Supporting Linked Data Production for Cultural Heritage Institutes: The Amsterdam Museum Case Study

Victor de Boer[1], Jan Wielemaker[1], Judith van Gent[2], Michiel Hildebrand[1], Antoine Isaac[1], Jacco van Ossenbruggen[1], and Guus Schreiber[1]

[1] Department of Computer Science, VU University, Amsterdam, The Netherlands
{v.de.boer,j.wielemaker,a.isaac,
m.hildebrand,j.r.van.ossenbruggen,guus.schreiber}@vu.nl
[2] Amsterdam Museum, Amsterdam, The Netherlands
j.vangent@amsterdammuseum.nl

Abstract. Within the cultural heritage field, proprietary metadata and vocabularies are being transformed into public Linked Data. These efforts have mostly been at the level of large-scale aggregators such as Europeana where the original data is abstracted to a common format and schema. Although this approach ensures a level of consistency and interoperability, the richness of the original data is lost in the process. In this paper, we present a transparent and interactive methodology for ingesting, converting and linking cultural heritage metadata into Linked Data. The methodology is designed to maintain the richness and detail of the original metadata. We introduce the XMLRDF conversion tool and describe how it is integrated in the ClioPatria semantic web toolkit. The methodology and the tools have been validated by converting the Amsterdam Museum metadata to a Linked Data version. In this way, the Amsterdam Museum became the first 'small' cultural heritage institution with a node in the Linked Data cloud.

1 Introduction

Cultural heritage institutions such as museums, archives or libraries typically have large databases of metadata records describing the objects they curate as well as thesauri and other authority files used for these metadata fields. At the same time, in the Linked Data cloud a number of general datasets exist, such as GeoNames, VIAF or DBPedia. Importing the individual cultural heritage metadata into the Linked Data cloud and linking to these general datasets improves its reusability and integration.

While larger cultural heritage institutions such as the German National Library[1] or British National Library[2] have the resources to produce their own Linked Data, metadata from smaller institutions is currently only being added through large-scale aggregators. A prime example is Europeana, whose goals are to serve as an aggregator for cultural heritage institution data. This is to be achieved through a process of ingesting the metadata records, restructuring it to fit the Europeana Data Model and publishing it

[1] http://permalink.gmane.org/gmane.culture.libraries.ngc4lib/7544
[2] http://www.bl.uk/bibliographic/datafree.html

E. Simperl et al. (Eds.): ESWC 2012, LNCS 7295, pp. 733–747, 2012.

as Linked Data on Europeana servers. This approach ensures a level of consistency and interoperability between the datasets from different institutions. The automatic ingestion process, conversion into new dataformats and external hosting by the aggregator however creates the problem of a disconnect between the cultural heritage institute original metadata and the Linked Data version.

Rather than having Linked Data ingestion being done automatically by large aggregators, we present a methodology that is both *transparent* and *interactive*. The methodology covers data ingestion, conversion, alignment and Linked Data publication. It is highly modular with clearly recognizable data transformation steps, which can be evaluated and adapted based on these evaluations. This design allows the institute's collection managers, who are most knowledgeable about their own data, to perform or oversee the process themselves. We describe a stack of tools that allow collection managers to produce a Linked Data version of their metadata that maintains the richness of the original data including the institute-specific metadata classes and properties. By providing a mapping to a common schema interoperability is achieved. This has been previously investigated by Tordai et al. [9], of which this work is a continuation. We provide a partial validation of both these tools and the general methodology by using it to convert the metadata from the Amsterdam Museum to RDF and serving it as Linked Data.

2 Methodology Overview

To convert collection metadata into Linked Data, we here describe the general methodology. The input is the original collection metadata as provided by aggregators or individual cultural heritage institutions. The result of the workflow process is the collection metadata in semantic format (RDF). Links are established between vocabulary terms used in the collections.

Figure 1 shows the general workflow for the conversion and linking of the provided metadata. The methodology is built on the ClioPatria semantic server [11]. ClioPatria provides feedback to the user about intermediary or final RDF output in the form of an RDF browser and by providing statistics on the various RDF graphs. This feedback is crucial for the intended interactivity. The approach takes the form of a modular workflow, supported by two tools.Both the XMLRDF and Amalgame are packages of the ClioPatria semantic web toolkit. ClioPatria itself is based on SWI-Prolog and XMLRDF can therefore use its expressiveness for more complex conversions. ClioPatria and its packages are available from `http://ClioPatria.swi-prolog.org/`.

In the first step of this workflow, we ingest the XML into the ClioPatria environment. This can be either a static XML file with the metadata or the result of OAI harvesting operation. We give an example in the case study in Section 6.3. In the second step, the XML is converted to crude RDF format. This is done using the XMLRDF tool, which is documented in Section 3. This crude RDF is then rewritten in RDF adhering to the chosen metadata format, which is done using graph rewrite rules. These rules are executed by the XMLRDF tool to produce the final RDF representation of the collection metadata. An example of ingested XML, crude RDF and rewritten RDF is presented in Section 6.3 and Figure 3. Next, the user can provide an RDFS metadata schema which relates the produced classes and properties to the metadata schema of choice.

The XMLRDF tool provides support for this by presenting the user with a schema template based on the RDF data loaded. In Step 5, links are established between vocabulary concepts that are used in the collection metadata and other vocabularies. This is done using the Amalgame tool, which is documented in van Ossenbruggen et al. [7]. In Section 5, we describe Amalgame and its methodology insofar it is part of the more general conversion strategy.

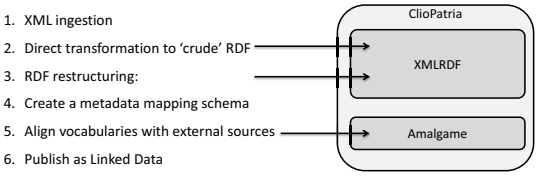

Fig. 1. General workflow for converting and linking metadata. The figure lists the various steps and relates them to either the XMLRDF or Amalgame tool or to the ClioPatria server itself.

The RDF data can be served as Linked Open Data using ClioPatria. The server performs HTTP content negotiation. If the HTTP request asks for 'text/html', the server responds with an HTML page showing the (object) metadata in human-readable form. When 'application/rdf+xml' is requested, the server responds by providing the Concise Bounded Description in RDF triples[3]. This adheres to the Linked Data requirements [1]. Additionally, a SPARQL endpoint is provided, where the data can be queried.

3 XMLRDF Conversion Tool

We here describe the XMLRDF tool, argue why it is specifically useful for converting cultural heritage data and provide a number of 'recipes' for converting such data into RDF. A more complete documentation is available [10].

When rewriting XML source data to an RDF datamodel, two extreme cases can be identified. In one case, the original metadata is forced into a target metamodel such as Dublin Core. This means that specific metadata values are copied to the new model, deleting other values. While this produces 'clean' data in the new metamodel, the original complexity and detail of the data may be lost. On the other end of the spectrum is a purely syntactic transformation to RDF, where each XML construct is converted to RDF. For datasets that are more or less 'flat' record structures, the produced RDF is usually of sufficient quality. However, cultural heritage datasets usually are inherently graph-like and will consist of more complex XML. They often include implicit links to vocabulary terms and other records, for which RDF provides a way to make these explicit. On the other hand, the XML often contains purely syntactical constructs such as some sub-trees used for meaningless grouping of elements that should not be maintained in the RDF version. In many cases, it is unclear whether a construct is meaningless or meaningful. For example, in most cases, XML element ordering is meaningless,

[3] http://www.w3.org/Submission/CBD/

but for some archival data, the ordering is important and should be explicitly modeled in the target RDF.

The XMLRDF tool is designed to translate each of these syntactic artifacts into a proper semantic model where objects and properties are typed and semantically related to a semantic metamodel such as the widely used SKOS and Dublin Core vocabularies. It does so in two main steps, shown in Figure 1. First, a syntactic conversion of the source XML into crude RDF is made. The produced RDF is as close as possible to the source metadata. This step is described in Section 3.1. The RDF produced in this way can be re-structured and enriched in the second step, which is in turn subdivided into multiple sub-steps. We describe this in Section 3.2.

The interactivity in XMLRDF stems from the fact that individual rules, or combinations of rules, can be run independently of each other and the resulting intermediate RDF can be quickly inspected through the ClioPatria browser/visualization. This allows the user to write rules, evaluate the result and adapt the rule if necessary.

3.1 Step 2: Syntactic Conversion to RDF

The core idea behind converting 'data-xml' into RDF is that every complex XML element maps to a resource (often a blank node) and every atomic attribute maps to an attribute of this bnode. Such a translation gives a valid RDF document, which can be processed using XMLRDF's graph-rewrite rules. There are a few places where we must be more subtle in the initial conversion: The **xml:lang** attribute is kept around and if we create an RDF literal, it is used to create a literal in the current language and **xmlns** declarations are ignored (they make the XML name space declarations available to the application, but the namespaces are already processed by the XML parser).

Second, we may wish to map some of our properties into rdfs:XMLLiteral or RDF dataTypes. In particular the first must be done in the first pass to avoid all the complexities of turning the RDF back into XML. In the conversion configuration, the user can specify properties and classes that are to be preserved and mapped to either an XMLLiteral or a typed RDF Literal.

The initial XML to RDF mapper uses the XML attribute and tag-names for creating RDF properties. A namespace prefix is optionally added to each XML name to create a fully qualified URI. It also 'restyles' XML identifiers: notably identifiers that contain a dot (.) are hard to process using Turtle. In the top part of Figure 3, we show the an example XML snippet and its crude RDF form.

3.2 Step 3: Enriching the RDF

The RDF produced in the steps is generally rather crude and the graph can be rewritten using rules in the XMLRDF rewrite language. The rewrite language is a production-rule system, where the syntax is modelled after Constraint Handling Rules, a committed-choice language for constraint programming [3] and the triple notation is based on Turtle/SPARQL. The transformation process for actual data however can be complicated. For these cases the rule-system allow mixing rules with arbitrary Prolog code, providing an unconstrained transformation system. We provide a (dynamically extended) library

```
title_nl @@
{ S, am:title, TitleNL }
<=>
{ S, am:title, TitleNL@nl}.

content_person @@
{PersonURI, am:name, CP},
{PersonURI, rdf:type, am:'Person'}\
{S, am:contentPersonName, CP}
<=>
{S, am:contentPersonName, PersonURI}.

use_to_altlabel @@
{UseUri, am:term, UseTerm},
{S, am:term, AltLab}\
{S, am:use, UseTerm}
 <=>
{UseUri, skos:altLabel, AltLab@nl}.

dimensions @@
{ _S, am:dimension, B},
{ B, am:dimensionValue, literal(Val)},
{ B, am:dimensionUnit, literal(Unit)}?,
{ B, am:dimensionType, literal(Type)}?,
{ B, am:dimensionPrecision, literal(Prec)}?,
{ B, am:dimensionPart, literal(Part)}?,
{ B, am:dimensionNotes, literal(Notes)}?
==>
rdf_is_bnode(B),
concat_m([Type,Val,Unit,Prec,Part,Notes], ConcatVal),
{B, rdfs:label, literal(ConcatVal)}.

assign_uris @@
{S, am:priref, literal(Priref)}\
{ S } <=>
literal_to_id(['proxy-', Priref], am, URI),
{ URI }.

clean_empty @@
{ _, _, "" } <=> true.
```

Fig. 2. Examples of XMLRDF rules used in the Amsterdam Museum conversion

of Prolog routines for typical conversion tasks. In some cases these will not satisfy, in which case expertise in programming Prolog becomes necessary. We here give an overview of the XMLRDF rewriting rules and provide a number of RDF transformation recipes. There are 3 types of production rules:

1. *Propagation rules* add triples
2. *Simplication rules* delete triples and add new triples.
3. *Simpagation rules* are in between. They match triples, delete triples and add triples,

The overall syntax for the three rule-types is (in the order above):

```
<name>? @@ <triple>* ==> <guard>? , <triple>*.
<name>? @@ <triple>* <=> <guard>? , <triple>*.
<name>? @@ <triple>* \ <triple>* <=> <guard>? , <triple>*.
```

Here, `<guard>` is an arbitrary Prolog goal. `<triple>` is a triple in a Turtle-like, but Prolog native, syntax: `{ <subject>, <predicate>, <object> }.` Any of these fields may contain a variable, written as a Prolog variable: an uppercase letter followed by zero or more letters, digits or the underscore. Resources are either fully (single-)quoted Prolog atoms (E.g. 'http://example.com/myResource', or terms of the form `<prefix>:<local>`, e.g., vra:title or ulan:'Person' (note the quotes to avoid Prolog interpretation as a variable). Figure 2 shows six rules that were used in the Amsterdam Museum example, the rules relate to recipes that will be discussed in the following sections. The rules can be run all consecutively or they can be executed one at a time allowing the user to evaluate the intermediary results. In the bottom part of Figure 3, we also show the RDF graph resulting from applying these rules to the example crude RDF.

Fixing the Node-Structure. In some cases, we might want to add triples by concatenating multiple values. The **dimensions** rule in Figure 2 is an example of this, where we add a concatenation of dimension values as an additional triple, using a new predicate. Since we do not delete the original metadata, some data duplication occurs. In addition to this, some literal fields need to be rewritten, sometimes to (multiple) new literals and sometimes to a named or bnode instance.

The simplication rules can be used to map the record-based structure to the desired structure. An example is the **use_to_altlabel** rule in Figure 2, which converts the ISO term-based thesaurus constructs to the SKOS variant. This rule takes the alternative term and re-asserts it as the skos:altLabel for the main concept.

Another use for the simplication rules is to delete triples, by having an empty action part (Prolog 'true'), as shown by the **clean_empty** rule in Figure 2. This can be used to delete triples with empty literals or otherwise obsolete triples.

Some blank nodes provide no semantic organization and can be removed, relating its properties to the parent node. At other places, intermediate instances must be created (as blank nodes or named instances).

Re-establish Internal Links. The crude RDF often contains literals where it should have references to other RDF instances. Some properties represent links to other works in the collection. The property value is typically a literal representing a unique identifier to the target object such as the collection identifier or a database key. This step replaces the predicate-value with an actual link to the target resource. The rewrite rules can use the RDF background knowledge to determine the correct URI.

Re-establish External Links. This step re-establishes links from external resources such as vocabularies which we know to be used during the annotation. In this step we only make mapping for which we are absolutely sure. I.e., if there is any ambiguity, we maintain the value as a blank node created in the previous step. An example of such a rule is the **content_person** rule shown in Figure 2.

Assign URIs to Blank Nodes Where Applicable. Any blank node we may wish to link to from the outside world needs to be given a real URI. The record-URIs are typically created from the collection-identifier. For other blank nodes, we look for distinguishing (short) literals. The construct $\{X\}$ can be used on the condition and action side of a rule. If used, there must be exactly one such construct, one for the resource to be deleted and one for the resource to be added. All resources for which the condition matches are renamed. The **assign_uris** rule in Figure 2 is an example of this. The $\{S\}$ binds the (blank node) identifier to be renamed. The Prolog guard generates a URI (see below) which replaces all occurrences of the resource.

Utility Predicates. The rewriting process is often guided by a guard which is, as already mentioned, an arbitrary Prolog goal. Because translation of repositories shares a lot of common tasks, we developed a library for these. An example of such a utility predicate is the literal_to_id predicate, which generates an URI from a literal by

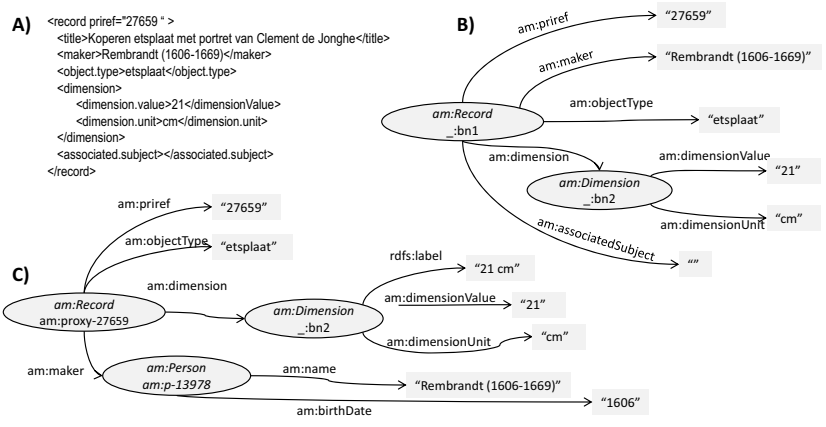

Fig. 3. Example of the different steps of XMLRDF. A) shows an XML sample snippet describing a single record is shown. B) is the result of the direct conversion to crude RDF is displayed. C) shows the graph after the rules from Figure 2 have been applied. In that final graph, the URI for the creator is used instead of the literal; a concatenated dimension label is added to the blank node; the empty 'associated subject' triple is removed and the record has a proxy-based URI.

mapping all characters that are not allowed in a (Turtle) identifier to _, as shown in the **assign_uris** rule in Figure 2.

4 Step 4: Mapping to Interoperability Layer

It is advised to maintain the original property- and type-names (classes) in the RDF because this allows to reason about possible subtle differences between the source-specific properties and properties that come from generic schemas such as Dublin Core. E.g., a creator as listed for a work in a museum for architecture is typically an architect and the work in the museum is some form of reproduction on the real physical object. If we had replaced the original creator property by dcterms:creator, this information is lost. A second reason to maintain the original property- and type-names makes it much easier to relate the RDF to the original collection data. One of the advantages of this is that it becomes easier to reuse the result of semantic enrichment in the original datasource. This implies that the converted data is normally accompanied by a schema that lists the properties and types in the data and relates them using rdfs:subPropertyOf or rdfs:subClassOf to one or more generic schemas (e.g., Dublin Core). ClioPatria provides a facility to generate a schema for a graph from the actual data, which can be used as a starting point. An example is shown in Figure 4.

ClioPatria supports RDFS entailment and any application that is built on top of its infrastructure is able to exploit the subproperty mappings that have been added in the metadata schema. In this way, the EDM datasets are integrated.

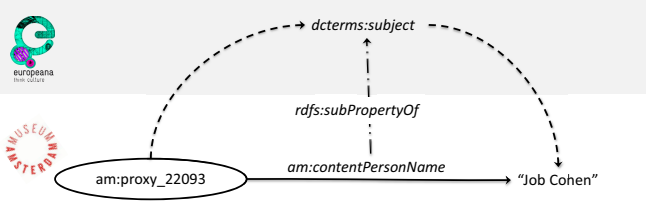

Fig. 4. RDF fragment showing how metadata mapping ensures interoperability. The bottom part of the figure shows an example triple relating an object to the name of a depicted person. Dublin Core (the metadata standard used in Europeana for object descriptions) only has a single notion of the subject of a work. By mapping the specific properties to the more general property using the rdfs:subProperty in the metadata schema, an application capable of RDFS reasoning can infer that the object has "Job Cohen" as its subject. We therefore achieve interoperability without discarding the complexity of the original data.

5 Step 5: Vocabulary Alignment Using Amalgame

Amalgame (AMsterdam ALignment GenerAtion MEtatool) is a tool for finding, evaluating and managing vocabulary alignments. The explicit goal for this platform is not to produce 'yet another alignment method' but rather support an interactive approach to vocabulary alignment where the user can combine existing matching techniques into an alignment workflow targeted to the data set at hand using a workflow setup [7]. Amalgame is freely available and documented at http://semanticweb.cs.vu.nl/amalgame/.

Which blocks to use and in what order or combination is fully controlled by the user. Furthermore, produced alignments (both intermediate and end results) can be easily evaluated to give insight in their quality. It is designed for the large but shallow vocabularies typical in the cultural heritage domain and to provide support to analyze large sets of correspondences.

Alignment within Amalgame is a process where the user iteratively applies matchers, partitions the result set, and applies new matchers or a filter. After each step the user typically analyzes the results to determine the next step. User can quickly test various matching techniques (based on labels, structure etc.) with different settings on their specific source data. Through this testing, the user gains insight into which techniques perform well, both with regards to precision (what percentage of the produced correspondences are correct) as well as coverage (how many source concepts are mapped). High-precision result sets of correspondences can be consolidated and published alongside the source RDF. To this end, Amalgame features:

- An interactive workflow composition functionality. Using this setup, the user can iteratively select various actions on vocabularies or intermediate mapping results. Filters can be applied to select subsets of concepts or of mapping results. By concatenating these actions, an alignment workflow emerges.
- A statistics function, where statistics for intermediate and end-result alignment sets are be shown.
- An evaluation view, where subsets of alignments can be evaluated manually.

6 Case Study: Amsterdam Museum

In this section, we describe how we used the above described methodology to convert the Amsterdam Museum metadata and vocabularies to five-star Linked Data that is compatible with the Europeana Data Model (EDM). This includes linking the vocabularies used in the metadata values to external sources. The input files, the intermediary and converted RDF, the schema mapping files as well as the alignment strategies are all available online at `http://semanticweb.cs.vu.nl/lod/am/data.html`, where they are listed for each step. We here present an overview.

6.1 Amsterdam Museum Metadata

The Amsterdam Museum[4] is a Dutch museum hosting cultural heritage objects related to Amsterdam and its citizens. Among these objects are paintings, drawings, prints, glass and silver objects, furniture, books, costumes, etc. all linked to Amsterdam's history and modern culture. At any given moment, around 20% of the objects are on display in the museum's exhibition rooms, while the rest is stored in storage depots.

As do many museums, the Amsterdam Museum uses a digital data management system to manage their collection metadata and authority files, in this case the proprietary Adlib Museum software[5]. As part of the museum's policy of sharing knowledge, in 2010, the Amsterdam Museum made their entire collection available online using a creative commons license. The collection can be browsed through a web-interface[6]. Second, for machine consumption, an XML REST API was provided that can be used to harvest the entire collection's metadata or retrieve specific results based on search-terms such as on creator or year. The latter API has been used extensively in multiple Cultural Heritage-related app-building challenges.

6.2 The Europeana Data Model

Europeana enables people to explore the digital resources of Europe's museums, libraries, archives and audio-visual collections. Among it's goals, Europeana will act as an aggregator for European Linked Cultural Data. The idea is that the data from individual cultural heritage institutions can be integrated by mapping them to a common metadata model: the Europeana Data Model (EDM) [5].

EDM adheres to the principles of the Web of Data and is defined using RDF. The model is designed to support the richness of the content providers metadata but also enables data enrichment from a range of third party sources. EDM supports multiple providers describing the same object, while clearly showing the provenance of all the data that links to the digital object. This is achieved by incorporating the *proxy-aggregation* mechanism from the Object Re-use and Exchange (ORE) model[7]. For our purpose, this means that an Amsterdam Museum metadata record gives rise to both a

[4] http://amsterdammuseum.nl
[5] http://www.adlibsoft.com/
[6] http://collectie.ahm.nl
[7] http://www.openarchives.org/ore/

proxy resource as well as an aggregation resource. The RDF triples that make up the object metadata (creator, dimensions etc.) have the proxy as their source while the triples that are used for provenance (data provider, rights etc.) as well as alternate representation (e.g. digital thumbnails) have the aggregation resource as their source.

For it's actual metadata, the EDM builds on proven metadata standards that are used throughout the cultural heritage domain. Dublin Core is used to represent object metadata and SKOS to represent thesauri and vocabularies. Next to these properties, a limited number of EDM-specific properties are introduced, including predicates that allow for event-centric metadata.

6.3 Producing RDF Using XMLRDF

We here report on the XML to RDF conversion, give examples of rewrite rules and discuss specific issues. For the first step of our conversion effort, we used the data as harvested through the OAI-PMH interface. This resulted in three separate XML datasets: The *object metadata*, a *thesaurus* with concepts used in the object metadata fields and a *person authority file*. The three datasets were first transformed into three RDF datasets using the pure syntactic rewriting facility of XMLRDF (step 2). Then each set was restructured using separate rewriting rules (step 3) and separate schema mapping files (step 4). We report on each of these in the next sections.

For all three datasets, we mapped the XML attributes to RDF using a base namespace *http://purl.org/collections/nl/am/*. We used these purl.org URIs since for this conversion we were not in the position to use the Amsterdam Museum namespace for our ClioPatria server.

Object Metadata. The object metadata consist of metadata records for the 73.447 objects including creator, dimensions, digital reproductions, related exhibitions etc. This dataset was converted to 6,301,012 triples in the crude RDF transform. 669,502 distinct RDF subjects were identified as well as 94 distinct properties. 497,534 of the RDF objects were identified as literals.

To enrich the crude RDF, a total of 58 XMLRDF rewrite rules were made. Of these rules, 24 were used to re-establish links to the thesaurus and 5 rules reestablished links to persons. An additional 4 rules made inter-object relations explicit. 10 rules were 'clean-up' rules . The remaining rules include rules that provide URIs to resources, rules that rewrite untyped literals into language-typed RDF literals , rules re-ifying nested blank nodes and rules combining literal values in one human-readable literal . Examples of these rules are shown in Figure 2, specifically **clean_empty**, **assign_uris**, **title_nl**, **content_person** and **dimensions**. The rules shown there are relatively simple for the sake of clarity.

The 55 rules that are executed first are not EDM-specific, as they translate the XML record structure into their RDF equivalent. The three rules that are executed last map the data to the EDM. These rules explicitly build the aggregation-proxy construct for each of the records and moves the record properties to the appropriate resource (object metadata to the proxy, provenance data and reproduction info to the aggregation).

In total, executing the rewriting rules resulted in 5,700,371 triples with 100 predicates and 933,891 subjects, of which 566,239 are blank nodes.

We constructed an RDFS mapping file relating the 100 Amsterdam Museum properties to the EDM properties through the rdfs:subPropertyOf construct. Seven properties were mapped to EDM-specific properties (ens:hasMet, ens:happenedAt, etc.) and three properties were defined as subproperties of rdfs:label, the rest of the properties are defined as subproperties of Dublin Core properties. Two Amsterdam Museum classes 'am:Exhibition' and 'am:Locat' were defined as rdfs:subClassOf of the EDM class 'ens:Event'.

Thesaurus Metadata. This dataset consists of 28.000 concepts used in the object metadata fields, including geographical terms, motifs, events etc. In the crude RDF transformation step, this was converted to 601,819 RDF triples about 160,571 distinct subjects, using 19 distinct properties. 55,780 RDF objects are literals.

Most term-based thesauri, including the AM thesaurus, have a more or less uniform structure (ISO 25964) for which the standard RDF representation is SKOS. We therefore chose to rewrite the AM thesaurus directly to SKOS format. For this purpose, we constructed 23 rewriting rules. 6 rules establish links by mapping literal values to URIs resulting in the SKOS object relations skos:broader, skos:narrower and skos:related. 4 rules mapped the original thesaurus' USE/USEFOR constructs to skos:altLabels (cf. the **use_to_altlabel** in Figure 2), 6 rules were clean-up rules. The remaining rules include rules that give URIs, give type relations and relate the skos:Concepts to a concept scheme. In total after the rewrite, 160,701 RDF triples remain, describing 28,127 subjects using 13 properties. Since the conversion already produced most of the SKOS properties, the RDFS mapping file only contains the (new) skos:ConceptScheme triples and mappings that relate the Amsterdam Museum notes to from the skos:notes.

Person Authority File. This dataset contains biographical information on 66.968 persons related to the objects or the metadata itself. This relation includes creators, past or present owners, depicted persons etc. In the crude RDF transformation step, the person authority file was converted to 301,143 RDF triples about 66,968 distinct subjects, using 21 distinct properties. 143,760 RDF objects are literals.

Since the crude RDF was already well structured and no additional literal rewriting or mapping to URIs was required, only 2 rules are needed for the people authority file. One changes the type of the records to 'Person', while the second one gives URIs to the persons. These minor translations did not change the above statistics.

Since the current version of the EDM does not specify how biographical metadata is to be represented, we mapped to properties from the RDA Group 2 metadata standard[8]. These properties include given and family names, birth and death dates etc. As a side note, informed by this conversion, this metadata set is currently considered as the EDM standard for biographical information. In total 20 rdfs:subProperty relations were defined. The am:Person class was also mapped as a rdfs:subClassOf ens:Agent.

Discussion. Of course, any (semi-)automatic transformation using rules produces some erroneous results. For example, the rules re-establishing links such as the **content_person** link in Figure 2 assume unique property values (in this case the name of the person) and

[8] http://rdvocab.info/ElementsGr2

that those values exist. Although in this case the name should be available and unique, there were three unmapped values after this rule was applied (for the remaining 2775 values, the correct URI was found). ClioPatria allows us to identify the erroneous values quickly. A new rule can be constructed for these, either rewriting them or removing the unmapped triples. Alternatively, the triples can be maintained, as was done in the Amsterdam Museum case.

Another example where the method is only partially successful is for two AM properties am:contentSubject and am:contentPersonName. These relate an object to either a concept or a person (for example a painting depicting a nobleman and a building). Dublin Core provides the dcterms:subject property which does not differentiate between the types. In the schema, we defined the AM properties as rdfs:subProperty of dcterms:subject. An application capable of RDFS reasoning can infer that the object has dcterms:subject both the person and the concept. We therefore achieve some interoperability without discarding the complexity of the original data as expressed using the properties of the 'am' namespace.

6.4 Producing Links to External Sources Using Amalgame

To illustrate step 5, we aligned the thesaurus and person authority file with a number of external sources using Amalgame. We report on the final alignment strategies.

Thesaurus. We mapped the thesaurus partly to the Dutch AATNed[9] thesaurus and partly to GeoNames[10]. The thesaurus was first split into a geographical and a non-geographical part consisting of 15851 and 11506 concepts respectively. We then aligned the Dutch part of the geographic part (953 concepts with a common ancestor "Netherlands") to the Dutch part of GeoNames using a basic label-match algorithm. This resulted in 143 unambiguous matches. We performed an informal evaluation by manually assessing a random sample of the mappings. This resulted in indicated a high quality of the matches (90%+ precision). The AM concepts for which no match was found include Amsterdam street names or even physical storage locations of art objects, which are obviously not in GeoNames.

The non-geographic part of AM was aligned with the AATNed using the same basic label match algorithm. Here, 3820 AM concepts were mapped. We then split the mapping in an unambiguous (one source is mapped to one target) and an ambiguous part (one-to-many or many-to-one). The unambiguous part was evaluated as having a high precision, the ambiguous mappings could be further disambiguated but still have a good precision of about 75%. The coverage for the non-geographic part is about 33%.

Person Authority File. The person authority file was aligned to a subset of DBpedia[11] containing persons using only the target skos:prefLabels. This resulted in 34 high quality mappings. The unmapped concepts were then aligned using the skos:altLabels as well, and then split in 453 unambiguous and 897 ambiguous matches, with estimated

[9] http://www.aat-ned.nl
[10] http://www.GeoNames.org
[11] http://dbpedia.org

precisions of 25% and 10% respectively. These could also be further filtered by hand. The people database was also aligned with Getty Union List of Artist Names (ULAN)[12], resulting in 1078 unambiguous matches with a high precision (100%) and an additional 348 ambiguous matches, with a slightly lower estimated precision. Although ULAN itself is not in the Linked Data Cloud, it is incorporated in VIAF, which is in the Linked Data cloud.

The main reason for the low coverage is that a very large number of AM-persons are not listed in ULAN as they are relatively unknown local artists, depicted or related persons, museum employees, or even organizations. Mapping to alternative sources can increase coverage here.

Produced Alignments. We identify three categories of mappings: high precision (100% correct), mid-precision (80-90% correct) and low precision (<80%). The first category can be added to a semantic layer without any further processing, whereas the second and third category might require more filtering and processing. For the Amsterdam Museum, we found high-precision mappings for 143 + 2498 + 34 + 1078 = 3753 concepts. Although this is only a fraction of the total number of concepts, the usefulness of these mappings ismuch greater as they represent the part of the concepts with which the most metadata is annotated. In total, 70.742 out of the 73.447 (96%) objects are annotated with one or more concepts or persons that have been linked, with an average of 4.3 linked concepts per object. A relatively low number of high-precision mappings were found in this case study. Current work includes raising this number by mapping to other sources and using more sophisticated Amalgame alignment strategies. We here present the results mainly as an illustration of the overall methodology.

6.5 Serving Amsterdam Museum Linked Open Data

The Amsterdam museum data, consisting of the converted datasets, the schema mapping files and the high-quality mapping files are served as Linked Open Data on the Europeana Semantic Layer (ESL)[13]. The ESL is a running instance of ClioPatria that houses other datasets that have been mapped to EDM. More information, including how to access or download the data is found at `http://semanticweb.cs.vu.nl/lod/am`.

7 Related Work

In this paper, we presented a methodology for transforming legacy data into RDF format, restructuring the RDF, establishing links and presenting it as Linked Data. A set of tools developed at the Free University Berlin provides similar functionalities. Their D2R server is a tool for publishing relational databases on the Semantic Web, by allowing data providers to construct a wrapper around the database [2]. This makes the database content browsable for both RDF and HTML browsers as well as queryable by SPARQL. The R2R tool can be used to restructure the RDF and finally the Silk tool

[12] http://www.getty.edu/research/tools/vocabularies/ulan
[13] http://semanticweb.cs.vu.nl/europeana

is used to generate links to other data sources. One difference between D2R and our approach described here is that we assume an XML output of the original data, whereas D2R acts directly as a wrapper on the data. In the cultural heritage domain, many institutes already publish their data as XML using the OAI-PMH protocol[14] as part of their normal workflow. This XML can therefor be considered their 'outward image' of the internal database and is an ideal starting place for our Linked Data conversion. A second difference is that we explicitly provide tools for interactive conversion and alignment of the data. XMLRDF is part of the ClioPatria RDF production platform, allowing for rapid assessment and evaluation of intermediary RDF.

Other tools that can be used to produce RDF include tools based on XSL transformations (XSLT). An example of such a tool is the OAI2LOD Server, which also starts from an OAI-PMH input, converts this to RDF using XSLT and provides RDF-browser and SPARQL access to the data [4]. Another example is the OAI-PMH RDFizer which is one of Simile's RDF transformation tools [8]. Such tools make use of the fact that RDF can be serialized as XML and do the conversion by restructuring the XML tree. A lot of cultural heritage institutions have relatively complex datastructures and will therefore need more complex operations in parts of the conversion [6]. Even though XSLT as a Turing-complete language has the same level of expressivity as Prolog, a number of common rewriting operations are better supported by Prolog and its rewriting rule language. Specifically, the XMLRDF rules can use Prolog and Cliopatria's RDF reasoning ability, taking existing triples into account when converting new triples.

8 Discussion and Future Work

In this paper, we presented an interactive methodology to ingesting and converting cultural heritage metadata as well as linking it to external data sources and publishing it as Linked Open Data. We described how this is supported by the ClioPatria semantic server and more specifically by the XMLRDF tool for conversion and the Amalgame tool for alignment. We illustrated this through a case study where the entire collection of the Amsterdam Museum was converted using these tools. The tools are designed to support domain experts in the conversion in such a way that they will be able to convert the legacy XML data to RDF using as much of the original ontological commitments as possible. In other words, as explained in Sections 1 and 3, we aim to retain as much as possible of the richness (and semantic choices) of the original metadata.

The tools assume knowledge of XML and RDF. For basic data conversion, (Prolog) programming skills are not necessary. However, for more complex transformations, some programming will be required. In general, transformation of rich data requires technical skills to solve the more complex cases. As described in Section 4, we use a metadata mapping to a given interoperability level and through this adhere to that set of ontological commitments. This being said, in every individual conversion, there will be semantic choices have to be made by the data expert.

Although the tools are designed to be interactive and transparent and therefore usable by the institutions' collection managers, for this case study the authors performed the

[14] http://www.openarchives.org/OAI/openarchivesprotocol.html

conversion themselves, to showcase the tools and present a number of re-usable XML-RDF conversion recipes. In the context of Europeana, the XMLRDF tool has been used by the authors, as well as by external parties to convert archival, museum and library data. A number of these converted datasets are presented in the ESL. The Amalgame tool has also been used by external parties and we are currently in the process of having alignments done by actual collection managers.

Acknowledgements. We like to thank Marijke Oosterbroek from Amsterdam Museum and Steffen Hennicke from Humboldt University for their feedback and assistance. This work was partially supported by the PrestoPRIME and EuropeanaConnect projects.

References

1. Berners-Lee, T.: Linked data - design issues (2006), http://www.w3.org/DesignIssues/~LinkedData.html
2. Bizer, C., Cyganiak, R.: D2r server – publishing relational databases on the semantic web (poster). In: International Semantic Web Conference (2003)
3. Frühwirth, T.: Introducing simplification rules. Tech. Rep. ECRC-LP-63, European Computer-Industry Research Centre, Munchen, Germany (October 1991); Presented at the Workshop Logisches Programmieren, Goosen/Berlin, Germany, and the Workshop on Rewriting and Constraints, Dagstuhl, Germany (October 1991)
4. Haslhofer, B., Schandl, B.: Interweaving oai-pmh data sources with the linked data cloud. Int. J. Metadata, Semantics and Ontologies 1(5), 17–31 (2010), http://eprints.cs.univie.ac.at/73/
5. Isaac, A.: Europeana data model primer (2010), http://version1.europeana.eu/web/europeana-project/technicaldocuments/
6. Omelayenko, B.: Porting cultural repositories to the semantic web. In: Proceedings of the First Workshop on Semantic Interoperability in the European Digital Library (SIEDL 2008), pp. 14–25 (2008)
7. van Ossenbruggen, J., Hildebrand, M., de Boer, V.: Interactive Vocabulary Alignment. In: Gradmann, S., Borri, F., Meghini, C., Schuldt, H. (eds.) TPDL 2011. LNCS, vol. 6966, pp. 296–307. Springer, Heidelberg (2011)
8. Project, T.S.: Oai-pmh rdfizer (2012), http://simile.mit.edu/wiki/OAI-PMH_RDFizer (retrieved December 2011)
9. Tordai, A., Omelayenko, B., Schreiber, G.: Semantic Excavation of the City of Books. In: Proceedings of the Semantic Authoring, Annotation and Knowledge Markup Workshop (SAAKM 2007), pp. 39–46. CEUR-WS (2007), http://ftp.informatik.rwthaachen.de/Publications/CEUR-WS/Vol-289/
10. Wielemaker, J., Hildebrand, M., van Ossenbruggen, J., Schreiber, G.: Thesaurus-Based Search in Large Heterogeneous Collections. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 695–708. Springer, Heidelberg (2008), http://dx.doi.org/10.1007/978-3-540-88564-1_44
11. Wielemaker, J., de Boer, V., Isaac, A., van Ossenbruggen, J., Hildebrand, M., Schreiber, G., Hennicke, S.: Semantic workflow tool available. EuropeanaConnect Deliverable 1.3.1 (2011), http://www.europeanaconnect.eu/documents/D1.3.1_eConnect_Workflow_automation_method_implementation_v1.0.pdf