Lightweight research data management with *dtool*: a use case

Johannes L. Hörmann
IMTEK, *liv*MatS
University of Freiburg

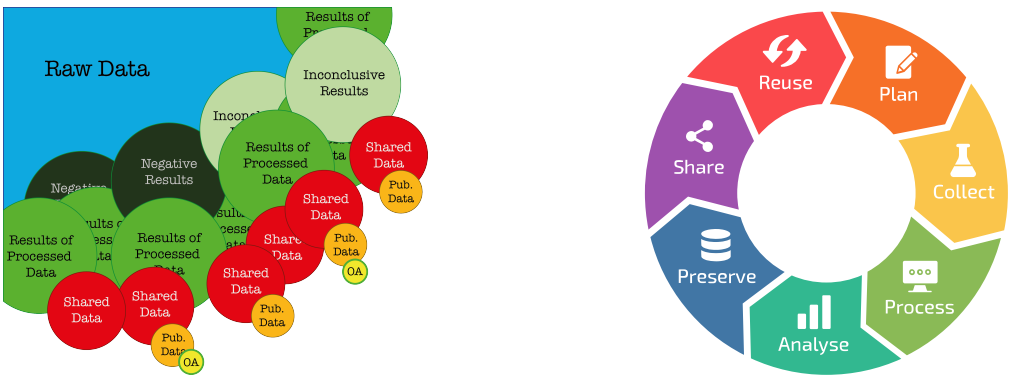
Lars Pastewka
IMTEK, *liv*MatS
University of Freiburg

Abstract

With the *dtool* data management framework, we adhere to the FAIR principles – findability, accessibility, interoperability, reusability – beginning at an early stage of the data lifecycle without introducing overwhelming administrative overhead. We show how the use of *dtool* has been implemented within IMTEK Simulation. In particular, we make data accessible, interoperable and reusable by packaging data and descriptive metadata in *dtool* datasets. The *dtool lookup server* makes data on a group-wide S3 object storage repository findable. The *dtool* ecosystem has proven applicable both for manual research data management as well as rapid generation of thousands of datasets in automized workflows.

1 Introduction

The ever-increasing amount of data underlying today's scientific publications needs to be **f**indable, **a**ccessible, **i**nteroperable and **r**eusable for ensuring efficient collaboration as well as reproducibility and quality of scientific results. Holding up these FAIR principles [3] means increased documentation work for researchers and curation work for administrative staff. Facilitating research data management under FAIR principles is addressed on different levels by initiatives like *Dublin Core* [4] and *DataCite* [5] promoting metadata conventions, by file layout conventions like *bagIt* [6] and by software ecosystems that may comprise both such conventions and their implementation, such as *RO-Crate* [7], *Annotated Research Contexts* [8, 9], *DataLad* [10], and *dtool* [11]. Stringent metadata standards and data management software may help to guarantee FAIR research data, but added value for the scientific community as a whole may come at the cost of administrative overhead and hindering constraints for the researcher. The right balance is inherently case-specific, there is no one-size-fits-all solution. On a use case from the computational material science community, we show how FAIR principles have been implemented with minimal overhead and few constraints with the help of *dtool*, a lightweight research data management ecosystem.



(a) Data taxonomy: From raw to published data [1].

Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0),
https://creativecommons.org/licenses/by-nc-sa/4.0/deed.en

(b) Data lifecycle [2].

Creative Commons Attribution 4.0 International (CC BY 4.0),
https://creativecommons.org/licenses/by/4.0/

Figure 1: Conceptual schemes.

Licensed under the Creative Commons License CC BY 4.0:
https://creativecommons.org/licenses/by/4.0/legalcode

Proceedings of the 7th bwHPC Symposium (ISBN 978-3-948303-29-7)
Article No. 6; pages 29–35

```
2021-11-08-measurement-and-evaluation : the dataset's name
|--.dtool : administrative and structural metadata
|--README.yml : descriptive metadata
|--data : items, i.e...
   |-- raw_data.bin : some binary raw data
   |-- analysis.xlsx : some derived analysis in Excel sheet
   |-- README.md : some more textual documentation
```

Figure 2: Sample layout of a dataset on a hierarchical file system. Note that the packaged data reside as items nested below `data` while a top-level `README.yml` provides YAML-formatted descriptive metadata. Olsson and Hartley [11] provide an in-detail description of the layout, including the administrative and structural metadata nested below subfolder `.dtool`.

```yaml
project: Project name
description: Short description
owners:
  - name: {DTOOL_USER_FULL_NAME}
    email: {DTOOL_USER_EMAIL}
    username: {username}
    orcid: Please obtain an ORCID at orcid.org
funders:
  - organization: Please add funding organization
    program: Please add the specific program within which funding occurred
    code: Please add funding code of that organization
creation_date: '{date}'
expiration_date: '{date}'
derived_from:
  - uuid: UUID of the primary data or the previous simulation step
```

Figure 3: Sample `README.yml` template applied in this use case. The `derived_from` field with a `uuid` key-value list is optional.

dtool is lean and decentralized. It is implemented in Python and its core functionality is distributed across several modular packages of minimal scope, facilitating customization and extension. Comparable solutions often target community-wide data sharing and publication at outer layers of the data taxonomy bubbles shown in Fig. 1a, or at the late stages of the data lifecycle shown in Fig. 1b. For example, sophisticated metadata schemes as promoted by DataCite [5] aim at standardizing the description of published, citable data. *dtool*, on the contrary, does not necessarily target the late stages of the data lifecycle only. Instead, its lightweight design empowers the user to treat their data as FAIR as they see fit over the whole data lifecycle without much overhead. Hence, this use case specifically covers the voluminous inner data taxonomy bubbles in Fig. 1a.

The design of *dtool* has been described exhaustively by its developers Olsson and Hartley [11]. Only a few concepts are essential and emphasized here. Within the *dtool* ecosystem, the core discretization unit is a *dataset*. The dataset bundles data and metadata. Any such dataset is assigned a universally unique identifier (UUID) [12] at creation that makes it unambiguously identifiable. After filling in data and descriptive metadata, a dataset is made immutable via *freezing*. This precludes any incremental versioning within one dataset. Instead, a version increment is achieved simply by deriving a new dataset with modified content from another frozen entity. Fig. 2 outlines the simple structure of a dataset.

2 Implementation components

Documentation requirements

dtool itself does not impose any metadata ontology constraints. A dataset's top-level `README.yml` may contain any kind of YAML-formatted descriptive metadata. In the context of this use case, we only require a minimum of administrative and bibliographic metadata. These requirements on structured metadata are imposed by providing a YAML-formatted template `README.yml` shown in Fig. 3. The user configures *dtool* to refer to such a template when creating a new dataset to fill the required metadata interactively.

Storage brokers

The *dtool* API abstraction layer renders the manipulation of datasets independent of their actual representation on the underlying storage system. In a highly volatile environment of decentralized infrastructure as academic researches find themselves in, this is advantageous. Storage brokers map dataset manipulation tasks to low-level operations. Currently, storage brokers exist for the local file systems, for iRODS [13], for Azure storage [14], for S3 object storage [15], and for Windows (smb) network shares [16]. In this use case, we operate on S3 object storage provided by *bwSFS* [17].

Lookup framework

The decentralized framework described above helps to increase the *accessibility*, *interoperability* and *reusability* of our data. The *dtool lookup server* [18] makes data *findable*, satisfying the first of the FAIR principles as well. In our case, the lookup server has been configured to monitor a *bwSFS* S3 repository of several thousand datasets. Group members are encouraged to deposit all datasets on the group-wide repository. Next to (re-)indexing the repository at regular intervals, the *dtool lookup server notification plugin* [19] listens to notifications from the S3 storage platform for the immediate ingestion of newly deposited datasets. A reference container composition [20] serves both testing purposes and as the foundation of our production instance. Continuous integration workflows assure the composition's reliability.

Querying and aggregation. On the client side, the *dtool lookup client* [21] and *dtool lookup API* [22] facilitate simple listing and querying of registered datasets and their associated metadata by UUID or full text search from the *dtool lookup server*. Submitting sophisticated queries and aggregation pipelines is possible with the server-side *dtool lookup server direct mongo plugin* [23]. The lookup server uses the source-available MongoDB [24] to ingest administrative metadata and the full `README.yml`. The ingestion of a dataset maps all JSON- and YAML-formatted metadata onto the native MongoDB-internal JSON-like data format. The *dtool lookup server direct mongo plugin* [23] allows to submit direct mongo language queries and aggregation pipelines to the lookup server's underlying database. Similarly to its native data format, MongoDB language queries and aggregations are formulated as JSON-like documents as well. This NoSQL peculiarity makes formulating queries from nested lists and dictionaries easy to learn for the Python-affine researcher. The appendix illustrates a sophisticated aggregation pipeline submitted via *dtool lookup API* at the end of this document.

Dependency graph. In our group, users are encouraged to record dependencies between datasets by filling out the `README.yml` template's `derived_from` list (Fig. 3). The *dtool lookup server dependency graph plugin* [25] allows querying the complete dependency graph of a dataset. By default, UUIDs of parent datasets are expected within the `README.yml` below `derived_from`. This, however, is configurable arbitrarily.

GUI.The *dtool lookup GUI* [26] facilitates browsing and searching datasets registered at the lookup server. Search queries can be plain text search or sophisticated MongoDB language. Administrative metadata and content of the `README.yml`, the manifest and a visual representation of the dependency graph are directly accessible. Fig. 4 illustrates the GUI search and metadata overview, Fig. 5 shows the dependency graph of an automized parametric study.

Workflow management framework integration

We provide custom tasks [27] to embed *dtool* dataset creation, filling, freezing and transfer as well as lookup server queries into the *FireWorks* workflow management framework [28]. *FireWorks* uses JSON-like documents maintained by MongoDB for describing workflows and attached metadata as well, enabling seamless transfer from *FireWorks* workflow metadata into *dtool* datasets and vice versa. Thereby, conservation of data provenance in rapid throughput studies becomes independent of the actual *FireWorks* instance.

3 Conclusion

With the modular components of *dtool*, we are able to implement FAIR principles flexibly and efficiently. While many other solutions aim at data publication at the late stages of the data lifecycle, we offer group members the possibility and encourage them to adhere to those principles over the whole data lifecycle, beginning at

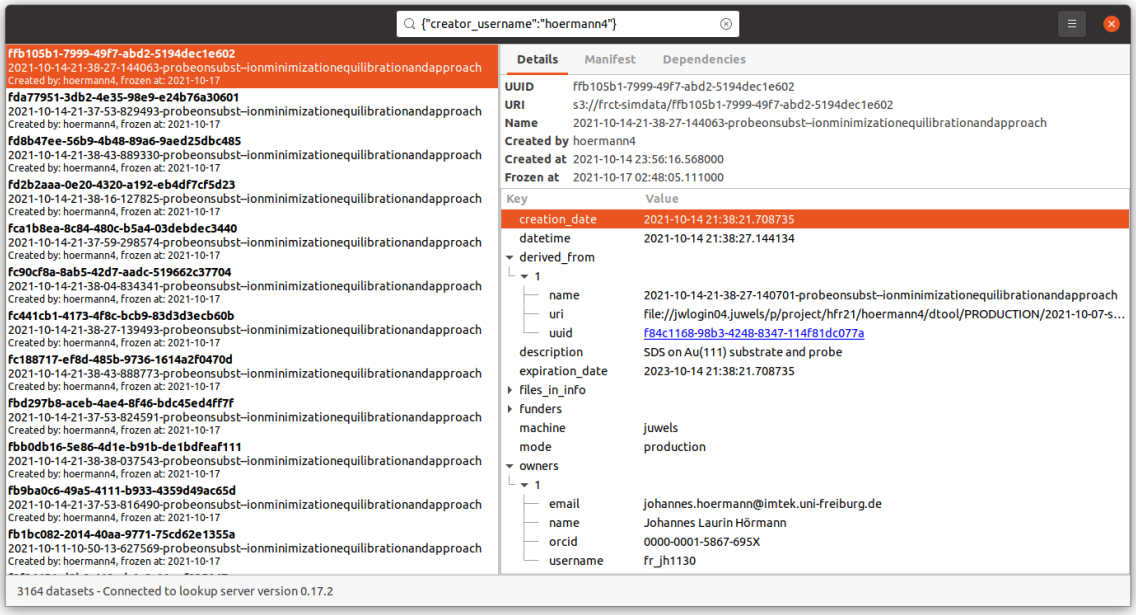


Figure 4: Datasets of specific creator are filtered by MongoDB language query in the *dtool lookup GUI*. Administrative metadata and expandable content of `README.yml` for the selected dataset are visible.

data creation. Both manual data management with textual documentation as well as rapid automized data processing with complex machine-readable metadata schemes are possible. Eventually, the decision on how far to go in terms of FAIRness lies with each user and their choice of documentation standards. The approach has proven scalable – by now, our group-central repository has grown beyond 10,000 datasets of over 200 TB total volume without noticeable loss of search performance. The pythonic and modular nature of *dtool* allows quick development of user extensions whenever the need to fill a functionality gap arises. The custom server-side plugins as well as the client-side extensions introduced in this article underline this. The learning curve for Python-affine users is low. Making datasets on existing storage infrastructure searchable by instantiating a *dtool lookup server* only requires moderate effort.

4 Acknowledgements

We thank Tjelvar Olsson, the lead developer behind *dtool*, for many fruitful discussions.

The group-wide dataset repository resides on object storage provided by *bwSFS*, Baden-Württemberg's *Storage for Science*, funded by the Deutsche Forschungsgemeinschaft (DFG) under project number 405998531.

References

[1] R. Ganguly. "Data from research process," Phaidra, Universität Wien. (2015), [Online]. Available: `https://phaidra.univie.ac.at/o:387241` (visited on 10/24/2021).

[2] RDMKit. "Data life cycle diagram basic," ELIXIR Europe. (2021), [Online]. Available: `https://rdmkit.elixir-europe.org/images/data_life_cycle.svg` (visited on 10/24/2021).

[3] 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, p. 160 018, Mar. 2016. DOI: 10.1038/sdata.2016.18. [Online]. Available: `https://www.nature.com/articles/sdata201618`.

[4] DCMI Usage Board, *DCMI metadata terms*, Dublin Core Metadata Initiative, 2020. [Online]. Available: `http://dublincore.org/specifications/dublin-core/dcmi-terms/2020-01-20/`.

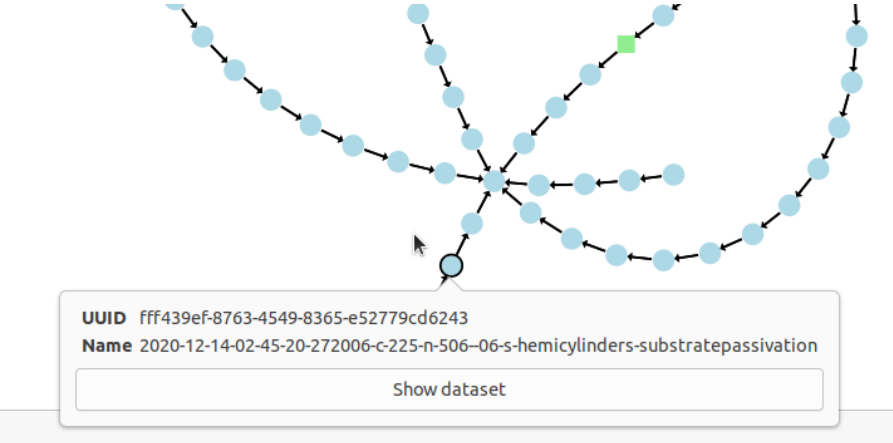


Figure 5: The dependency graph of datasets generated in an automized parametric study. The currently selected dataset appears as a green box. When hovering the cursor over other datasets in the graph, a tooltip shows their UUID and name and offers to jump there.

[5] DataCite Metadata Working Group, "Datacite metadata schema documentation for the publication and citation of research data and other research outputs," Tech. Rep., version 4.4, 2021. DOI: 10.14454/3w3z-sa82. [Online]. Available: https://doi.org/10.14454/3w3z-sa82.

[6] J. Kunze, J. Littman, E. Madden, *et al.*, "The bagit file packaging format (v1. 0)," Internet Engineering Task Force, RFC 8493, Oct. 2018. DOI: 10.17487/RFC8493. [Online]. Available: https://www.rfc-editor.org/info/rfc8493.

[7] S. Soiland-Reyes, P. Sefton, M. Crosas, *et al.*, "Packaging research artefacts with ro-crate," unpublished, Aug. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5146228.

[8] D. Von Suchodoletz, T. Mühlhaus, D. Brillhaus, *et al.*, "Data stewards as ambassadors between the NFDI and the community," 2022, Publisher: heiBOOKS. DOI: 10.11588/HEIBOOKS.979.C13750. [Online]. Available: https://books.ub.uni-heidelberg.de/index.php/heibooks/catalog/book/979/c13750 (visited on 06/24/2022).

[9] C. Garth, J. Lukasczyk, T. Mühlhaus, *et al.*, "Immutable yet evolving: ARCs for permanent sharing in the research data-time continuum," 2022, Publisher: heiBOOKS. DOI: 10.11588/HEIBOOKS.979.C13751. [Online]. Available: https://books.ub.uni-heidelberg.de/index.php/heibooks/catalog/book/979/c13751 (visited on 06/24/2022).

[10] Y. O. Halchenko, K. Meyer, B. Poldrack, *et al.*, "DataLad: Distributed system for joint management of code, data, and their relationship," en, *Journal of Open Source Software*, vol. 6, no. 63, p. 3262, Jul. 2021. DOI: 10.21105/joss.03262. [Online]. Available: https://joss.theoj.org/papers/10.21105/joss.03262 (visited on 10/19/2021).

[11] T. S. G. Olsson and M. Hartley, "Lightweight data management with dtool," *PeerJ*, vol. 7, e6562, 2019. DOI: 10.7717/peerj.6562. [Online]. Available: https://peerj.com/articles/6562.

[12] P. Leach, M. Mealling, and R. Salz, "A universally unique identifier (uuid) urn namespace," RFC 4122, Jul. 2005. DOI: 10.17487/RFC4122. [Online]. Available: https://www.rfc-editor.org/info/rfc4122.

[13] T. S. G. Olsson and M. Hartley, *dtool-irods*, version 0.10.1, Feb. 2021. [Online]. Available: https://github.com/jic-dtool/dtool-irods.

[14] T. S. G. Olsson and M. Hartley, *dtool-azure*, version 0.7.1, Jun. 2020. [Online]. Available: https://github.com/jic-dtool/dtool-azure.

[15] T. S. G. Olsson, M. Hartley, A. Carrgilson, *et al.*, *dtool-s3*, version 0.12.0, Aug. 2021. [Online]. Available: https://github.com/jic-dtool/dtool-S3.

[16] L. Pastewka and J. L. Hörmann, *dtool-smb*, version 0.1.0, May 2020. [Online]. Available: https://github.com/livMatS/dtool-smb.

[17] Research Data Management Group, "Bwsfs – storage for science," Blog entry, 2021. [Online]. Available: https://rdmg.uni-freiburg.de/posts/2021/06/bwsfs-storage-for-science/ (visited on 10/24/2021).

[18] T. S. G. Olsson, J. L. Hörmann, and M. Hartley, *dtool-lookup-server*, version 0.17.2, May 2021. [Online]. Available: https://github.com/jic-dtool/dtool-lookup-server.

[19] L. Pastewka and J. L. Hörmann, *dtool-lookup-server-direct-mongo-plugin*, version 0.1.1, May 2021. [Online]. Available: https://github.com/livMatS/dtool-lookup-server-notification-plugin.

[20] J. L. Hörmann and L. Pastewka, *dtool-lookup-server-container-composition*, version 0.3.0, Oct. 2021. [Online]. Available: https://github.com/jotelha/dtool-lookup-server-container-composition.

[21] T. S. G. Olsson and M. Hartley, *dtool-lookup-client*, version 0.1.0, Apr. 2019. [Online]. Available: https://github.com/jic-dtool/dtool-lookup-client.

[22] J. L. Hörmann, L. Pastewka, and A. Sanner, *dtool-lookup-api*, version 0.1.1, Nov. 2020. [Online]. Available: https://github.com/livMatS/dtool-lookup-api.

[23] J. L. Hörmann, *dtool-lookup-server-direct-mongo-plugin*, version 0.1.4, May 2021. [Online]. Available: https://github.com/livMatS/dtool-lookup-server-direct-mongo-plugin.

[24] MongoDB Inc., *MongoDB*, version r4.4.0, Jul. 2020. [Online]. Available: https://github.com/mongodb/mongo.

[25] J. L. Hörmann and L. Pastewka, *dtool-lookup-server-dependency-graph-plugin*, version 0.1.5, May 2021. [Online]. Available: https://github.com/livMatS/dtool-lookup-server-dependency-graph-plugin.

[26] J. L. Hörmann, L. Pastewka, A. Sanner, *et al.*, *dtool-lookup-gui*, version 0.6.1, Jun. 2021. [Online]. Available: https://github.com/livMatS/dtool-lookup-gui.

[27] J. L. Hörmann, *imteksimfw*, version 0.1.7, Jan. 2021. [Online]. Available: https://github.com/IMTEK-Simulation/imteksimfw.

[28] A. Jain, S. P. Ong, W. Chen, *et al.*, "Fireworks: A dynamic workflow system designed for high-throughput applications," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 17, pp. 5037–5059, 2015. DOI: 10.1002/cpe.3505. [Online]. Available: http://dx.doi.org/10.1002/cpe.3505.

Appendix: Code listing

The following aggregation pipeline groups all datasets owned by user `hoermann` by a custom `project` identifier found within their `README.yml`, counts the datasets for each such project and sorts these records by the earliest date a dataset under the specific project identifier has been frozen. A *pandas* dataframe is used to visually inspect the results within a *Jupyter notebook*.

In[1]:

```
import pymongo
import pandas as pd
import dtool_lookup_api as dl
aggregation_pipeline = [
    {
        '$match': {'creator_username': 'hoermann'}
    }, {
        '$group': {
            '_id': { 'project': '$readme.project' },
            'object_count': {'$sum': 1}, # count matching data sets
            'earliest':  {'$min': '$frozen_at' },
            'latest':  {'$max': '$frozen_at' },
        },
    }, {  # pull 'project' field up in hierarchy
        '$addFields': {'project': '$_id.project'},
```

```
    }, {  # drop nested '_id.project'
        '$project': {'_id': False},
    }, {  # sort by earliest date, descending
        '$sort': {'earliest': pymongo.DESCENDING}
    }
]
res = await dl.aggregate(aggregation_pipeline)
df = pd.DataFrame(res)
df[['earliest','object_count','project']].iloc[:3] # only show first three records
```

Out[1]:

	earliest	object_count	project
0	Wed, 13 Oct 2021 12:13:46 GMT	345	2021-10-07-sds-on-au-111-probe-and-substrate-merge-and-approach
1	Wed, 13 Oct 2021 12:12:48 GMT	217	2021-10-06-sds-on-au-111-substrate-passivation
2	Tue, 16 Mar 2021 04:44:20 GMT	13	2021-03-01-sds-on-au-111-cluster-passivation