From MARCXML to Records in Contexts (RiC) – a configurable mapping pipeline

Documentation of the SWITCH Innovation Lab “Linked Archival Ontology and Pipeline”[1]

5.12.2022

Tobias Wildi, tobias.wildi@fhgr.ch
Fachhochschule Graubünden, Chur, Switzerland

Introduction

This paper documents the work in the context of a SWITCH Innovation Lab where MARCXML-metadata from a library catalog has been mapped to the new Records in Contexts (RiC) standard for archival metadata. The documentation will start describing the mapping in a conceptual way and then go on to demonstrate the toolchain that has been developed and applied to execute the mapping in an efficient and automated way. The mapping project has been conducted to bring catalog metadata from the library repository Patrinum as a data source to the Connectome platform. Patrinum, short for PATRImoine NUMérique, is a platform provided by the Cantonal and University Library Lausanne (BCUL) to secure the digital heritage and the digital legal deposit of the canton de Vaud (Switzerland).[2] It is based on software from Tind[3] and provides an OAI-interface for data harvesting in various formats, among others MARCXML. Connectome is a platform for linked open research data (ORD) in Switzerland, developed and run by SWITCH.[4] The vision of the Research Data Connectome is to connect and organize metadata for research sustainably across various disciplines to

[1] https://www.switch.ch/about/innovation/overview/switch-innovation-lab-linked-archival-ontology-and-pipeline/

[2] https://patrinum.ch/pages/?page=About&ln=en

[3] https://www.tind.io/

[4] https://www.switch.ch/connectome/

make it widely accessible, interoperable and valuable for the scientific community and the broader public. The Connectome project is currently developing their infrastructure for acquiring, storing and disseminating information about datasets, publications and research data providers. The infrastructure should be able to acquire, normalize, and store data from a wide variety of data providers. These providers include libraries, archives and museums, among many others. Internally, Connectome is based on its own ontology, called RESCS (Research Commons), which in turn is based on schema.org.[5] To avoid having to write separate data mappings to RESCS for each individual data provider, it was decided to map the metadata of libraries and archives to a domain-specific intermediate format as a first step and then in a second step to RESCS. For this first step, a format had to be selected that the institutions could ideally generate with the export functions of their catalog systems or that could be generated with automated mapping tools.

Records in Contexts (RiC) as a flexible archival standard fulfills these requirements of an exchange format in an ideal way. The standard is developed by the International Council on Archives (ICA).[6] It is described in a conceptual model (RiC-CM) (Experts Group on Archival Description, 2021) as well as in the form of an OWL ontology (RiC-O).[7] RiC is able to model and accomodate complex metadata structures from different source systems that are encountered in heritage institutions.

The project focused on the first mapping step from MARCXML to Records In Contexts, where as the overall pipeline looks as follows:

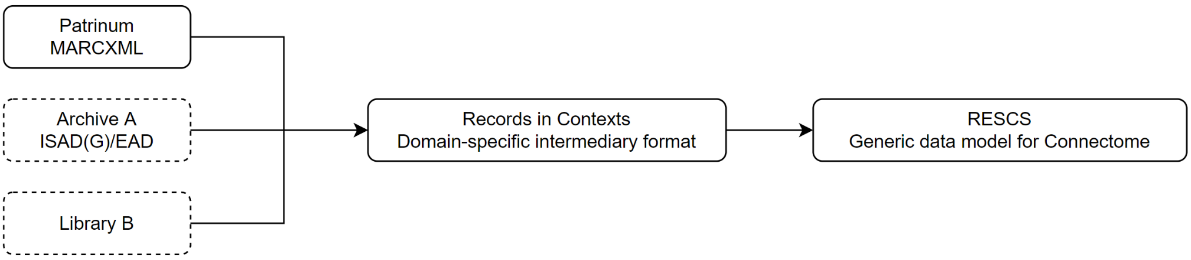

Patrinum
MARCXML
Archive A
ISAD(G)/EAD
Library B
Records in Contexts
Domain-specific intermediary format
RESCS
Generic data model for Connectome

Fig. Two step mapping pipeline: from data sources to RiC as domain-specific intermediary format and from RiC to RESCS

[5] https://rescs.org/

[6] https://www.ica.org/

[7] https://www.ica.org/standards/RiC/ontology

Goal

The goal of this paper is to document three areas regarding the data mapping from MARCXML to RiC in the context of the SWITCH Innovation Lab.

- First, the mapping has to be defined on a conceptual level. Two data models need to be aligned and checked if and where they overlap or differ from each other.
- Second, the mapping should be implemented in a flexible and versionable manner. This raises the question of technologies and tools that are currently available for this purpose.
- Third, the mapping will potentially be performed on large data sets with tools that must be automated and scaled when needed. For this purpose a toolchain and an appropriate approach for scaling has to be developed.

Mapping on a conceptual level

In the use case of Patrinum as data source, catalog metadata is internally managed in the MARC format and can be exported in various formats such as MARCXML, Dublin Core or specific formats for reference systems such as EndNote, RefWorks or BibTex. MARCXML offers by far the richest dataset of all these formats and has thus been chosen as source format for the mapping.

MARCXML is a XML encoding of the MARC standard. MARC (machine-readable cataloging) is a format for encoding bibliographic data. The standard dates back to the end of the 1960s and has been updated in many iterations ever since.[8] In many cataloging systems MARCXML is available as a rich export format. In the case of Patrinum, a MARCXML file can be exported for every record either manually via the web interface or in an automated manner via the OAI harvesting protocol. The MARCXML files have a flat structure and are human readable. Simply put, MARC database-fields are encoded with three digits, which are used for naming the tags in the MARCXML files. The subfields are encoded with letters, which again are tags in MARCXML. This leads us to the following structure of the XML file:

```
<?xml version="1.0" encoding="UTF-8"?>
<collection xmlns="http://www.loc.gov/MARC21/slim">
<record>
  <controlfield tag="001">173427</controlfield>
  <controlfield tag="005">20211007005827.0</controlfield>
```

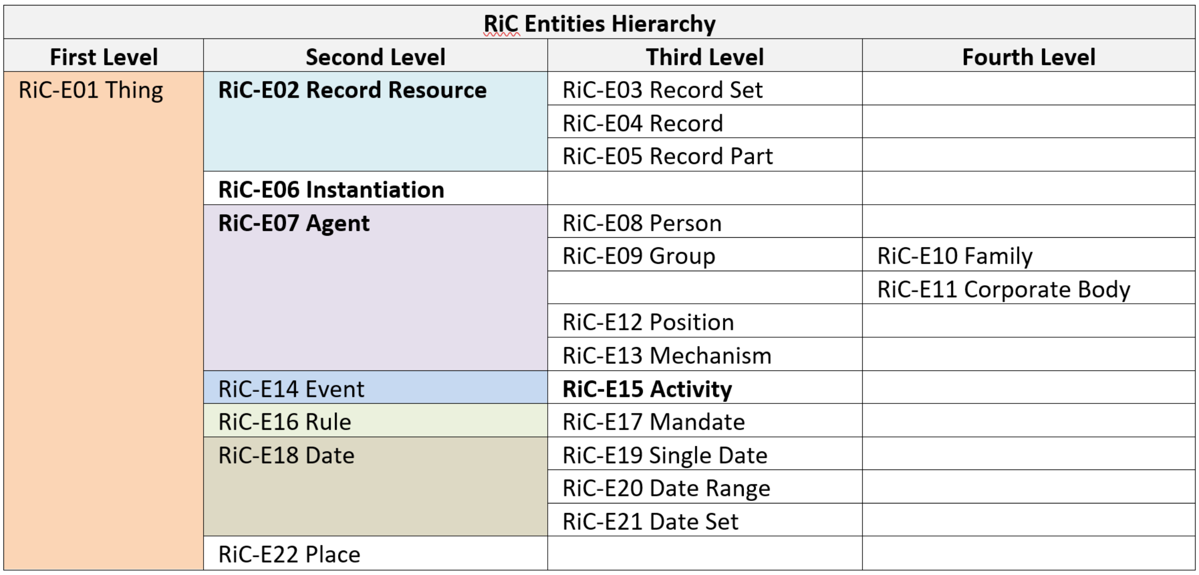
[8] https://www.loc.gov/marc/

```
  <datafield tag="024" ind1="7" ind2=" ">
    <subfield code="2">doi</subfield>
    <subfield code="a">10.22005/bcu.173427</subfield>
  </datafield>
  <datafield tag="037" ind1=" " ind2=" ">
    <subfield code="a">ISADG</subfield>
  </datafield>
  <datafield tag="041" ind1=" " ind2=" ">
    <subfield code="a">fre</subfield>
  </datafield>
…
</record>
</collection>
```

Fig. Example of a record in MARCXML-Format

To map such a serialized structure to RDF is much more straightforward than it would be for example for EAD (Encoded Archival Description)[9] with its nested hierarchical structure that is needed for encoding archival metadata according to the ISAD(G) standard.[10]

The target format for the mapping is RDF with the Records in Contexts (RiC)-ontology. RiC is based on a semantic model with different entities, attributes and relations between the entities. The main entities of RiC are:



RiC Entities Hierarchy

First Level	Second Level	Third Level	Fourth Level
RiC-E01 Thing	**RiC-E02 Record Resource**	RiC-E03 Record Set	
		RiC-E04 Record	
		RiC-E05 Record Part	
	RiC-E06 Instantiation		
	RiC-E07 Agent	RiC-E08 Person	
		RiC-E09 Group	RiC-E10 Family
			RiC-E11 Corporate Body
		RiC-E12 Position	
		RiC-E13 Mechanism	
	RiC-E14 Event	**RiC-E15 Activity**	
	RiC-E16 Rule	RiC-E17 Mandate	
	RiC-E18 Date	RiC-E19 Single Date	
		RiC-E20 Date Range	
		RiC-E21 Date Set	
	RiC-E22 Place		

Fig. Overview of the entities of RiC (Experts Group on Archival Description, 2021, p. 17)

[9] https://www.loc.gov/ead/

[10] https://www.ica.org/en/isadg-general-international-standard-archival-description-second-edition

These entities, together with some of the most important relations can be visualized as follow:

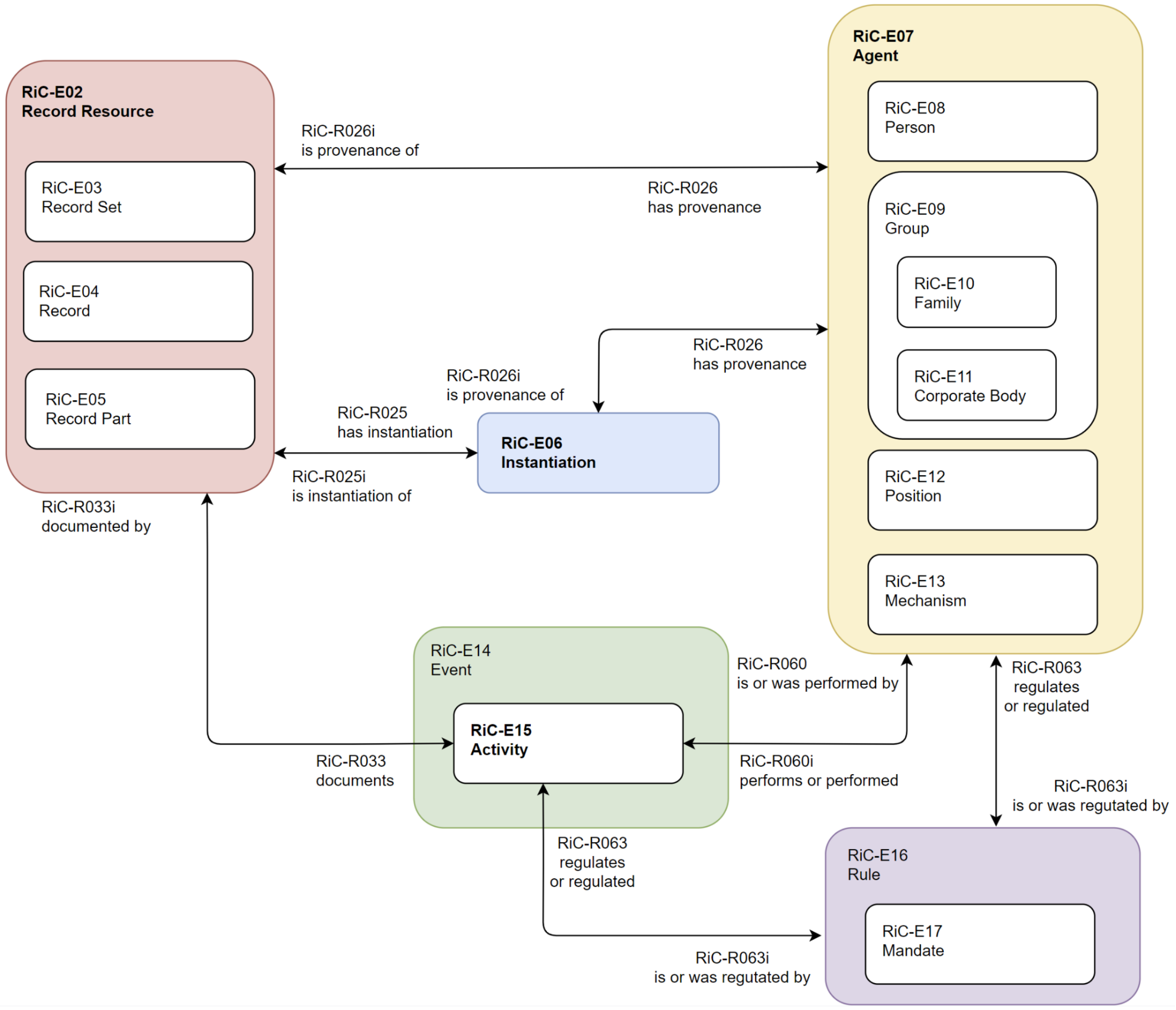


Fig. Main Entities and relations of RiC (Experts Group on Archival Description, 2021, p. 18)

In a first step of the mapping, the main sections of MARCXML were assigned to classes in the RiC-ontology. The following table doesn't show the detailed mapping yet, but gives a first overview of which sections and fields are mapped to what RiC-O-classes:

MARCXML-section	**RiC-O Class**	**RiC-O Attribute**
00X: Control Fields	rico:Record	rico:identifier
041: Language	rico:Language	rico:name

100, 110: Personal Name, Corporate Name	rico:Group	rico:name
111: Meeting, Conference	rico:Activity	rico:name
20X-24X: Title fields	rico:Title	rico:title
260: Publisher	rico: Group	rico:name
30X: Physical Description	rico:Extent	
336: Content Type	rico:Record	rico:physicalCharacteristics
340: Physical Medium	rico:PhysicalMedium	
347: File Format	rico:CarrierType	rico:name
351c: Hierarchical Level	rico:Record	rico:type

Fig. Overview of Mappings of MARCXML-sections and fields to RiC-classes

The last entry regarding the MARC-field 351c needs some explanation. In Patrinum this field is used to describe the hierarchical level of archival material, like eg. “fonds”, “series” or “item”. Since the Patrinum repository holds a lot of archival material, this MARC field is used to model the ISAD(G)-hierarchy within the library management system.

A majority of fields in the MARCXML file can be mapped as attributes of the central rico:Record class. The rico:Record class is then connected to other classes using the following relations:

Subject	**Relation**	**Object**
rico:Record	rico:hasOrHadLanguage	rico:Language
	rico:hasAuthor	rico:Group
	rico:isAssociatedWithEvent	rico:Event
	rico:hasOrHadTitle	rico:Title
	rico:hasPublisher	rico: Group
	rico:hasExtent	rico:Extent

	rico:hasContentOfType	rico:ContentType
	rico:hasCategory	rico:PhysicalMedium
	dct:file	rico:CarrierType

This leads to a different visualization of the mapping from non-RDF MARCXML as data source to RDF based metadata in RiC-O:

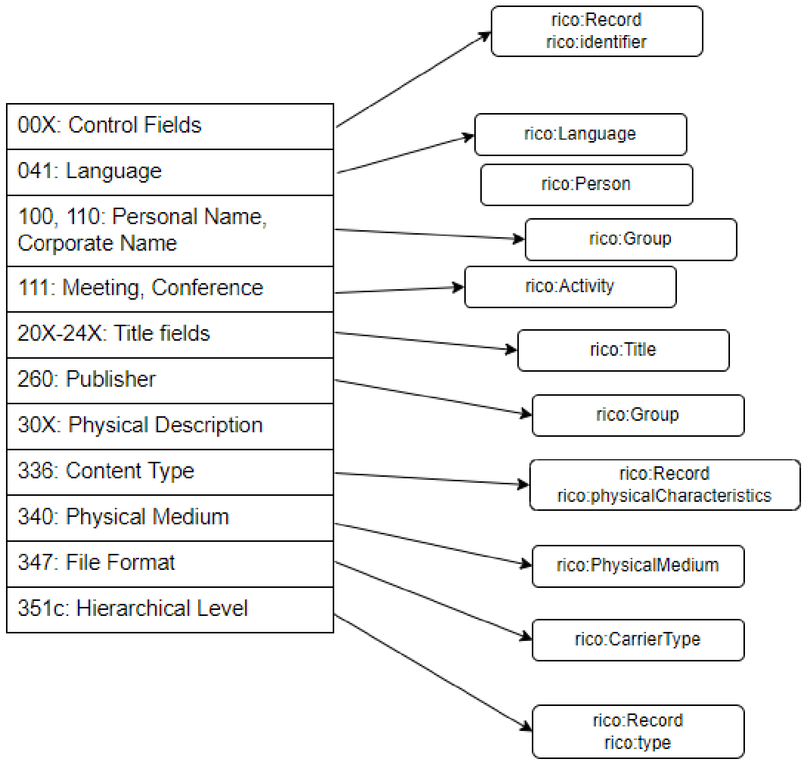


Fig. Conceptual mapping from MARC to RiC Entities and Attributes

This figure illustrates that one MARCXML record is mapped to a whole interconnected network of RiC Entities.

Implementation of the data mapping

The goal of the SWITCH Innovation Lab was to not only define the mapping on a conceptual level, but implement it with adequate tools. The prerequisites for choosing the tools were a) a certain flexibility for adapting the mapping and b) scalability if larger datasets were to be imported into Connectome.

The flexibility in the mapping definition is one key aspect in the context of the Connectome project. Data from various sources and in many different formats are mapped to the national data aggregator. If new mappings are a matter of adapting configurations and not of writing new scripts, additional data sources can much easier be added to Connectome. Furthermore, the mappings can directly be written by domain specialists without special programming knowledge. One observation in the context of Connectome is that some data sources tend to change their export- and API-formats over time. For example when CSV or Excel spreadsheets are used als source formats, new columns get added or are deleted from one version of the data set to the next version. This is a problem for data sets that are not only imported once but new data is added periodically. It saves a lot of time if only the mapping definition has to be updated and no hard coded mapping software has to be adapted, tested and recompiled.

In the present project, the need for flexibility in the data mapping led to the decision to work with a set of declarative mapping rules to define the RDF representation of the MARCXML data sources. The goal was to have a separation between the mapping definition and the conversion tool (or tools). For the mapping of heterogeneous source data to RDF, a number of tools are available that fulfill these requirements, examples are Ontop,[11] Virtuoso[12] or the RDFlib in Python.[13] The decision was finally made in favor of RML, the RDF Mapping Language (De Meester 2020). RML, as an extension of R2RML, is a mapping language to bring heterogeneous structured data and serializations to RDF (Dimou 2014) (Das 2012). To execute the conversion several processors are available, which run either locally or server-based.

At the current state, RML can handle data sources in formats like relational databases, CSV, TSV, XML and JSON.[14] This covers a lot of potential use cases for Connectome. RML definitions are written in Turtle syntax.[15] Doing this by hand can be a tedious and error prone task. As an alternative the Expressive RDF Mapper (XRM) by Zazuko can be used to create RML.[16] The XRM tool assists domain specialists and information architects to create RML mappings, the tool is both available as an Eclipse and Visual Studio Code Plugin.

[11] https://ontop-vkg.org/

[12] https://virtuoso.openlinksw.com/

[13] https://rdflib.dev/

[14] Databases are mapped with R2RML, https://www.w3.org/TR/r2rml/. R2RML is a predecessor of RML, whereas the latter has been defined as a superset of R2RML.

[15] https://www.w3.org/TR/turtle/

[16] https://zazuko.com/products/expressive-rdf-mapper/

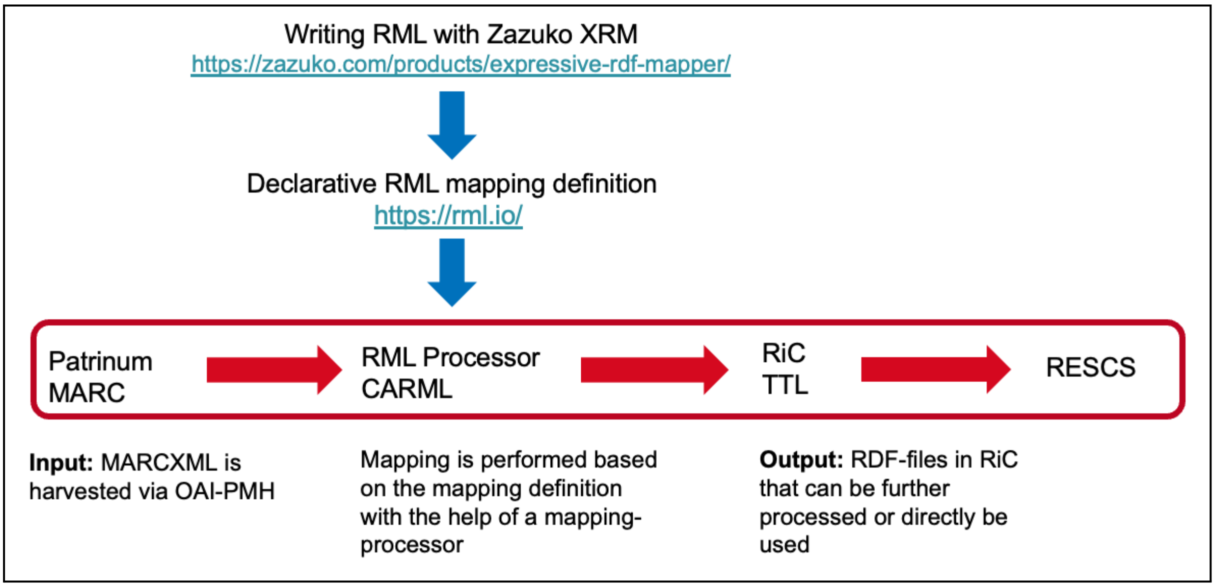

Writing RML with Zazuko XRM
https://zazuko.com/products/expressive-rdf-mapper/
Declarative RML mapping definition
https://rml.io/
Patrinum MARC
RML Processor CARML
RiC TTL
RESCS
Input: MARCXML is harvested via OAI-PMH
Mapping is performed based on the mapping definition with the help of a mapping-processor
Output: RDF-files in RiC that can be further processed or directly be used

Fig. Overview of the mapping pipeline based on RML

To perform the mapping, an RML processor is required. This processor receives an input file and executes the transformation based on the RML mapping. An overview of different processors can be found in (Arenas-Guerrero et. al. 2021), where the authors conduct a detailed survey to identify the strengths and weaknesses of different RML processors. In the context of Connectome CARML[17] is currently used for processing, other tools like RMLMapper[18] are tested.

Automating and scaling

To automate the process, a server based ETL (Extract, Transform, Load) pipeline was implemented, orchestrated by a simple bash script. Further enhancements for improved automation, coordination of acquisitions from multiple data sources, and scheduling of acquisitions could be implemented with much more elaborate tools like Apache Airflow, for example (Harenslak & Ruiter 2021) or Luigi.[19] The present pipeline consists of the following steps:

[17] https://github.com/carml/carml
[18] https://github.com/RMLio/rmlmapper-java
[19] https://luigi.readthedocs.io/

1. **Extract:** The source data are acquired from the end point https://patrinum.ch/oai2d via OAI-PMH, the Open Archives Initiative Protocol for Metadata Harvesting.[20] These metadata are harvested incrementally in the MARCXML-format and stored locally. The harvesting is done with a tool called metha-harvester, developed by the Leipzig University Library.[21]
2. **Transform:** In a first preparatory step, metha creates a large XML-file containing the metadata of all records. To perform the mapping, the large XML is split into smaller chunks, one XML-file per record. The splitting is done with the csplit-command.
   After the preparation the transformation can be executed. With the small XML-files the mapping can be streamed and hasn't to be performed on one large XML-file. For each MARCXML input file, CARML produces RDF data, in our case in JSON-LD format. This process is based on the predefined mapping configuration.
3. **Load:** The data are now in RiC format as a domain-specific intermediate format. In the Connectome Linked Data Pipeline they are converted to RESCS, the generic data model for Connectome. In the last step they are Ingested into Connectome.

A generic and simplified version of the pipeline without the OAI harvester can be downloaded from Github.[22] This version has not only been tested with MARXML files from Patrinum, but with other data sources as well.

Conclusion and Future Work

This paper shows how a knowledge graph can be created from structured XML data using a configurable and modularized approach. "Configurable" means that the mapping can be adapted at any time without having to intervene in the source code of the mapping tools. This is useful and necessary because the structure of data sources can and most probably will change over time, and in the present use case of Patrinum the data is not acquired once, but at regular time intervals. And "modularized" means that the individual building blocks developed for the pipeline can be reassembled for the acquisition of other data sources.

Records in Contexts (RiC), developed as a domain ontology for archives, proves to be a generic and flexible data model to accommodate catalog data from a library

[20] http://www.openarchives.org/OAI/openarchivesprotocol.html
[21] https://github.com/miku/metha
[22] https://github.com/wildit/marc2ric

environment. However, whether it is useful and efficient to work with an intermediate format such as RiC before data is then converted to the RESCS ontology of Connectome needs to be verified using further use cases. The advantage of this intermediate step is that data from libraries, archives and even museums can be mapped to the RiC format with relatively little effort. Plus, in the future more and more archival information systems will have direct export mechanisms to RiC. The step from RiC to the RESCS data model can then be completely automated.

Further research and development will have to be done regarding the scaling, monitoring and central control of the mapping process, especially when data mappings for several data acquisition processes run in parallel. It must be possible to monitor complex acquisition processes and in the event of errors, the processes should only be interrupted but not be terminated. It is also important to have a wide range of test data and quality control mechanisms at hand. Tests should run completely automated, both with regard to the syntax and the content (semantics) of the data, because manual testing is simply not possible at scale.

Acknowledgements

This project was supported by SWITCH in the context of a SWITCH Innovation Lab. The author would like to thank the colleagues at SWITCH, namely Kurt Baumann, Andrea Bertino, Tobias Schweizer and Sebastian Sigloch. The constructive and supportive collaboration during the project was a key factor for success. Further thanks go to BCU Lausanne, which made an important contribution by providing the Patrinum data and explaining their implementation of the MARC standard.

References

Arenas-Guerrero, Julian, Mario Scrocca, Ana Iglesias-Molina, John Toledo, Luis Pozo-Gilo, Daniel Dona, Oscar Corcho, and David Chaves-Fraga (2021). Knowledge Graph Construction with R2RML and RML: An ETL System-Based Overview. CEUR Workshop Proceedings (CEUR-WS.Org) Vol-2873. http://ceur-ws.org/Vol-2873/paper11.pdf.

Delva, T., Oo, S. M. & Assche, D. V. (2021). RML2SHACL: RDF Generation Is Shaping Up, 8.

De Meester, Ben; Heyvaert, Pieter; Delva, Thomas (2020). RDF Mapping Language (RML)

Unofficial Draft 06 October 2020. https://rml.io/specs/rml/

Dimou, Anastasia; Sande, Miel Vander; Colpaert, Pieter (2014). RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data. Proceedings of the 7th Workshop on Linked Data on the Web, 1184.

Das, Souripriya; Sundara, Seema; Cyganiak, Richard (2012). R2RML: RDB to RDF Mapping Language. W3C Recommendation 27 September 2012. https://www.w3.org/TR/r2rml/

Experts Group on Archival Description. (2021). Records in Contexts, Conceptual Model. Consultation Draft v0.2. https://www.ica.org/sites/default/files/ric-cm-02_july2021_0.pdf

Harenslak, Bas; Ruiter, Julian de. (2021). Data pipelines with Apache Airflow. Manning Publications Co.