Realising Data-Centric Scientific Workflows with Provenance-Capturing on Data Lakes (Post-print)

Authors: Hendrik, Nolte, Philipp, Wieder, Hendrik, Nolte

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

Abstract

Since their introduction by James Dixon in 2010, data lakes get more and more attention, driven by the promise of high reusability of the stored data due to the schema-on-read semantics. Building on this idea, several additional requirements were discussed in literature to improve the general usability of the concept, like a central metadata catalog including all provenance information, an overarching data governance, or the integration with (high-performance) processing capabilities. Although the necessity for a logical and a physical organisation of data lakes in order to meet those requirements is widely recognized, no concrete guidelines are yet provided. The most common architecture implementing this conceptual organisation is the zone architecture, where data is assigned to a certain zone depending on the degree of processing. This paper discusses how FAIR Digital Objects can be used in a novel approach to organize a data lake based on data types instead of zones, how they can be used to abstract the physical implementation, and how they empower generic and portable processing capabilities based on a provenance-based approach.

Full Text

Preamble

RESEARCH PAPER
Realising Data-Centric Scientific Workflows with Provenance-Capturing on Data Lakes
Hendrik Nolte† & Philipp Wieder
Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen
37077 Göttingen, Germany

Keywords: Data lake; Provenance; Workflows; FAIR Digital Objects; CWFR

Citation: Nolte, H., Wieder, P.: Realising data-centric scientific workflows with provenance-capturing on data lakes. Data Intelligence 4(2), 426-438 (2022). doi: 10.1162/dint_a_{00141}
Received: September 18, 2021; **Revised:** November 26, 2021; **Accepted:** February 5, 2022

ABSTRACT

Since their introduction by James Dixon in 2010, data lakes have garnered increasing attention, driven by the promise of high data reusability enabled by schema-on-read semantics. Building on this foundation, literature has discussed several additional requirements to improve the general usability of the concept, including a central metadata catalog containing all provenance information, overarching data governance, and integration with (high-performance) processing capabilities. Although the necessity of logical and physical organization in data lakes to meet these requirements is widely recognized, no concrete guidelines have yet been provided. The most common architecture implementing this conceptual organization is the zone architecture, where data is assigned to specific zones based on its degree of processing. This paper discusses how FAIR Digital Objects can be used in a novel approach to organize a data lake based on data types rather than zones, how they can abstract the physical implementation, and how they enable generic and portable processing capabilities through a provenance-based approach.

1. INTRODUCTION

Data pipelines are widely used to (i) collect data from heterogeneous sources, (ii) perform a series of transformations, and (iii) ingest the transformed data into a destination system for analytics. Such pipelines are often referred to as ETL (extract, transform, load) processes, commonly used to ingest clean, transformed data into data warehouses [1]. Despite being considered state-of-the-art, data pipelines suffer from well-known drawbacks [2]. One major problem is information loss during the ETL process, which can occur when relevant attributes are aggregated to make the format suitable for a data warehouse. Consequently, only a predetermined subset of attributes remains available for subsequent analysis. The schema-on-write semantics of this approach thus limits the reuse of data stored in traditional data management systems for analytics beyond the original scope.

To prevent this information loss, Dixon introduced the concept of data lakes in 2010 [3]. Unlike data warehouses, data lakes are designed as central repositories for all datasets from all sources in their raw format [4]. Since no transformation is needed for data ingestion and no assumptions are made about subsequent

analysis, a schema-on-read approach ensures high flexibility and reusability.

Although no commonly accepted definition of a data lake exists in the literature [5], Dixon' s original outline requires a scalable storage system for heterogeneous data where scientists can explore and analyze datasets. These requirements align with the need for low-cost technologies, initially leading to a strong association between data lake implementations and Apache Hadoop [6], later superseded by proprietary cloud solutions based on Azure or AWS [4]. These cloud solutions introduced the advantage of separating storage and compute resources into the data lake concept.

While data lakes implement schema-on-read semantics, some modeling is mandatory to ensure proper data integration, comprehensibility, and quality [7]. Such data models, created by extracting descriptive metadata from ingested raw data, are stored in a central data catalog. This catalog does not enforce a fixed schema and therefore allows frequent changes [8]. Although many studies focus primarily on this data catalog and gradual improvement of provisioned metadata [9, 10], such as semantic information about datasets [11], provenance data is not yet extensively covered by current approaches. However, since raw data in a data lake is likely subjected to many consecutive transformations, resulting in several artifacts that are ingested back into the data lake, maintaining concise provenance information is both challenging and crucial for retaining manageability [12]. Processed data simply stored back into the data lake becomes difficult to find afterward and may be impossible to comprehend and reproduce, potentially rendering it useless.

This paper discusses (i) how the strong association between a single data entity and its metadata can be expressed within a FAIR Digital Object [13] in the data lake context, (ii) how typed digital objects can conceptually organize a data lake architecture, (iii) how packaging generic analysis tasks within FAIR Digital Objects can help monitor fine-grained provenance information, and (iv) how exploiting these concepts within Canonical Workflow Frameworks for Research (CWFR) can increase cross-disciplinary collaboration.

2. RELATED WORK

Various solutions tailored for specific purposes have been proposed for provenance capturing in data lakes. Goods [14] analyzes log files post-hoc to determine which jobs created datasets based on which inputs, an approach requiring applications to write suitable log files. Similarly, Komadu was integrated into a data lake [12] to support messaging of provenance information via RabbitMQ, also relying on explicit application support. DataHub [15] was equipped with ProvDB [16] for provenance auditing. ProvDB analyzes shell scripts, user annotations, post-hoc logs from frameworks like Caffe, and table-based input files for SQL-based provenance capture. However, this approach is error-prone, and monitoring system calls introduces additional runtime overhead.

A common data lake architecture is the so-called zone architecture [5, 17]. The general idea is to divide the data lake into different zones and store data according to the amount of processing it has undergone. Raw data is stored in a raw zone, while (pre-)processed data resides in dedicated zones. Although recent progress has been made in defining a standardized zone reference model [18], this architecture lacks a homogeneous, standardized interface for integrating processes and users, a central data catalog to maximize reusability, and reliable provenance auditing by design.

It is now common practice that scientific publications are complemented with an organized aggregation of all related digitized entities to promote comprehensibility and further reusability of data and methods. Research Objects [19] have been proposed as structured aggregations that semantically link their entities, defining standards regarding repetition, traceability, and essential metadata. Tools like Sciunits [20] have been developed to automate building research objects from workflow enactments, using application virtualization based on system call monitoring to build ready-to-use containers including all software, dependencies, and used data.

All these developments independently tackle different problems in collaborative data analytics. However, their full potential can only be realized when these individual steps are integrated into a cohesive solution.

---

3. IMPLEMENTATION

This section presents a data lake implementation tailored for data- and compute-intensive research areas, with special focus on traceability and reproducibility of processing steps such as data ingestion and analysis operations. Since the presented implementation is a generic framework aiming to support various use cases, we separate the implementation of these processing steps from the core data lake, making it configurable and extensible for individual use cases. Additionally, the data lake includes central role-based user access management to support scalability and large teams.

3.1 Architecture Overview

The core of our data lake implementation is a central web application that provides the REST API and implements orchestration functions to enforce a consistent state and essential minimum data quality. This is where the mapping between logical and physical organization occurs.

A schematic view is provided in Figure 1 [Figure 1: see original paper]. The web application serves as the single point of contact for data sources to ingest data and for users to interact with the data lake. All services needed to realize the complete data lake are accessed exclusively through this web application, not directly by users. This design allows back-end services to be easily swapped or

added without requiring users to modify existing workflows and pipelines. One exception is direct user interaction with version control tools like GitLab, which is tightly integrated to support widely used development workflows.

Figure 1. Schematic view of the implemented data lake architecture. The organization of underlying services is abstracted by the unifying data lake layer where overarching governance is enforced.

Additional advantages of this implementation include scalability, consistency, and maintainability, as well as fine-grained control and high-level abstractions of underlying processes that empower non-experts to use the data lake.

3.2 Ingestion

It is widely accepted that a data lake, as an institution' s central data repository, should accept any data type and format from any source [4]. Therefore, the ingestion layer must be highly flexible, with implications for two aspects. First, data transmission should use common protocols like HTTP. It can be advantageous if the data lake actively pulls data from other systems to stay up-to-date, reducing maintenance overhead by managing many heterogeneous data pipelines from a single system and interface. Second, storage and metadata systems must cope with or easily adapt to any input data to automatically extract descriptive metadata for updating the central data catalog.

To upload a new file containing raw data, users first configure a new metadata type consisting of key-value pairs with their respective mappings. Then, another API is used to upload containers holding a metadata extraction tool (e.g., a simple Python script). The metadata extracted from ingested raw data can be categorized into pre-visualization, summary, and semantic metadata [10], primarily containing informative and descriptive information.

3.3 Data Analytics

After raw data ingestion, the next step is transforming and analyzing these datasets. This requires a job manifest that unambiguously describes the job to be performed. The manifest is interpreted by a dedicated adapter that executes the defined computation. Currently, two adapters have been implemented: one using local resources and another connecting remotely to HPC systems via HPC-SerA [21]. New adapters can be easily defined to add compute environments like cloud-based function-as-a-service (FaaS). Due to the computational intensity of most analytics tasks, outsourcing to HPC systems is preferred over local computation.

To execute a job on an HPC system, pre-processing first sets up the environment exactly as specified in the job manifest. The runtime script is then submitted to the HPC batch system for actual computation. The entire environment with all dependencies is completely virtualized within a container. To guarantee reproducibility, container images are uploaded to the data lake beforehand and

retained as long as processed data links to that particular image. These images are represented as FAIR Digital Objects, like job manifests, all data types, and other entities. Additionally, only input data specified in the job manifest is available to the analytics job. Since all computation parts are completely encapsulated, this approach provides high portability and eliminates the need for runtime provenance auditing, as all degrees of freedom are either specified in the job manifest or determined during pre-processing (such as executed git commits).

After computation, post-processing ingests resulting data artifacts back into the data lake. Users can choose between two options: (i) data is ingested simply as a processed data type with only retrospective provenance information as descriptive metadata, or (ii) the manual ingestion process described in Section 3.2 is executed again. In the latter case, descriptive metadata is extracted in addition to automatically captured retrospective provenance information, enabling queries based on content rather than only data lineage.

3.4 Constantly Evolving Metadata

To achieve consistent metadata quality despite evolving schemas, we employ a simple modeling approach. First, four top-level data types are defined: raw data, processed data, container, and manifest, from which other types can be derived. Users register derived data types by defining a new schema consisting of associated attributes and an assigned name. This ensures only known data types with known, typed attributes are indexed and ingested. Based on this controlled attribute environment, a data lake-wide ontology can be maintained, defining categories, associated properties, and relations between data types and their attributes.

Versioning data types enables continuous change while maintaining necessary provenance information for full control of the data lake state. Performing analytics on raw data types and re-ingesting processed data types automatically builds a provenance-centered graph model, where derived data entities (aggregated data and metadata) are connected to their input data via the job manifest.

---

4. EXAMPLE WORKFLOW

Applying our generic data lake implementation to a project-specific use case requires several configuration steps (see Section 3.2). First, data modeling produces a JSON document specifying a FAIR Digital Object metadata template. This template contains typed attributes for raw or processed data types, the FAIR Digital Object type itself, and information for deriving a Digital Object Identifier. Next, a metadata extraction container is uploaded to the data lake, followed by configuration of the execution command and related options. At this point, data can be uploaded as the previously defined FAIR Digital Object.

The upload triggers automatic metadata extraction, which is added to the data lake in JSON format and checked for consistency.

Finally, metadata is indexed in a NoSQL database like ElasticSearch, while the actual data (the bit sequence) is stored in scalable storage such as an S3-based object store. Analysis can now be performed on this dataset, in this case on an HPC system. Similar to the metadata extraction container, an analysis container image is uploaded to the data lake. Assuming the necessary adapter (e.g., HPCSerA) is configured for the user, a job manifest can be sent to manually trigger analysis execution (see Section 3.3). The container image and input data are copied to the HPC system. If the container lacks necessary software, a git repository can be dynamically cloned and built, with all information (used commit, build commands) captured. After processing, specified artifacts are automatically ingested. Based on user configuration, the data lake either performs full ingestion to create a specific Digital Object type from these artifacts or ingests them as a generic Processed Data Type. In the latter case, only provenance information is available as descriptive metadata.

Currently, all provenance information is written as JSON and indexed in ElasticSearch, though a refined model compliant with the W3C PROV specification using a graph database is under development.

---

5. TOWARDS A GENERIC DATA ANALYTICS PIPELINE

The current implementation covers only data processing, fulfilling primary use case requirements. More complex research workflows (e.g., including experiments) can be mapped onto the data lake by adding elaborate experimental protocol descriptions to measured data metadata or uploading the protocol itself as raw data and linking it to corresponding datasets. To support collaboration, developed analysis tools must be shared. While containers provide a convenient sharing mechanism, we go further by enabling precise configuration for machine actionability on available data. Since scientific workflows typically consist of sequential steps, individual tasks must be linked into workflows.

5.1 Publishing Analysis Tasks

The general analysis process described in Section 3.3 is highly flexible and supports rapid algorithm and software development. Once software reaches maturity, developers typically publish it—a process achievable entirely within the data lake. Similar to a job manifest, a slightly modified analysis service must be defined. Other data lake users can then employ this service by uploading suitable data and initiating the analysis. Uploaded data access policies can remain completely private. The analysis service is accessed via its unique resource identifier resolvable by the data lake. While using published analysis services is

straightforward for users (as most configuration is done by developers), services should provide, upon request, minimum documentation of available options, particularly regarding suitable input data specifications.

5.2 Linking Tasks to Workflows

To obtain automated workflows, individual tasks must be chained together. Since data is the essential element of a data lake, focusing on data-centric workflows represented as directed acyclic graphs is reasonable. With necessary task definitions in place (generalized job manifests from Section 5.1), independent tasks are linked by their input and output data in a workflow manifest. Tasks should be either generalized job manifests or published analysis services. The data lake again functions as an abstraction layer between workflow definition and implementation. This requires adapters to map workflow manifests onto suitable engines like CWL [22] or Apache Airflow. These mappings also support portability outside the data lake context for local workflow re-execution and verification. The intermediary workflow manifest concept makes the data lake highly adaptable, enabling tailor-made project solutions while enforcing well-defined, homogeneous provenance information and artifact handling.

---

6. DATA ANALYTICS PIPELINE USING FAIR DIGITAL OBJECTS AND CWFR

Expressing all entities in a data lake as FAIR Digital Objects provides a novel approach and practical guideline for meeting the general requirement that data lakes need logical and physical organization [4]. This is realized through direct user interaction with different FAIR Digital Objects by calling functions specifically defined for each type. Furthermore, all FAIR Digital Objects have their own governance, which can be integrated into a data lake-wide homogeneous governance layer to achieve high scalability across back-end services.

6.1 Example Interaction with FAIR Digital Objects

As previously stated, FAIR Digital Objects are the fundamental building blocks of our architecture. Users work not directly on services but call functions that are part of specific Digital Object instances. This process is illustrated through an example of working with a metadata extraction container.

First, a user registers a Digital Object Identifier resolving to the metadata extraction container. At this level, it is an encapsulation that can contain multiple images and configurations. This step creates a corresponding object representation in the NoSQL database, where information about every Digital Object is maintained. A specific container image is then uploaded into this Digital Object using a predefined upload function. Although this image is itself a Digital Object, we will not discuss it further as it is mostly immutable. The function

call stores the container image in back-end storage and adds the image to the metadata extraction container Digital Object' s record. This record, currently implemented as a SQL table, is part of the Digital Object' s metadata and includes image-specific configurations like execution commands or optional bind mounts.

Since exact configuration is part of ingested data' s provenance information, an update function can be exposed without losing reproducibility. An update image function can add a new image to the Digital Object representing a metadata extraction container with a new configuration. When used, the Digital Object contains multiple images with different configurations logged in the record. Delete functions defined on both the wrapping metadata extraction Digital Object and the image Digital Object can only execute if the Digital Object Identifier is not present in any provenance information, ensuring reproducibility.

In this example, users interact only with predefined Digital Object functions through the REST API exposed by the web application, without needing direct interaction with underlying database or storage systems. This ensures consistent state across the entire data lake.

6.2 Providing Canonical Steps

As described in Section 5.1, data scientists and solution developers can provide analysis and metadata extraction tools to other data lake users. This requires a more generic service manifest that primarily differs from a job manifest by implicitly selecting input data through specified data types and metadata attributes, restricting possible inputs to suitable subsets. In contrast, job manifests select input data explicitly via lists or queries. Depending on configured specifications of typed metadata attributes and their exact values, fine-grained conditions can control service execution. Relying on typed attributes enables automation and achieves interoperability between data and analysis tasks. Since these services can be encapsulated in FAIR Digital Objects with associated functions, and data scientists can develop and share these pre-configured canonical steps, we foster reusability, sharing, and collaboration.

6.3 Encapsulation and Aggregations of Data and Workflows

Upon raw data ingestion, data is associated with a specific type and encapsulated with corresponding metadata. The actual data (the bit sequence) is physically stored on suitable storage systems, while extracted metadata can be split into different logical sub-units and indexed individually in various database systems, all made resolvable by a single unique resource identifier. Physical organization information can be completely abstracted from users by allowing interactions with data entities only through specific services, promoting data objectification. Importantly, associating data types with raw data must not lead to fixed schemas but should integrate smoothly with the required data catalog.

Access control lists defined on Digital Objects must also be integrated into the data lake' s overarching governance layer. Based on these implicitly built data objects, a data lake-wide ontology enables automated classification of new data within the data lake context. Further analysis tasks can add more metadata and context to data entities.

Data science projects often begin with an ad-hoc exploration phase where users frequently neglect reproducible workflows in favor of development speed. Treating data and metadata strictly as a unified object eases integration of ad-hoc capabilities. Users can work interactively on data lake data using a library that imports data objects into their programming environment. The challenge is to have a provenance monitoring system transparently running to store resulting artifacts with their provenance data. One approach is to tap into data collected by tools like ProvBook [23] and send this data along with artifacts when saved. This artifact and its metadata then become a data lake object. However, accessing the entire workflow as a whole requires arbitrary aggregations of different data lake entities. This requirement also persists in automated workflows, where raw data should be bundled with workflow manifests, resulting artifacts, used environments, and job manifests to promote reproducibility and reusability [21]. This can be done entirely in metadata space, as analysis automatically constructs a provenance-centered graph model. Simple back-traversal from all nodes to raw data yields all involved entities, and aggregating them into a single wrapper entity represents the exact workflow state. To work more easily with these entities, the general taxonomy of data lake data types can provide users with overview and access to constituents at desired granularity. Since every action is already abstracted (e.g., analysis is defined in a job manifest), the adapter concept can be reused to enable portability of these aggregations and re-execute tasks on different systems using different software stacks. To promote high interoperability, the data lake should export these aggregations in established formats like Research Objects [24], with standardized operations for inspecting entities or re-executing workflows or individual steps.

6.4 Challenges

Analysis of our current approach identified two challenges arising from our explicit implementation that may require more formal consideration.

Automating workflows using FAIR Digital Objects to achieve interoperability between data and analysis tasks inherently risks uncontrolled data multiplication and corresponding surges in compute demand. Consider a potentially misconfigured analysis task that takes n different data types as input and outputs m different artifacts. If a new data source becomes available, causing a huge increase in entities of those n data types, this could automatically trigger innumerable compute- and storage-intensive analysis tasks, even if the task author never intended such large-scale use. This could quickly exhaust data lake resources or personal quotas, causing undesirable service interruption.

Currently, wrappers called manifests abstract task and workflow definitions from underlying software stacks. Adapters are needed to map these abstract instructions into specific tool languages, requiring development and maintenance of numerous adapters. This problem can only be partially mitigated through suitable inheritance hierarchies. Additionally, different software stacks vary in functionality, making it impossible to support all state-of-the-art features of one technology while offering the same portability as a manifest using a stripped-down functionality version.

7. CONCLUSION

We present a novel data lake architecture based on typed digital objects that offers significant advantages. It allows easy adaptation by defining new data types, provides a uniform research-oriented user and process interface, and maintains technology independence. Importantly, no prior assumptions about specific use cases are made, maximizing reusability of ingested data—supported by a central data catalog containing all data regardless of processing degree. All FAIR Digital Objects are linked in a single provenance graph from raw data to final products. The additional abstraction layers (manifests and adapters) make our data lake highly portable and provide homogeneous retrospective provenance auditing independent of workflow engine or computational environment. Representing all entities as objects supports integration into application- or user-specific processes, from ad-hoc data exploration to fully automated workflows. Packaging data analytics tasks as services facilitates exchange and collaboration across teams and scientific domains.

ACKNOWLEDGEMENTS

We acknowledge funding by the "Niedersächsisches Vorab" funding line of the Volkswagen Foundation.

AUTHOR CONTRIBUTION STATEMENT

Hendrik Nolte (hendrik.nolte@gwdg.de): Conceptualization, Software, Writing—Original draft.
Philipp Wieder (philipp.wieder@gwdg.de): Writing—Review editing, Supervision.

REFERENCES

[1] Ali El-Sappagh, S.H., Ahmed Hendawi, A.M., El Bastawissy, A.H.: A proposed model for data warehouse ETL processes. Journal of King Saud University –Computer and Information Sciences 23(2), 91–104 (2011)

[2] Munappy, A.R., Bosch, J., Olsson, H.H.: Data pipeline management in practice: Challenges and opportunities. In: Morisio, M., Torchiano, M., Jedlitschka, A. (eds). Product-Focused Software Process Improvement, pp. 168–184. Springer, Cham (2020)

[3] Dixon, J.: Pentaho, Hadoop, and data lakes. Available at: https://jamesdixon.wordpress.com/2010/10/14/p hadoop-and-data-lakes/. Accessed 5 February 2022

[4] Madera, C., Laurent, A.: The next information architecture evolution: The data lake wave. In: Proceedings of the 8th International Conference on Management of Digital EcoSystems, pp. 174–180 (2016)

[5] Giebler, C., et al.: Leveraging the data lake–current state and challenges. DaWaK 2019: Big Data Analytics and Knowledge Discovery, pp. 179–188 (2019)

[6] Khine, Pwint Phyu and Wang, Zhao Shun. Data lake: a new ideology in big data era. ITM Web Conf., 17, 03025 (2018)

[7] Hai R., Quix C., Zhou C. Query Rewriting for Heterogeneous Data Lakes. Advances in Databases and Information Systems, 11019 (2018)

[8] Mathis, C. Data Lakes. Datenbank Spektrum, 17 (2017)

[9] Hai, Rihan and Geisler, Sandra and Quix, Christoph. Constance: An Intelligent Data Lake System. pages 2097–2100, 2016.

[10] Sawadogo, Pegdwendé Nicolas and Scholly, Etienne and Favre, Cécile and Ferey, Eric and Loudcher, Sabine and Darmont, Jérôme. Metadata Systems for Data Lakes: Models and Features. 2019.

[11] Hai, Rihan and Quix, Christoph and Zhou, Chen. Query Rewriting for Heterogeneous Data Lakes. In Benczúr, András and Thalheim, Bernhard and Horváth, Tomáš, editors, Advances in Databases and Information Systems, pages 35–49, Cham, 2018. Springer International Publishing.

[12] Suriarachchi, Isuru and Plale, Beth. Crossing analytics systems: a case for integrated provenance in data lakes. 2016 IEEE 12th International Conference on e-Science (e-Science), pages 349–354, 2016. IEEE.

[13] De Smedt, Koenraad and Koureas, Dimitris and Wittenburg, Peter. FAIR Digital Objects for Science: From Data Pieces to Actionable Knowledge Units. Publications, 8(2), 2020.

[14] Halevy, Alon Y and Korn, Flip and Noy, Natalya Fridman and Olston, Christopher and Polyzotis, Neoklis and Roy, Sudip and Whang, Steven Euijong.

Managing Google' s data lake: an overview of the Goods system. IEEE Data Eng. Bull., 39(3): 5-14, 2016.

[15] Bhardwaj, Anant and Bhattacherjee, Souvik and Chavan, Amit and Deshpande, Amol and Elmore, Aaron J and Madden, Samuel and Parameswaran, Aditya G. Datahub: Collaborative data science & dataset version management at scale. arXiv preprint arXiv:1409.0798, 2014.

[16] Miao, Hui and Chavan, Amit and Deshpande, Amol. Provdb: Lifecycle management of collaborative analysis workflows. Proceedings of the 2nd Workshop on Human-in-the-Loop Data Analytics, pages 1-6, 2017.

[17] Sawadogo, Pegdwendé and Darmont, Jérôme. On data lake architectures and metadata management. Journal of Intelligent Information Systems, 56(1): 97-120, 2021.

[18] Giebler, C., et al.: A zone reference model for enterprise-grade data lake management. In: 2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC), pp. 57-66 (2020)

[19] Bechhofer, Sean and De Roure, David and Gamble, Matthew and Goble, Carole and Buchan, Iain. Research Objects: Towards Exchange and Reuse of Digital Knowledge. Nature Precedings, 2010.

[20] Dai Hai Ton That and Gabriel Fils and Zhihao Yuan and Tanu Malik. Sciunits: Reusable Research Objects. CoRR, abs/1707.05731, 2017.

[21] Bingert, Sven and Köhler, Christian and Nolte, Hendrik and Alamgir, Waqar. An API to Include HPC Resources in Workflow Systems. INFOCOMP 2021: The Eleventh International Conference on Advanced Communications and Computation, pages 15-20, 2021.

[22] Amstutz, P., et al.: Common workflow language, v1. 0 (2016). Available at: https://figshare.com/articles/dataset/Common_{Workflow}*{Language}*{draft}3/3115156/2. Accessed 5 February 2022

[23] Samuel, Sheeba and König-Ries, Birgitta. ProvBook: Provenance-based Semantic Enrichment of Interactive Notebooks for Reproducibility. International Semantic Web Conference (P&D/Industry/BlueSky), 2018.

[24] Chard, Kyle and D' Arcy, Mike and Heavner, Ben and Foster, Ian and Kesselman, Carl and Madduri, Ravi and Rodriguez, Alexis and Soiland-Reyes, Stian and Goble, Carole and Clark, Kristi and Deutsch, Eric W. and Dinov, Ivo and Price, Nathan and Toga, Arthur. I' ll take that to go: Big data bags and minimal identifiers for exchange of large, complex datasets. 2016 IEEE International Conference on Big Data (Big Data), pages 319-328, 2016. IEEE.

AUTHOR BIOGRAPHY

Hendrik Nolte obtained his M.Sc. in Physics from the University of Göttingen and has worked since November 2019 as a research assistant at the Gesellschaft für wissenschaftliche Datenverarbeitung mbH. He collaborates with different institutes to build data lakes tailored to their specific needs. His particular interests lie in secure processing of sensitive data and provenance auditing. He is pursuing a Ph.D. in data lakes in collaboration with the University' s Medical School to improve MR image processing. ORCID: 0000-0003-2138-8510

Philipp Wieder received his Ph.D. degree from TU Dortmund, Germany. He is the Deputy Head of the Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen (GWDG), Germany, and teaches Data Science Infrastructures at the Georg-August-Universität Göttingen, Germany. He has contributed to distributed infrastructures, research data management, and data analytics for several years. His research interests lie in distributed systems, service level agreements, and resource scheduling. He is actively involved in the German National Research Data Initiative (NFDI) and several projects related to the European Open Science Cloud. ORCID: 0000-0002-6992-1866

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