Metashare as an ontology for the interoperability of linguistic datasets

Philipp Cimiano[1], Jorge Gracia[2], Penny Labropoulou[3], John P. McCrae[1], Víctor Rodríguez Doncel[2], and Marta Villegas[4]

[1] Cognitive Interaction Technology, Excellence Cluster, Bielefeld University, Inspiration 1, D-33619 Bielefeld, Germany,
{cimiano, jmccrae}@cit-ec.uni-bielefeld.de
[2] Ontology Engineering Group, Universidad Politécnica de Madrid, Boadilla del Monte, Madrid, Spain
{jgracia, vrodriguez}@fi.upm.es
[3] ILSP/Athena R.C., Athens, Greece,
penny@ilsp.athena-innovation.gr
[4] University Pompeu Fabra, Barcelona, Spain,
marta.villegas@upf.edu

Abstract. **Keywords:** keywords

1 Introduction

META-SHARE [10] is a subproject of META-NET aiming to create high-quality metadata for a large number of language resources and make them available in a structured form. Up until now the main methodology that META-SHARE has used to make data available is by means of XML Schema Definitions (XSD) and XML description of individual language resources conforming to these definitions. However, META-SHARE still covers only a small percentage of language resources and its resource-intensive curation methodology means it is unlikely to cover all language resources. In contrast, there have been a number of schemes that have attempted to collect much more metadata from either existing institutional repositories [3, CLARIN] or by crowd-sourcing this data from researchers [5, LRE-Map]. The former method has lead to data in different incompatible formats and the latter to noisy, incomplete and duplicative records.

In this paper, we propose a solution to these issues by means of the development of a single ontology for the representation of language resources based on the original META-SHARE schema, but represented using the Web Ontology Language [9] and building on an existing standard, namely DCAT [8]. This is a necessary step to the development of a resource called LingHub[5], which incorporates META-SHARE data as well as data from other sources and aims to make it queriable by humans and software agents.

We hope that this will improve the representation of language resource metadata in two fronts. Firstly, *RDF is super, we can SPARQL it, yada, yada... reasoning... yada, yada, Web technologies are the future*. Secondly, we hope that the

[5] http://linghub.org/

use of this ontology will enable the representation of metadata in a manner that allows existing resources to adopt a common core vocabulary, while still being able to represent specific extensions to their existing model and we evaluate this hypothesis by reference to the CLARIN and LRE-Map data models.

As an ancillary contribution of this paper, we also describe our experience in technically converting the META-SHARE schema and data to RDF. This was unusually complex as the META-SHARE schema is very complex and as such we needed to develop a new tool, we call the Lightweight Invertible XML to RDF Mapping Language (LIXR), and we demonstrate quantitatively how this ameliorated the process of conversion, and thus as a result proved to be a tool that may significantly help in future conversions from XML.

The rest of this paper is structured as follows: In section 2 we will describe the related work in the fields of language resource metadata and metadata harmonization. The development of the META-SHARE ontology is described in section 3 and then in section 4 we describe how the data was converted for use in the LingHub portal and how the ontology was used for other data sources in that resource. Finally, in section 5 we consider the broader impact of this ontology as a tool for computational linguists and as a method to realize an architecture of (linked) data-aware services.

2 Related Work [JPM]

Representation of CLARIN metadata as RDF was proposed[12].

3 The META-SHARE Ontology

3.1 Original MS XSD schema [PL]

3.2 Purpose of the ontology [MV,JG,JPM]

(e.g., why do RDF and OWL for an already defined vocabulary?)

3.3 Formal modelling and mapping issues [MV, JPM, PL]

When mapping an XML scheme to RDF there are naturally differences that must be accounted for, which generic mapping methodologies cannot accommodate without tending to a high degree of verbosity. The META-SHARE metadata model is formalised in a XSD schema that 'transcodes' a component-based model as suggested by CLARIN [4]. Essentially, the component-based approach revolves around two central concepts: *elements* and *components*. *Elements* are used to encode specific descriptive features of the resources and are linked to conceptually similar existing elements in the Dublin Core and/or the ISOcat registry. *Components* are complex elements and can be seen as bundle of semantically coherent *elements*.

In the META-SHARE XSD schema, *elements* are formalized as simple elements whereas *components* are formalized as complex-type elements. When mapping the XSD schema to RDF, *elements* can be naturally understood as properties (e.g. name, gender, etc.). *Components* (i.e. complex-type elements), however, deserve a careful analysis. General mapping rules from XSD to RDF stablish that a local element with complex type translates into an object property and a Class. An insight analysis of the META-SHARE schema showed that the straightforward application of such a principle may derive into unnecessary verbose graphs.

META-SHARE distinguishes between three kinds of *components*, namely: 'special status components', 'linked components' and 'bare components'. The former include concepts such as person and document and they can be attached to various *components* performing different roles (i.e. creator, validator, documentation, etc.). 'Linked components' can be understood as relations between *components* and include concepts such as validationReport or validator, among many others. Finally, 'bare components' are used to group together semantically coherent information (i.e. metadataInfo, validationInfo etc.). In the XSD schema, 'special status components' are formalised as complex types whereas 'linked components' are complex elements. Thus, when applying the conversion rules, the special status components become Classes and the linked components become object properties which correctly captures the semantics behind. For 'bare components' things are more complex as they are formalised as both complex elements and complex types. This means that the general conversion rule will produce an object property and the corresponding Class which, in most cases, may be unnecessary. For example: in the META-SHARE schema, the resourceInfo node contains a number of elements which organise information into coherent sets [6]:

```
resourceInfo/identificationInfo(1)/...
resourceInfo/distributionInfo(1)/...
resourceInfo/contactPerson(n)/...
resourceInfo/metadataInfo(1)/...
resourceInfo/versionInfo(1)/...
resourceInfo/validationInfo(n)/...
resourceInfo/usageInfo(1)/...
resourceInfo/resourceDocumentationInfo(1)/...
resourceInfo/resourceCreationInfo(1)/...
resourceInfo/relationInfo(1)/...
resourceInfo/resourceComponentType(1)/...
```

For elements such as the identificationInfo above, the application of the rule will produce an unnecessary node. Following [13], we identified potentially removable nodes before the actual RDFication process. The criteria applied take into account the tree structure of the nodes, their cardinality and the XPath axes. Thus, embedded complex elements with cardinalityMax=1 are identified

[6] We use XPath expressions. Number in brackets shows nodes cardinality.

as potentially removable, provided they do not contain text nor attributes. This allows for a simplification of the model, as exemplified below.

```
resource/identificationInfo/resourceName
resource/identificationInfo/description
resource/identificationInfo/resourceShortName
resource/identificationInfo/url

becomes

resource/resourceName
resource/description
resource/resourceShortName
resource/url
```

Note that such a simplification rule can be applied provided this does not derive in sibling conflicts: promoted nodes may cause naming conflicts in their new axe. Thus, a careful checking is needed in order to avoid possible clashes.

Besides the 'bare elements' described so far, a number of potentially superfluous nodes were also identified: namely complex elements with one and only one simple element. This is, for example, what happens with the path validationInfo/validationTool/targetResourceNameURI. In such cases the terminal node can be removed.

Beyond this, we made the following extensions to our mapping strategy:

- We shorten some names such as ConformanceToBestStandardsAndPractices
  JPM: Perhaps we introduce sameAs links to handle this
- Developed novel classes based on existing values, e.g., Corpus $\equiv \exists$resourceType.corpus
- Removing unnecessary properties such as `versionInfo`.
- Generalized elements such as `userNature`, `notAvailableThroughMetashare`
- Grouping similar elements under novel superclasses, e.g., `DiscourseAnnotation`, `genre`
- Extending existing classes with new values and including new properties (see section 3.5

3.4 Interface with DCAT and other vocabularies [JPM]

The META-SHARE model can be considered broadly similar to DCAT in that there are classes that are nearly an exact match to ones in DCAT for three out of four cases. DCAT's *dataset* corresponds nearly exactly to the *resource info* tag and similarly, *distributions* are similar to *distribution info* classes and *catalog record* is similar to *metadata info*. The fourth main class, *catalog* covers a level not modelled by META-SHARE.

DCAT uses Dublin Core properties for many parts of the metadata, and often these properties are in fact deeply nested into the description. For example,

language is found in several places deeply nested under six tags [7]. Similarly, it also the case that some Dublin Core properties are not directly specified in the META-SHARE model, but can be inferred from related properties, e.g., Dublin Core's 'contributor' follows from people indicated as 'annotators', 'evaluators', 'recorders' or 'validators'. Similarly, several DCAT specific-properties, such as 'download URL', are nearly exactly equivalent to those in Metashare but occur in places that do not fit the domain and range of the properties. In this particular case, it was a simple fix to move the property to the enclosing `DistributionInfo` class. Inevitably, several properties from DCAT did not have equivalences in META-SHARE, notably 'keyword' and 'byte size'. We did something about this... I am not sure what though

3.5 Licensing module [VRD, PL]

Skeleton

We describe first the Metashare schema, whose licensing information is described in an independent XML Schema file, available on git [8].

We discuss on the needs that motivated the evolution from the previous model. We describe (if not done before) also the precedure and methodology.

Short introduction on the ODRL vocabulary.

We describe the most important changes that we have introduced.

And going beyond ODRL: License Templates as an easy entry points for Semantic Web - laymans.

Example of license template, example of license. Directly in TTL. Maybe introducing a figure depecting what is metadata for resource/distrubiont/license?

4 META-SHARE in LingHub

LingHub [9] is a large resource containing information about a wide range of language resources, but unlike META-SHARE it does not directly collect this information, but instead harmonizes the metadata from a wide range of sources. In this section, we will first describe how the original META-SHARE data was translated into RDF and the alignment with DCAT [8], previously described, was achieved. Furthermore, we will then consider how we have used the META-SHARE vocabulary as a base vocabulary to align terms from other resources included in LingHub.

---

[7] `resourceInfo` > `resourceComponentType` > `corpus*` > `corpusMediaType` > `corpusVideoInfo` > `languageInfo` (* multiple tags also lead to the language information)

[8] https://github.com/metashare/META-SHARE/blob/master/misc/schema/v3.0/META-SHARE-LicenseMetadata.xsd

[9] http://linghub.org

4.1 Mapping META-SHARE to RDF [JPM]

When translating XML documents into RDF, one of the most common approaches is based on Extensible Stylesheet Language Transformations (XSLT) [15, 11, 2], which has been extended by some authors into a significant framework [7]. However, XSLT has a number of disadvantages for this task:

- The set of functions and operators suported by most processors is limited.
- Limited ability to declare new functions.
- Does not allow stream (SAX) processing of large files.
- XSLT is a one-way transformation language and it is not this possible to 'round-trip' the conversion, i.e., convert RDF to XML.
- XSLT syntax is expressed in XML and thus is very verbose and aesthetically unpleasing. For this reason, many people use alternative more compact syntaxes[10][11]

Furthermore, the META-SHARE syntax is very complex consisting of 111 complex types and 207 simple types. As such we deemed that the development of a new language for transformation and writing our converter in that language would take less development effort than writing a conversion entirely in XSLT. The mapping methodology we developed is a domain-specific langauge [6] called Lightweight Invertible XML to RDF Conversion (LIXR) and aims to improve on the situation by fixing the concerns above.

To begin with we selected the Scala programming language as the basis for LIXR as it has a proven syntactic flexibility that makes it easy to write domain-specific languages [14]. A simple example of a LIXR mapping is given below:

```
object Metashare extends eu.liderproject.lixr.Model {
  val dc = Namespace("http://purl.org/dc/elements/1.1/")
  val ms = Namespace("http://purl.org/ms-lod/MetaShare.ttl#")
  val msxml = Namespace("http://www.ilsp.gr/META-XMLSchema")

  msxml.resourceInfo --> (
    a > ms.ResourceInfo,
    handle(msxml.identificationInfo)
  )

  msxml.identificationInfo --> (
    forall(msxml.resourceName)(
        dc.title > (content @@ att("lang"))
    )
  )
}
```

In this example, we first create our model extending the basic LIXR model and define namespaces as dynamic Scala objects[12]. We then make two mapping declarations for the tags `resourceInfo` and `identificationInfo`. LIXR

[10] Compact XML: https://pythonhosted.org/compactxml/
[11] Jade: http://jade-lang.com/
[12] This is a newer feature of Scala only supported since 2.10 (Jan 2013)

(as XSLT) simply searches for a matching declaration at the root of the XML document to begin the transformation. Having matched the `resourceInfo` tag, the system first generates the triple that states that the base element has type `ms:resourceInfo`, and then 'handles' any children `identificationInfo` tags by searching for an appropriate rule for each one. For `identificationInfo` the system generates a triple using the `dc:title` property whose value is the content of the `resourceName` tag tagged with the language given by the attribute `lang`.

Name	Tags	Implementation	LoC	LoC/Tag
TBX	48	Java	2,752	57.33
CLARIN (OLAC-DMCI)	79	XSLT	404	5.11
CLARIN (OLAC-DMCI)	79	XSLT (Compact Syntax)	255	3.22
TBX	48	LIXR	197	4.10
CLARIN (OLAC-DMCI)	79	LIXR	176	2.23
MetaShare	730	LIXR	2,487	3.41

Table 1. Comparison of XML to RDF mapping implementations, by number of tags in XML schema, and non-trivial lines of code (LoC)

To evaluate the effectiveness of our approach we compared directly with two other XML to RDF transformations, we had carried out in this project, and reimplemented them using the LIXR language. In particular these were the TBX model [1] as well as the OLCA-DMCI profile of the CLARIN metadata [13]. In table 1, we see the effort to implement these using LIXR is approximately half of using XSLT and about ten times less than writing a converter from scratch.

In addition to the reduction in effort using this approach, we also note several other advantages of the LIXR approach, due to its declarative declaration

- We can easily switch to using a stream-based parse for XML (e.g., SAX) so we can process large files without having to use much memory
- A reverse mapping can be extracted that re-generates the XML from the outputted RDF
- We can extract the type, range and domain of RDF entities generated during this procedure. This export formed the initial version of the ontology described in this paper

4.2 Harmonizing other resources with META-SHARE [JPM]

LingHub brings resources from a wide-range of sources and while we can use standards such as DCAT and Dublin Core to guarantee a common representation of the basic Metadata of a resource, there does not a standard for the represenation of metadata specific to linguistics. For that reason, we use the META-SHARE model as a standard for other resources to use in LingHub. In

[13] http://catalog.clarin.eu/ds/ComponentRegistry/rest/registry/profiles/clarin.eu:cr1:p_1288172614026/xsd

particular, we will focus on the application of META-SHARE as a model to harmonize CLARIN data. The CLARIN repository describes its resources using a small common set of metadata and a larger description defined by the Component Metadata Infrastructure [4, CMDI]. These metadata schemes are extremely diverse as shown in table 2.

Component Root Tag	Institutes	Frequency
Song	1 (MI)	155,403
Session	1 (MPI)	128,673
OLAC-DcmiTerms	39	95,370
mods	1 (Utrecht)	64,632
DcmiTerms	2 (BeG,HI)	46,160
SongScan	1 (MI)	28,448
media-session-profile	1 (Munich)	22,405
SourceScan	1 (MI)	21,256
Source	1 (MI)	16,519
teiHeader	2 (BBAW, Copenhagen)	15,998

Table 2. The top 10 most frequent component types in CLARIN and the institutes that use them. Abbreviations: MI=Meertens Institute (KNAW), MPI=Max Planck Insitute (Nijmegen), BeG=Netherlands Institute for Sound and Vision, HI=Huygens Institute (KNAW), BBAW=Berlin-Brandenburg Academy of Sciences

5 Discussion

5.1 Applications of the MetaShare model (beyond LingHub) [MV]

5.2 Challenges and future outlooks [PC]

6 Conclusion [JG]

References

1. Systems to manage terminology, knowledge and content – TermBase eXchange (TBX). Tech. Rep. 30042, ISO (2008)
2. Borin, L., Dannells, D., Forsberg, M., McCrae, J.P.: Representing Swedish lexical resources in RDF with lemon. In: Proceedings of the ISWC 2014 Posters & Demonstrations Track a track within the 13th International Semantic Web Conference (2014)
3. Broeder, D., Kemps-Snijders, M., Van Uytvanck, D., Windhouwer, M., Withers, P., Wittenburg, P., Zinn, C.: A data category registry-and component-based metadata framework. In: Proceedings of the Seventh Conference on International Language Resources and Evaluation. pp. 43–47 (2010)

4. Broeder, D., Windhouwer, M., Van Uytvanck, D., Goosen, T., Trippel, T.: CMDI: a component metadata infrastructure. In: Describing LRs with metadata: towards flexibility and interoperability in the documentation of LR workshop programme. p. 1 (2012)
5. Calzolari, N., Del Gratta, R., Francopoulo, G., Mariani, J., Rubino, F., Russo, I., Soria, C.: The LRE Map. Harmonising community descriptions of resources. In: Proceedings of the Eighth Conference on International Language Resources and Evaluation. pp. 1084–1089 (2012)
6. Fowler, M., Parsons, R.: Domain-specific languages. Addison-Wesley Professional (2010)
7. Lange, C.: Krextor – an extensible XML → RDF extraction framework. Scripting and Development for the Semantic Web (SFSW) (449), 38 (2009)
8. Maali, F., Erickson, J., Archer, P.: Data catalog vocabulary (DCAT). W3C recommendation, The World Wide Web Consortium (2014)
9. Motik, B., Patel-Schneider, P.F., Parsia, B., Bock, C., Fokoue, A., Haase, P., Hoekstra, R., Horrocks, I., Ruttenberg, A., Sattler, U., Smith, M.: OWL 2 web ontology language structural specification and functional-style syntax. W3C recommendation, The World Wide Web Consortium (2012)
10. Piperidis, S.: The META-SHARE language resources sharing infrastructure: Principles, challenges, solutions. In: Proceedings of the Eighth Conference on International Language Resources and Evaluation. pp. 36–42 (2012)
11. Van Deursen, D., Poppe, C., Martens, G., Mannens, E., Walle, R.: XML to RDF conversion: a generic approach. In: Automated solutions for Cross Media Content and Multi-channel Distribution, 2008. AXMEDIS'08. International Conference on. pp. 138–144. IEEE (2008)
12. Ďurčo, M., Windhouwer, M.: From CLARIN component metadata to linked open data. In: Proceedings of the 3rd Workshop on Linked Data in Linguistics. pp. 13–17 (2014)
13. Villegas, M., Melero, M., Bel, N.: Metadata as linked open data: mapping disparate xml metadata registries into one rdf/owl registry. In: Chair), N.C.C., Choukri, K., Declerck, T., Loftsson, H., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., Piperidis, S. (eds.) Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14). European Language Resources Association (ELRA), Reykjavik, Iceland (may 2014)
14. Wampler, D., Payne, A.: Programming Scala, chap. 11. O'Reilly (2008)
15. Wüstner, E., Hotzel, T., Buxmann, P.: Converting business documents: A clarification of problems and solutions using XML/XSLT. In: Advanced Issues of E-Commerce and Web-Based Information Systems, International Workshop on. pp. 61–61. IEEE Computer Society (2002)