ARCHE Suite: A Flexible Approach to Repository Metadata Management

Mateusz Żółtak
ACDH-CH OEAW
Vienna, Austria
mateusz.zoltak@oeaw.ac.at

Martina Trognitz
martina.trognitz@oeaw.ac.at
Matej Ďurčo
matej.durco@oeaw.ac.at

Abstract

This article presents an innovative approach to metadata handling implemented in the ARCHE Suite repository solution. It first discusses the technical requirements for metadata management and contrasts them with the shortcomings of existing solutions. Then, it demonstrates how the ARCHE Suite addresses those problems. After one year of use, we can assert that the approach implemented in the ARCHE Suite is viable and provides important benefits. We aim to establish the ARCHE Suite as an open-source repository solution to be used also by other parties.

1 Introduction

The Austrian Centre for Digital Humanities and Cultural Heritage (ACDH-CH) at the Austrian Academy of Sciences in Vienna runs the repository ARCHE for persistent hosting of humanities research data. ARCHE is certified as a CLARIN B-centre. Between 2017 and 2020, the underlying software technology we used was Fedora Commons version 4 with Blazegraph as a metadata store. Due to many serious shortcomings related to metadata management, the increasing amount of technical issues, and a lack of adequate alternatives, we decided to develop our own repository solution: the *ARCHE Suite*.

This paper specifies core requirements for metadata management and explains why they are not met by the existing repository solutions Fedora Commons (The Fedora Leadership Group, 2016), DSpace (Smith et al., 2013), Dataverse (King, 2007) or Invenio (Holm Nielsen, 2019) [1]. We describe how the desired features have been implemented in our solution and how they are used in our metadata management workflows. Finally, we discuss the challenges posed by our solution and summarise our first-year experiences of using it.

2 Technical Requirements for Metadata Handling

Metadata is a vital part of every data repository, indispensable for finding, understanding, and reusing the data. To fully comply with the FAIR Data Principles that emphasise machine-actionability (Wilkinson et al., 2016), data and metadata have to be machine-readable and interoperable, which poses many challenges. The most important one includes ensuring metadata interoperability and consistency while preserving its descriptive precision. Handling these challenges governs our core technical requirements for the ARCHE Suite.

2.1 Ensuring Metadata Interoperability

In the humanities and cultural heritage disciplines, the tremendous amount of metadata standards (e.g., (Riley, 2010)) stands in the way of metadata interoperability. To overcome this, CLARIN has introduced the Component Metadata Infrastructure (CMDI) (Broeder et al., 2012), a standardised (ISO 24622-1, 24622-2) metadata framework with a built-in interoperability mechanism. Another compromise widely used across all disciplines is to apply the DCMI Metadata Terms (DCMI Usage Board, 2020), with the

[1] Webpage for Fedora Commons: duraspace.org/fedora/, for DSpace: duraspace.org/dspace/, for Dataverse: dataverse.org/, and for Invenio: invenio-software.org/

Mateusz Zołłtak,Martina Trognitz and Matej Ďurčo 2022. ARCHE Suite: A Flexible Approach to Repository Metadata Management. *Selected papers from the CLARIN Annual Conference 2021. Ed. by Monica Monachini and Maria Eskevich.* Linköping Electronic Conference Proceedings 189, pp. 190–199. DOI: https://doi.org/10.3384/9789179294441

caveat of losing the potentially richer metadata to one basic common set of metadata descriptors. The repository solutions most popular among CLARIN B centres - Fedora Commons 3 and DSpace (see Tables 1 and 2) - force users to use Dublin Core (DC) as a repository-native metadata format in a more or less explicit way.

CLARIN B Centre	City	Software
ASV Leipzig	Leipzig	Fedora Commons, v3
ACDH-CH - ARCHE	Vienna	ARCHE Suite
Bayerisches Archiv für Sprachsignale	Munich	own solution
Berlin-Brandenburg Academy of Sciences and Humanities	Berlin	Fedora Commons, v3
Center of Estonian Language Resources	Tartu	META-SHARE & own (Entu)
CLARIN.SI Language Technology Centre	Ljubljana	DSpace
Eberhard Karls Universität Tübingen	Tübingen	Fedora Commons, v3 (v4 planned)
Hamburger Zentrum für Sprachkorpora	Hamburg	Fedora Commons, v3
Institut für Maschinelle Sprachverarbeitung	Stuttgart	Fedora Commons
Instituut voor de Nederlandse Taal	Leiden	own
Leibniz-Institut für Deutsche Sprache	Mannheim	Fedora Commons
LINDAT/CLARIAH-CZ	Praha	DSpace
MPI for Psycholinguistics	Nijmegen	Fedora Commons, v3 (v4 planned)
PORTULAN CLARIN	Lisboa	META-SHARE & own
Språkbanken	Gothenburg	DSpace & own (Korp, etc.)
The ILC4CLARIN Centre at the Institute for Computational Linguistics	Pisa	DSpace
The Language Bank of Finland	Helsinki	META-SHARE & own (tools)
Universität des Saarlandes	Saarbrücken	Fedora Commons
ZIM Centre for Information Modelling	Graz	Fedora Commons, v3 (v4 planned)
CLARIN-PL Language Technology Centre	Wrocław	DSpace
CLARINO Bergen Center	Bergen	DSpace
CMU-TalkBank	Pittsburgh	own (talkbank)
The CLARIN Centre at the University of Copenhagen	Copenhagen	DSpace & eSciDoc

Table 1: Repository software solutions used by CLARIN B centres according to the CLARIN's Centre Registry. The information is based on a centre's registry entry or its latest CoreTrustSeal document.

In the last years, a new concept for (meta-)data interoperability has gained prominence: the Linked Open Data (LOD) principles with five levels (stars) of compliance(Berners-Lee, 2009). Four-star LOD (Berners-Lee, 2009; Holborn, 2014) requires data to be provided in a W3C-compliant standard like RDF (W3C et al., 2014) or SPARQL (W3C et al., 2013). This is easily met by using DC because of a well-defined mapping to RDF (W3C et al., 2014; Nilsson et al., 2008). The real challenge, however, is to additionally meet the requirements of five-star LOD, which includes the use of external links (Berners-Lee, 2009; Holborn, 2014). Using external URLs as DC term values meets the requirements but results in a repository inaccessible to human users, who expect human-readable labels like *'Karl Baedeker'* rather than URIs or URLs like *'arche.acdh.oeaw.ac.at/api/35998'*. Using both URLs and human-friendly text labels as values results in problems with DC properties used multiple times (e.g. *dc:creator*) because the corresponding labels and URLs cannot be paired anymore. Overall, the only viable solution to fully adopt five-star LOD seems to be providing full RDF support.

Fedora Commons 3 does not have any RDF support. This has been changed in Fedora Commons 4 where RDF became a native metadata format. Unfortunately, Fedora Commons 4 and 5 suffer from a serious feature drop compared to the previous version (most notably the lack of a search API and dissem-

Repository software	No. of centres
ARCHE Suite	1
DSpace	7
Entu	1
eSciDoc	1
Fedora Commons	9 (v3: 6)
META-SHARE	2
own solution	6

Table 2: Popularity of repository software solutions used by CLARIN B Centres listed in Table 1.

ination methods). As a result, the adoption of Fedora Commons 4 and 5 has never become widespread. The lack of the search API has been addressed in version 6[2] but the introduced API has no RDF support. On top of that, Fedora Commons 4-6 enforce a hard-coded metadata schema for all metadata properties managed by the service (media type, binary content size, creation and modification date, etc.).

Dataverse presents a mixed approach. On the input side, it requires metadata to follow a bespoke Dataverse schema making it interoperable with other Dataverse repositories only. On the output side, metadata can be serialised into a few schemas (Institute for Quantitative Social Science, 2021), e.g. schema.org's Dataset RDF schema serialised as JSON-LD.

Invenio allows any metadata schema which can be defined using the JSON Schema (CERN et al., 2021). Such a solution can be considered RDF-compliant to a large extent because RDF metadata can be serialised as JSON-LD, and the resulting JSON-LD structure can be described in the JSON Schema. The limitation here is that there can be many valid RDF to JSON-LD serialisations, and it can be impossible to describe all of them using the JSON Schema.

DSpace defaults to Dublin Core but can be set up to accept any flat metadata schema. The limitation is that it requires metadata to be provided serialised as XML in the way that a property is represented as an XML tag and the value is the tag's content. This is incompatible with RDF in two ways: First, it forbids ingestion of RDF metadata containing URI values because in RDF-XML the URIs are stored as XML tag attributes and not as a tag's content. Second, the flat internal metadata model makes it impossible to store multiple values (URLs and labels) of the same metadata property in such a way that relationships between them (e.g.*this is a label for this URL*) are maintained. Despite the limitations on the data input side, DSpace allows a few RDF serialisation options on the output side. E.g. generation of an OAI-PMH record in the RDF-XML format with an XSLT stylesheet. Another option is to couple DSpace with a triplestore (DuraSpace, 2021).

The interoperability imperative combined with the heterogeneous formats landscape implies that most repositories have to handle more than one metadata format. The enforcement of a metadata schema (often DC) by a repository software is undesired as it either prevents the handling of domain-specific metadata schemas or requires extensive customisation. A typical way of overcoming limitations imposed on the metadata schema by a repository software is to materialise domain-specific formats as separate repository data streams. The main disadvantage of this approach is making the information redundant, which brings the risk of inconsistency. Furthermore, if a presentation format has to be changed, e.g. because a CMDI profile definition is updated, all materialised metadata records have to be regenerated and updated even if there is no change in the metadata values themselves. Similarly, if a metadata value changes, both the repository-native metadata format as well as all materialised metadata data streams have to be updated.

A better solution is to keep a single copy of all metadata values in a schema-agnostic metadata store and to allow for on-demand conversion to the desired metadata format with a templating system. DSpace and Fedora Commons have no embedded support for on-the-fly metadata conversion, Dataverse provides a fixed set of built-in conversions as described above, and Invenio allows to write custom metadata schema conversion plugins in Python.

[2]Fedora Commons 6 was released on 30th June 2021, while development on the ARCHE Suite had already begun by end of 2019. The information on Fedora Commons 6 provided here originate from its technical documentation and not from testing.

2.2 Ensuring Metadata Consistency

Ensuring metadata consistency, preferably at the ingest stage already, involves several aspects to be considered in the context of the repository management software. First, the way in which metadata checks are defined. This can be done either by specifying the allowed schema when using configuration files or by plugging in own code which performs the checks. Dataverse only supports the former method, Fedora Commons 4 only the latter, DSpace and Invenio both, and Fedora Commons 3 has no support for custom metadata checks. Executing a pluggable code only after the data were stored in the repository, like in Fedora Commons 4, does not allow for reliable metadata checks because it either allows the metadata to stay in an inconsistent state or rejects it without notifying the client about the ingestion failure.

The second aspect regarding metadata consistency concerns the software layer, in which the metadata restrictions are verified. To ensure that checks can not be bypassed, they have to be enforced by a single software component responsible for handling all data irrespective of the ingestion interface.

The third important factor is the ability to ingest the data using ACID — atomicity, consistency, isolation, durability — (Haerder and Reuter, 1983) transactions. It is especially important from the LOD perspective where consistency of one repository resource metadata may depend on a successful creation (or update) of another repository resource. Unfortunately, ACID transactions are poorly supported by existing repository solutions.

Invenio provides only a basic optimistic concurrency control on a single resource modification request level. Dataverse, DSpace and Fedora Commons 3 lack any concurrency control on the client API level and our experience with the previous ACDH-CH repository based on Fedora Commons 4 proved its transaction support to be intrinsically broken. Reasons for this are that Fedora Commons 4 and 5 lack a built-in search feature and the synchronisation with an external search engine like Solr or a triplestore is done only after the transaction commit. This makes it impossible for the ingestion client to search for any ingested data until the transaction's end. Furthermore, there is no locking system preventing parallel transactions from modifying the same repository resource (a lack of a so-called *transaction separation*). As a consequence, Fedora Commons 4 and 5 commit and rollback transaction operations provide no guarantee regarding the final state of resources modified by a transaction. Additionally, there are smaller issues like requests made within a transaction not extending the transaction timeout. The latter can lead to the failure of a large resource upload (e.g. few gigabytes in size) when the upload takes more time than the transaction timeout. The Fedora Commons 6 documentation suggests no changes in this regard.

2.3 Requirements List

To sum up, the desired repository solution should:

- Provide RDF support as the only viable way of fulfilling the five-star LOD principles
- Not enforce any particular metadata schema
- Avoid metadata duplication that comes from materialising metadata in different formats
- Allow for defining upon-ingestion metadata consistency checks in a flexible way
- Ensure metadata consistency in a way that cannot be easily bypassed
- Provide fully ACID transactions
- Allow for writing extensions in many programming languages.

Unfortunately, none of the existing solutions provides support for all the points from this list. For this reason, we have developed a new repository software: the ARCHE Suite.

3 The ARCHE Suite

The ARCHE Suite is a bespoke, in-house repository solution that we developed from scratch within half a year in 2020, including the migration from the old Fedora 4-based repository. Before going into production it underwent an external code review. The ARCHE Suite is built in a modular, service-oriented manner, consisting of multiple interconnected components that communicate through well-defined APIs (see Figure 1). All software components are available on GitHub[3] and the documentation is provided at acdh-oeaw.github.io/arche-docs/.

[3] github.com/acdh-oeaw?q=arche

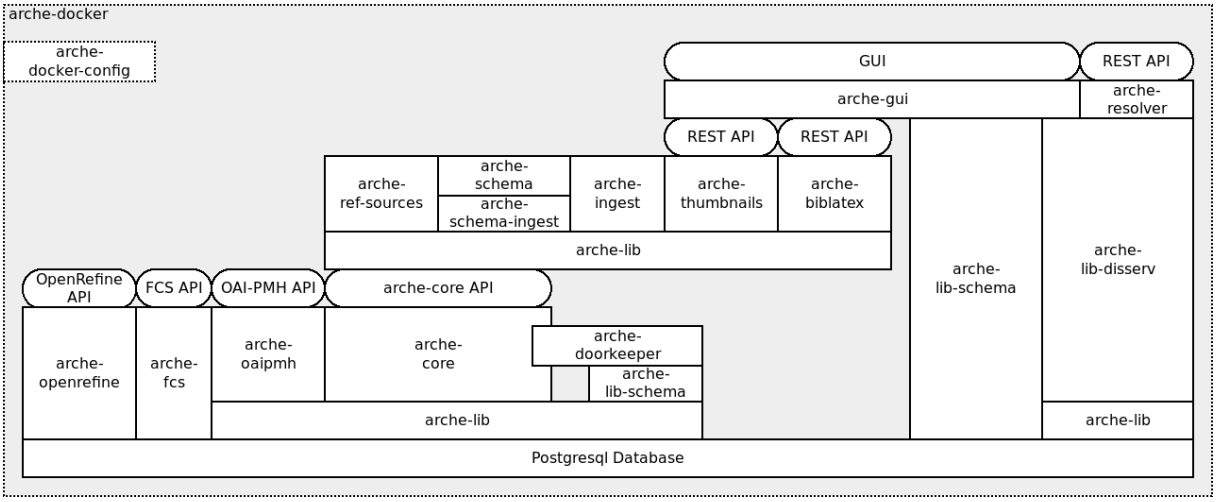

Figure 1: ARCHE Suite components. As can be seen, a microservice-based approach has been used.

Here, we detail the technical implementation of the metadata-related requirements formulated above. We focus on the developed software solution, ARCHE Suite, as opposed to ARCHE, the specific repository instance certified as a CLARIN B Centre provided by the ACDH-CH with the ARCHE Suite as the underlying technology. While ARCHE Suite is schema-agnostic, in ARCHE every resource must be described with metadata respecting a bespoke and elaborate schema (Trognitz and Ďurčo, 2018).

3.1 RDF Support

We decided to avoid dependency on a triplestore and to use a relational database as a metadata store instead. The database schema is developed in a way it can store any RDF data, i.e. does not enforce any particular RDF schema. There were two main reasons for this decision. First, using a triplestore makes it difficult to implement ACID transactions because triplestores do not recognize this concept. Second, using a relational database backend allowed us to significantly lower CPU and memory consumption of the repository (see Figure 2). On average we achieved 10 times lower memory usage and 10 to 25 times lower CPU usage. It is also important that we avoided resource usage peaks coming from the triplestore (see the middle of the right-hand column charts in Figure 2). Last but not least it sped up data ingestion by a factor of four. As a result, the ARCHE Suite supports RDF as metadata format both on the input and output side but does not natively provide a SPARQL endpoint. A dedicated search API is used instead. However, a triplestore can be paired with the ARCHE Suite either by using the plugins system described below or by periodic synchronisation. We already successfully tested the periodic synchronisation scenario.[4]

To compensate for the lack of a native SPARQL endpoint, the ARCHE Suite REST API allows to flexibly define the amount of linked data to be provided, e.g. it is possible to extend a REST API call response with metadata of *'all resources that are pointed to by a given resource'* or metadata of *'all resources that point to a given resource'* or metadata of *'all resources which can be reached by following a given RDF property'* or all of them. This solution proved to be very convenient and for performance reasons we strongly prefer it over a triplestore (see Figure 2)[5].

The data model assumes a direct connection between the metadata RDF graph and the repository structure: Every node in an ingested RDF metadata graph corresponds to a repository resource. The repository can be configured either to automatically create repository resources when an unknown RDF graph node is found in the metadata graph or to treat it as a metadata inconsistency and raise an error.

Figure 3 illustrates this connection by showing how ingested RDF nodes are processed into repository resources. In the upper part of Figure 3 an RDF graph representing a collection with the title *Collection 1* and the author *John Doe*, who comes with one custom (*https://myNmsp/Doe/John*) and one external

[4]See the arche2sparql Docker image: github.com/csae8092/arche2sparql

[5]See also ARCHE REST API scalability testing on acdh-oeaw.github.io/arche-docs/aux/metadata_api_performance.html

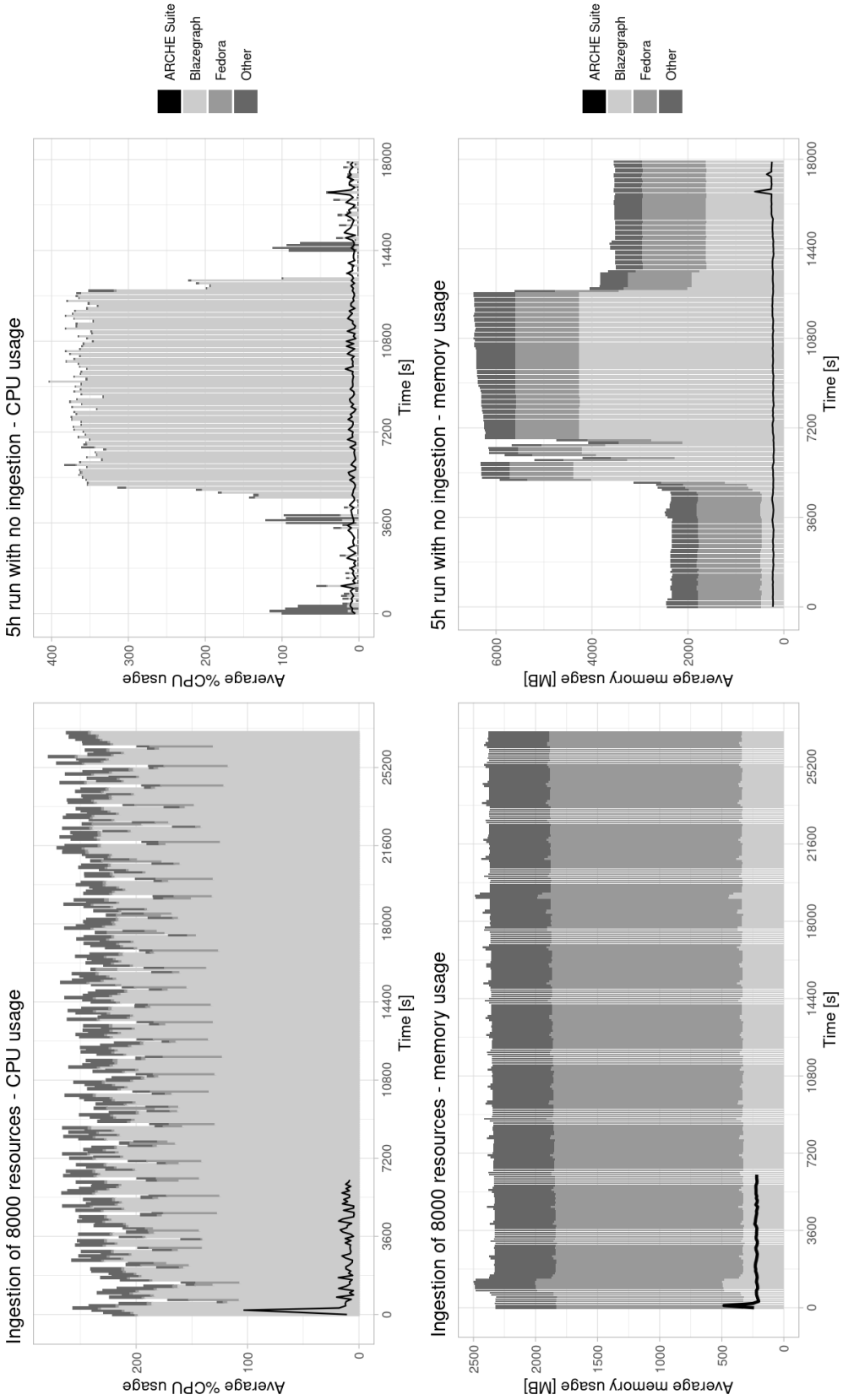

Figure 2: Comparison of hardware resources usage of the same repository implemented using Fedora Commons 4 coupled with a Blazegraph triplestore (stacked bars differentiating Fedora, Blazegraph and other components) and using the ARCHE Suite (denoted by the black line). The ingestion scenario data series for the ARCHE Suite is shorter because the ingestion finished faster.

identifier (*https://viaf/123*), is being ingested into the repository. The result is represented on the upper right-hand side of Figure 3: except for the identifiers, the RDF nodes now correspond to repository resources. Each resource was assigned an additional repository identifier (starting with *https://repoUrl/*). The identifiers of *John Doe* were imported from the RDF nodes as URIs into the repository and will be interpreted as RDF nodes upon export.

The lower part of Figure 3 represents a second ingest of another RDF graph with information about a collection with the title *Collection 2* that has the same author *John Doe*. The author *John Doe* is referenced with an already stored identifier (*https://viaf/123*) and an additional identifier is provided in the graph (*https://gnd/456*). The result from this second ingest is represented by the lower right-hand part of Figure 3: an additional repository resource for *Collection 2* was created and the additional identifier for the author is added as an URI to the already existing resource representing *John Doe*.

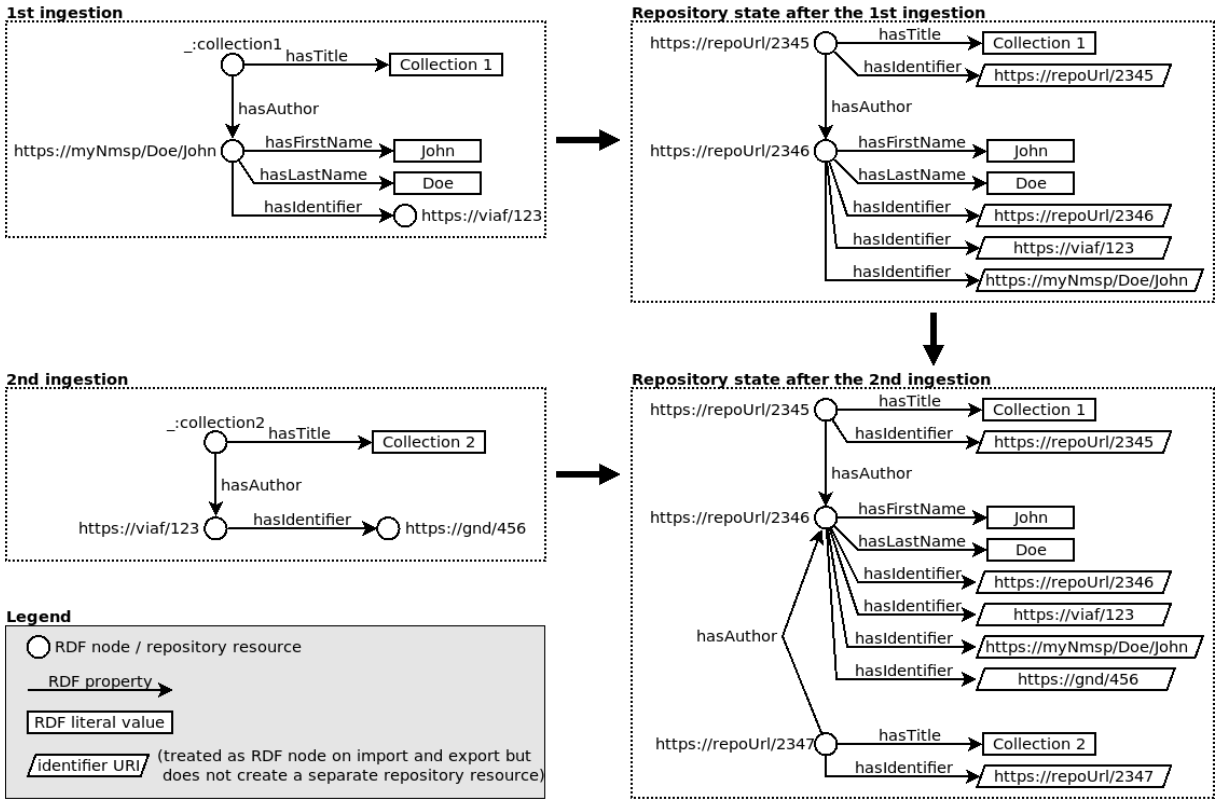

Figure 3: Example of the relation between the ingested RDF data (left) and the ARCHE Suite's internal data model (right). Identifiers are accumulated and the second ingestion does not create a new repository resource for the author but links to the already existing one.

The given example highlights that the data model used in the ARCHE Suite provides a flexible and uniform framework for handling external authoritative data. As each named entity has exactly one repository resource storing its data, e.g. information about a person (see Person *John Doe* in Figure 3), it is enough to update this resource for the change to be applied across the whole repository, i.e. all resources referring to a given person as an author (see *Collection 1* and *Collection 2* in Figure 3) do not require updating. Such an update can be done either by manual curation or by automated data retrieval from external authority files like GND[6], VIAF[7], ORCID[8], GeoNames[9], etc. We successfully employed both strategies[10].

[6] www.dnb.de/EN/Professionell/Standardisierung/GND/gnd_node.html
[7] viaf.org/
[8] orcid.org/
[9] www.geonames.org/
[10] For an example of the automatic approach see github.com/acdh-oeaw/arche-ref-sources

What makes named entities handling in ARCHE Suite even more convenient is its native support for multiple identifiers per resource. The ARCHE Suite uses a dedicated and configurable RDF property to store all possible URIs, i.e. identifiers, of a given resource. In Figure 3 this property is represented as *hasIdentifier*. All identifiers stored in this RDF property are synonymous and can be used interchangeably to denote the resource. For example, if there is a repository resource with multiple identifiers like the *John Doe* resource in the lower right-hand part of Figure 3, and a new ingestion is performed denoting *John Doe* as an author, any of *https://repoUrl/2346*, *https://myNmsp/Doe/John*, *https://viaf/123* and *https://gnd/456* can be used to refer to the already existing *John Doe* repository resource in the newly ingested RDF metadata. On a conceptual level, we can say the ARCHE Suite has built-in support for the *owl:sameAs* relation, which maps all URIs being values of the above mentioned configurable RDF property to a single repository resource.

The described data model also makes the ARCHE Suite well suited to serve as an entity reconciliation back end. In fact, one of the ARCHE Suite components is a microservice providing an OpenRefine-compatible API (see left part of Figure 1)[11], which we are already using for curation and enrichment of metadata.

3.2 Metadata Schema and Metadata Schema Conversion

The ARCHE Suite does not enforce any particular metadata schema. The only requirement is the metadata to be expressed in RDF. The RDF predicates used for storing metadata internally managed by the repository (e.g. resource checksum, last modification date, etc.) can be adjusted in the repository configuration on run time. For example, the date of a resource's last modification can be easily set either to *http://my.own.schema#creationDate*, *http://purl.org/dc/terms/created* or even *http://fedora.info/definitions/v4/repository#created* (for direct compatibility with Fedora Commons repositories).

The OAI-PMH service shipped with the ARCHE Suite allows converting metadata into various XML-serialisable formats using a flexible templating system. We have successfully implemented conversions from our internal metadata schema to CMDI profiles as well as to the schema used by Kulturpool (the Austrian Europeana aggregator) which allows us to entirely avoid materialising metadata in specific formats (cf. OAI_DC[12], Kuturpool[13] and CMDI profile p_1288172614023[14] serialisations of the same resource).

3.3 Custom Metadata Consistency Checks

The only metadata consistency check performed automatically by ARCHE Suite is the *foreign key* constraint. As described above, all nodes of the RDF metadata graph are represented by repository resources, making it impossible to remove a repository resource that is pointed to by another resource's metadata.

All other checks have to be implemented as plugins by the repository administrator. The plugins can be written in any programming language with the AMQP message queue support[15]. Plugins bind to given events (before/after metadata/binary/transaction creation/modification). When an event occurs, the plugin is provided with resource metadata in the n-triples format and is expected to return metadata in the n-triples format or to raise an error. Plugins can be used both for metadata checks and enrichment as well as for synchronisation with external services (e.g. a triplestore).

The plugins system has turned out to be a very flexible and powerful tool. Dedicated plugins have been implemented for the ARCHE repository: checking metadata property cardinalities (applying different rules for resources of different RDF classes), minting PIDs, casting metadata property values to their proper RDF datatypes (including mapping string value labels to SKOS concept URIs for properties

[11] github.com/acdh-oeaw/arche-openrefine

[12] arche.acdh.oeaw.ac.at/oaipmh/?verb=GetRecord&metadataPrefix=oai_dc&identifier=https://hdl.handle.net/21.11115/0000-000C-29F8-F

[13] arche.acdh.oeaw.ac.at/oaipmh/?verb=GetRecord&metadataPrefix=kulturpool&identifier=https://hdl.handle.net/21.11115/0000-000C-29F8-F

[14] arche.acdh.oeaw.ac.at/oaipmh/?verb=GetRecord&metadataPrefix=cmdi&identifier=https://hdl.handle.net/21.11115/0000-000C-29F8-F

[15] The are more than 20 languages with AMQP Client libraries including Java, C/C++, Python, PHP, Ruby, JavaScript/node.

with controlled vocabularies) and computing aggregated metadata property values (e.g. summary of the licence types used by resources within a collection).

3.4 Transactions Support

The ARCHE Suite provides full ACID support, although the isolation level is *read uncommitted* only. If consistency enforcement is undesired, it can be turned off by a configuration option. Importantly, all the *before event* plugins are considered part of an ACID transaction and thus, the ACID properties also extend to the plugins' actions. The transactions are backup-safe. In fact, the backup script uses its own transaction with a serialisable isolation level.

Transactions atomicity guarantees the repository can automatically get back to the pre-transaction state, i.e. perform a so-called *rollback*, if there was any error during the ingestion. This means compliance of metadata to be ingested, with the metadata schema in use can be safely checked by just performing an ingestion attempt. If there are errors, they are reported and the whole transaction is rolled back. We use such a workflow successfully for data curation and it proved to work reliably even for very large transactions, that involve an all day long ingestion of up to thousands of resources.

Due to the low isolation level, ARCHE Suite transactions have a negligible impact on the repository performance (see Figure 2) and the transaction commit is immediate. The price to be paid is a time-consuming *rollback* process taking up as much as half of the ingestion time. We did not find it troublesome in practice as the *rollback* happens only when data contains errors and the time is anyway needed to fix them.

Parallel transactions as well as parallel requests within the same transactions for faster data ingestion are also supported but discussing these complex topics in detail goes beyond the scope of this paper. More information can be found in the ARCHE Suite documentation[16].

3.5 Ingestion Workflows Automation

ARCHE Suite features, especially the flexible plugins system (see Section 3.3) coupled with the ACID transactions support (see Section 3.1), allow for automated checking of input metadata compliance with the ARCHE metadata schema and performing fully automated data ingestions in a safe way. Our latest achievement is a workflow that reads metadata from TEI/XML files, maps them to the ARCHE metadata schema, and then ingests both, the source XML files and the generated metadata into the ARCHE repository[17]. The TEI/XML data can be stored at any place accessible via the internet, e.g. in a dedicated repository on the GitHub platform. The metadata creation and repository ingestion workflow are set up as a continuous deployment workflow using GitHub Actions[18]. When data is stored inside GitHub, the workflow can be automatically triggered every time a new TEI/XML data release is made. Thanks to the atomicity of the transaction described in Section 3.4, the workflow execution comes with no risk, as the transaction is rolled back whenever an error is encountered. If there is no error, the new version of the data is published without the need for any human interaction.

4 Summary

After a year and a half of using the ARCHE Suite to run the ARCHE repository (as of March 2022, over 1.9 TB of data, 132k resources, 4.5m RDF metadata triples), we can confirm it has met our expectations. It allows us to use RDF metadata as input and output format, to perform metadata enrichment and complex consistency checks within the repository software, as well as to avoid duplicating metadata by materialising various metadata formats. Notably, using the ARCHE Suite has significantly reduced server resources consumption compared to the previous solution based on Fedora Commons 4 coupled with a Blazegraph triplestore. We are determined to develop the ARCHE Suite further and seek for cooperation with other CLARIN partners.

[16] acdh-oeaw.github.io/arche-docs/aux/parallel_ingestion.html
[17] For a practical use case see e.g. github.com/acdh-oeaw/kraus-static/actions
[18] docs.github.com/en/actions/learn-github-actions

References

Tim Berners-Lee. 2009. Linked data.

Daan Broeder, Menzo Windhouwer, Dieter Van Uytvanck, Twan Goosen, and Thorsten Trippel. 2012. Cmdi: a component metadata infrastructure. In *Describing LRs with metadata: towards flexibility and interoperability in the documentation of LR workshop programme*, volume 1.

CERN, Northwestern University, and contributors. 2021. Inveniordm - reference documentation: Metadata reference.

DCMI Usage Board. 2020. DCMI metadata terms.

DuraSpace. 2021. Dspace 7.x documentation - linked (open) data.

Theo Haerder and Andreas Reuter. 1983. Principles of transaction-oriented database recovery. *ACM Computing Surveys*, 15(4):287–317.

Timothy Holborn. 2014. What is 5 star linked data?

Lars Holm Nielsen. 2019. Inveniordm: a turn-key open source research data management platform.

Institute for Quantitative Social Science. 2021. Dataverse user guide - supported metadata export formats.

Gary King. 2007. An introduction to the dataverse network as an infrastructure for data sharing. *Sociological Methods and Research*, 36:173–199.

Mikael Nilsson, Andy Powell, Pete Johnston, and Ambjörn Naeve. 2008. Expressing dublin core metadata using the resource description framework (RDF).

Jenn Riley. 2010. Seeing standards: A visualization of the metadata universe.

MacKenzie Smith, Mary Barton, Mick Bass, Margret Branschofsky, Greg McClellan, Dave Stuve, Robert Tansley, and Julie Harford Walker. 2013. Dspace. an open source dynamic digital repository. *D-Lib Magazine*, 9(1).

The Fedora Leadership Group. 2016. Fedora and digital preservation.

Martina Trognitz and Matej Ďurčo. 2018. One schema to rule them all. the inner workings of the digital archive ARCHE. *Mitteilungen der Vereinigung Österreichischer Bibliothekarinnen und Bibliothekare*, 71(1):217–231, July.

W3C, Steve Harris, Andy Seaborne, and Eric Prud'hommeaux. 2013. Sparql 1.1 query language.

W3C, Richard Cyganiak, David Wood, Markus Lanthaler, Graham Klyne, Jeremy J. Carroll, and Brian McBride. 2014. Rdf 1.1 concepts and abstract syntax.

Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, Jildau Bouwman, Anthony J. Brookes, Tim Clark, Mercè Crosas, Ingrid Dillo, Olivier Dumon, Scott Edmunds, Chris T. Evelo, Richard Finkers, Alejandra Gonzalez-Beltran, Alasdair J.G. Gray, Paul Groth, Carole Goble, Jeffrey S. Grethe, Jaap Heringa, Peter A.C 't Hoen, Rob Hooft, Tobias Kuhn, Ruben Kok, Joost Kok, Scott J. Lusher, Maryann E. Martone, Albert Mons, Abel L. Packer, Bengt Persson, Philippe Rocca-Serra, Marco Roos, Rene van Schaik, Susanna-Assunta Sansone, Erik Schultes, Thierry Sengstag, Ted Slater, George Strawn, Morris A. Swertz, Mark Thompson, Johan van der Lei, Erik van Mulligen, Jan Velterop, Andra Waagmeester, Peter Wittenburg, Katherine Wolstencroft, Jun Zhao, and Barend Mons. 2016. The FAIR guiding principles for scientific data management and stewardship. *Scientific Data*, 3(1), March.