 2023.

[19] A. Ingargiola, T. Laurence, R. Boutelle, S. Weiss, and X. Michalet, "Photon-HDF5: an open file format for timestamp-based single-molecule fluorescence experiments," *Biophysical journal*, vol. 110, no. 1, pp. 26–33, 2016.

[20] P. Klosowski, M. Koennecke, J. Tischler, and R. Osborn, "NeXus: A common format for the exchange of neutron and synchroton data," *Physica B: Condensed Matter*, vol. 241, pp. 151–153, 1997.

[21] L. Gosink, J. Shalf, K. Stockinger, K. Wu, and W. Bethel, “HDF5-FastQuery: Accelerating complex queries on HDF datasets using fast bitmap indices,” in *18th International Conference on Scientific and Statistical Database Management (SSDBM’06)*, IEEE, 2006, pp. 149–158.

[22] S. Koranne and S. Koranne, “Hierarchical data format 5: HDF5,” *Handbook of open source tools*, pp. 191–200, 2011.

[23] R. E. McGrath, “XML and Scientific File Formats,” in *2003 Seattle Annual Meeting*, Citeseer, 2003.

[24] P. Greenfield, M. Droettboom, and E. Bray, “ASDF: A new data format for astronomy,” *Astronomy and computing*, vol. 12, pp. 240–251, 2015.

[25] S. Poirier, A. Buteau, M. Ounsy, *et al.*, “Common Data Model Access: A Unified Layer to Access Data From Data Analysis Point OF View,” *Icalepcs, Grenoble, October*, 2011.

[26] J. Gregory, “The CF metadata standard,” *CLIVAR Exchanges*, vol. 8, no. 4, p. 4, 2003.

[27] M. Probst, *Tutorial Standard Attribute Definition*, version v2.1, Sep. 2023. DOI: 10.5281/zenodo.8357399. [Online]. Available: https://doi.org/10.5281/zenodo.8357399.

[28] C. R. Harris, K. J. Millman, S. J. van der Walt, *et al.*, “Array programming with NumPy,” *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. DOI: 10.1038/s41586-020-2649-2.

[29] S. Hoyer and J. Hamman, “xarray: ND labeled arrays and datasets in Python,” *Journal of Open Research Software*, vol. 5, no. 1, 2017.

[30] K. Chodorow and M. Dirolf, *MongoDB - The Definitive Guide: Powerful and Scalable Data Storage*. O’Reilly, 2010, pp. I–XVII, 1–193, ISBN: 978-1-449-38156-1.

[31] T. Kluyver, B. Ragan-Kelley, F. Pérez, *et al.*, “Jupyter Notebooks-a publishing format for reproducible computational workflows.,” *Elpub*, vol. 2016, pp. 87–90, 2016.

[32] B. Schembera and D. Iglezakis, “EngMeta: metadata for computational engineering,” *International Journal of Metadata, Semantics and Ontologies*, vol. 14, no. 1, pp. 26–38, 2020.