Ontology-based Knowledge Acquisition through Semantic Profiling. An Application to the Cultural Heritage Domain

Dimitrios A. Koutsomitropoulos, George E. Paloukis and Theodore S. Papatheodorou

High Performance Information Systems Laboratory,
School of Engineering, University of Patras,
Buidling B, 26500 Patras-Rio, Greece
{kotsomit, gpalouk, tsp}@hpclab.ceid.upatras.gr

Abstract. Ontologies on the Semantic Web form a basis for representing human-conceivable knowledge in a machine-understandable manner. Ontology development for a specific knowledge domain is however a difficult task, because the produced representation has to be adequately detailed and broad enough at the same time. The CIDOC-CRM is such an ontology, pertaining to cultural heritage, which we align to the Semantic Web environment: first transforming it to OWL and then profiling it not in the usual flat metadata sense, but by refining and extending its conceptual structures, taking advantage of OWL semantics. This kind of profiling maintains applicability of the model, while enabling more expressive reasoning tasks. To this end, we construct a mechanism for acquiring implied and web-distributed information that is used to conduct and present a series of experimental inferences on the CRM profiled form.

1 Introduction

Ontologies play a key role in the Semantic Web idea. They serve as a means not only for conveying structural aspects and high level data about information, but also for providing for its understanding and intelligent manipulation by a computer machine. Furthermore, ontologies on the Semantic Web are web accessible and often distributed pieces of knowledge; a fact that at its own spawns a contemporary and challenging dimension in the way these ontologies are to be used, both from a technical as well as from a conceptual point of view.

The CIDOC Conceptual Reference Model [1] is such an ontology that attempts to model the knowledge domain of cultural heritage. As every human conceivable domain, cultural heritage is very hard to be accurately modeled. In addition and due to its nature, cultural heritage information use to be hidden in libraries and museum archives, and when available on-line are poorly or not at all structured. Moreover, the CIDOC-CRM has been recently appointed as an ISO standard (ISO-21127), a fact that further stresses its importance of use as a common conceptual basis between cultural heritage applications.

Using the CIDOC-CRM standard as our conceptual basis, we first create its machine meaningful counterpart by expressing it in the Web Ontology Language (OWL), a W3C standard. This process does not merely amount to a simple syntax transformation. Rather, taking advantage of OWL most expressive (but, simultaneously, decidable) structures we also enrich and upgrade the model, thus further narrowing the conceptual approximation [3].

One cannot go forever with this process however; there is always the danger of rendering the model too specific, thus putting its applicability in risk. To avoid this, we incorporate the OWL-specific and powerful statements in another OWL document, that involves concrete instances of the CRM's concepts and roles. This approach not only demonstrates the distributed knowledge discovery capabilities inherent in web ontologies; at the same time it suggests a *semantically enhanced application profiling* paradigm where the separating line between a standard and application-specific accuracy is thin and crucial. This kind of profiling takes the usual metadata-specific sense, where it is seen as an aggregation of disparate metadata elements [4], a step further: It does not so much deal with a horizontal extension of the ontology, but rather extends it in a semantic manner, as may dictated by a particular application.

The next step is to take advantage of this new ontology mostly by being able to reap the benefits of our semantic extensions. As standard the language and the model may be, the process for doing this is not; thus we employ a methodology [8] and implement a prototype web application, the Knowledge Discovery Interface (KDI) to be able to pose reasoning-based intelligent queries to the CRM profiled form.

This paper is mainly organized as follows: First we give an overview of the metadata application profiling idea and approaches. Then we present our process of transforming and profiling the CIDOC-CRM, pointing out our extensions and discussing semantic profiling; following there are the inferences conducted on the CRM using the KDI and their results. Finally, we summarize potential future work and the conclusions drawn from our approach.

2 Metadata Application Profiling

The need for efficient resource description in electronic archives quickly identified the lack of uniform ways for representing and maintaining information about resources. These pieces of information, known as *metadata* would therefore be organized in concrete metadata schemata produced and managed by content authorities, institutions and domain experts. The XML language eased this process by providing an official syntax for expressing both schemata and actual metadata information in machine readable format.

However, as these schemata tended to proliferate day by day, focusing on a particular domain of interest or function, there was often the case where a particular developer's needs were not satisfied by any existing schema or some elements he may find suitable where scattered over various standard implementations. Metadata application profiling came then as a natural means to overcome these obstacles while respecting the standards *raison d'être*: As defined in [3] application profiling is the assemblage of metadata elements selected from one or more metadata schemas and their combination in a compound schema. Application profiles provide the means to express principles of modularity and extensibility. The purpose of an application profile is to adapt or combine existing schemas into a package that is tailored to the functional requirements of a particular application, while retaining interoperability with the original base schemas.

There are a few ways of developing a particular metadata profile: Most obvious is to include in the same schema selected elements from different standards suitable for the particular application. If new elements are to be defined, this has to be absolutely necessary and the new elements must refer to their own namespace. Another technique is to restrict value ranges of elements, e.g. provide a specific controlled vocabulary as filler to an element or mandate specific formats for values. Finally a profile may refine existing elements defined in standards. These may involve for example the definition of sub-elements that intend to narrow the meaning of a definition or introduce some element qualifications.

Let's briefly examine an application profiling example [9]:

```
<?xml version="1.0"?>
<record
  xmlns="http://example.org/learningapp/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://example.org/learningapp/
http://example.org/learningapp/schema.xsd"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:ims="http://www.imsglobal.org/xsd/imsmd_v1p2">
  <dc:title>
    Frog maths
  </dc:title>
  <dc:description>
    Simple maths games for 5-7 year olds.
  </dc:description>
  <ims:typicallearningtime>
    <ims:datetime>
      0000-00-00T00:15
    </ims:datetime>
  </ims:typicallearningtime>
</record>
```

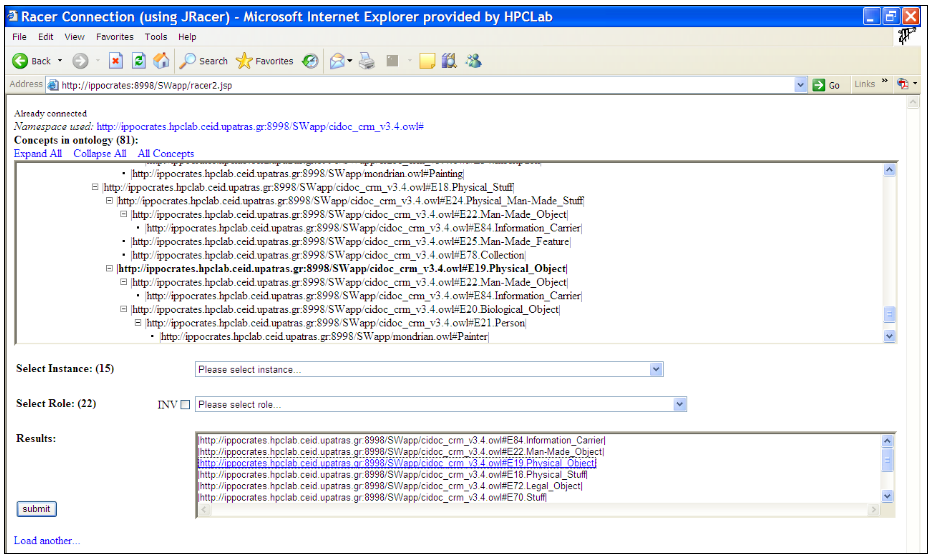
This is an instantiation of mixing Dublin Core elements with IMS learning metadata. It is noticeable that the most important means for actually implementing an application profile are *namespaces*. In the example above `dc` represents elements from the Dublin Core set, while `ims` denotes IMS-originating metadata. Namespaces play a crucial role not only in identifying provenance of distinct schemata, but also as a means to separate and then merge different elements and vocabularies.

It is clear that metadata schemata attempt to capture and convey human conceivable knowledge in the most basic unambiguous machine-compatible form: A horizontal aggregation of definitions (possibly with sub-elements) with specified value restrictions and formats that is expressed (most often) in XML. Metadata standards are perfectly successful in this manner; at the same time their representation of knowledge is considered quite poor and distanced from machine-understandability.

The Semantic Web and its ontologies give the chance of more accurate modeling of domain knowledge and thus upgrading metadata from a machine readable to machine comprehensible state. In fact ontologies are metadata schemata with precisely defined meaning and richer relations between elements and concepts of a conceptual model. With this new toolbox at hand a series of possibilities is now opened that may further ease the development of enhanced metadata profiles. These include a novel method for creating a metadata application profile, not just by combining, refining or restricting elements, but by the *semantic extension* of the model, and that is exactly what we are trying to do in the following section.

3 Transforming the CIDOC-CRM

CIDOC-CRM is currently at version 3.4.10 (aka version 4). In our work we used the initial 3.4 version, because this is the most up-to-date CRM's version that maintains a machine readable implementation. Later versions include small-scale updates regarding mostly insertion, deletion and renaming of concepts and roles in the model. Among its implementations we chose RDF(S), as the semantically richest and closest to OWL available format.


Racer Connection (using JRacer) - Microsoft Internet Explorer provided by HPCLab
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
Go
Links

http://ippocrates:8998/SWapp/racer2.jsp

Already connected
Namespace used: http://ippocrates.hpclab.ceid.upatras.gr:8998/SWapp/cidoc_crm_v3.4.owl#
Concepts in ontology (81):
Expand All   Collapse All   All Concepts

```
        • |http://ippocrates.hpclab.ceid.upatras.gr:8998/SWapp/mondrian.owl#Painting|
    ⊟ |http://ippocrates.hpclab.ceid.upatras.gr:8998/SWapp/cidoc_crm_v3.4.owl#E18.Physical_Stuff|
      ⊟ |http://ippocrates.hpclab.ceid.upatras.gr:8998/SWapp/cidoc_crm_v3.4.owl#E24.Physical_Man-Made_Stuff|
        ⊟ |http://ippocrates.hpclab.ceid.upatras.gr:8998/SWapp/cidoc_crm_v3.4.owl#E22.Man-Made_Object|
          • |http://ippocrates.hpclab.ceid.upatras.gr:8998/SWapp/cidoc_crm_v3.4.owl#E84.Information_Carrier|
        • |http://ippocrates.hpclab.ceid.upatras.gr:8998/SWapp/cidoc_crm_v3.4.owl#E25.Man-Made_Feature|
        • |http://ippocrates.hpclab.ceid.upatras.gr:8998/SWapp/cidoc_crm_v3.4.owl#E78.Collection|
      ⊟ |http://ippocrates.hpclab.ceid.upatras.gr:8998/SWapp/cidoc_crm_v3.4.owl#E19.Physical_Object|
        ⊟ |http://ippocrates.hpclab.ceid.upatras.gr:8998/SWapp/cidoc_crm_v3.4.owl#E22.Man-Made_Object|
          • |http://ippocrates.hpclab.ceid.upatras.gr:8998/SWapp/cidoc_crm_v3.4.owl#E84.Information_Carrier|
        ⊟ |http://ippocrates.hpclab.ceid.upatras.gr:8998/SWapp/cidoc_crm_v3.4.owl#E20.Biological_Object|
          ⊟ |http://ippocrates.hpclab.ceid.upatras.gr:8998/SWapp/cidoc_crm_v3.4.owl#E21.Person|
            • |http://ippocrates.hpclab.ceid.upatras.gr:8998/SWapp/mondrian.owl#Painter|
```

Select Instance: (15) Please select instance...

Select Role: (22) INV ☐ Please select role...

Results:

```
|http://ippocrates.hpclab.ceid.upatras.gr:8998/SWapp/cidoc_crm_v3.4.owl#E84.Information_Carrier|
|http://ippocrates.hpclab.ceid.upatras.gr:8998/SWapp/cidoc_crm_v3.4.owl#E22.Man-Made_Object|
|http://ippocrates.hpclab.ceid.upatras.gr:8998/SWapp/cidoc_crm_v3.4.owl#E19.Physical_Object|
|http://ippocrates.hpclab.ceid.upatras.gr:8998/SWapp/cidoc_crm_v3.4.owl#E18.Physical_Stuff|
|http://ippocrates.hpclab.ceid.upatras.gr:8998/SWapp/cidoc_crm_v3.4.owl#E72.Legal_Object|
|http://ippocrates.hpclab.ceid.upatras.gr:8998/SWapp/cidoc_crm_v3.4.owl#E70.Stuff|
```

submit

Load another...

Fig. 1. CIDOC-CRM taxonomy as shown by the KDI.

As of Jan. 2005 there exists an OWL transcription of the CRM's RDF document. However this version adds only role specific constructs (inversion, transitivity etc) which, semantically, do not exceed OWL Lite.

Version 3.4 includes about 84 concepts and 139 roles, not counting their inverses (that is, a total of 278 roles) (Figure 1). In terms of expressivity, the CRM employs structures enabled by RDF(S), which may be summarized as follows:

- Concepts as well as roles are organized in hierarchies.
- For every role, concepts are defined that form its domain and its range.
- For every role, its inverse is also defined, as a separate role, because RDF(S) cannot implicitly express inversion relation between two roles.
- There is no distinction between object and datatype properties (roles) as in OWL; Rather, roles that are equivalent to datatype properties have rdf:Literal as their range.

In the following we discuss our process of transforming CIDOC-CRM in terms first of its syntactic transcription and then its semantic enhancement and profiling (Figure 2).

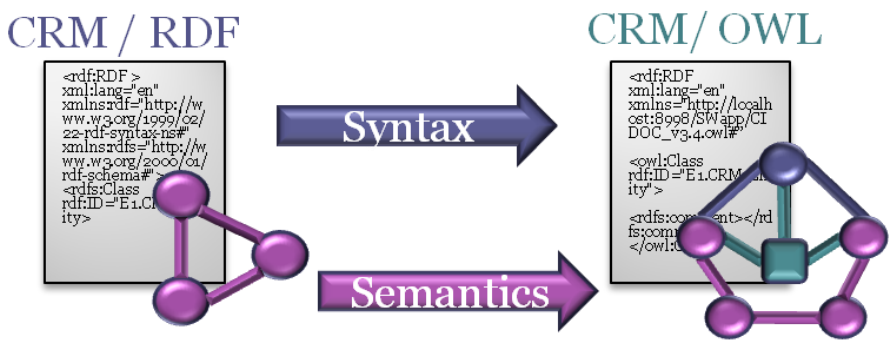


Fig. 2. The CIDOC-CRM transformation process.

3.1 Transforming Syntax

In order to transform the ontology to OWL syntax, we initially utilized the RACER system [5]. RACER has the ability to load and process ontologies expressed in various formats, including RDF(S) and OWL. One can instruct RACER to load TBoxes expressed in RDF(S) by using the `rdfs-read-tbox-file` command. Once loaded, the TBox can then be exported to the appropriate format by using the `save-tbox` command along with the `:syntax` parameter.

Following these steps, we actually received a formal OWL document representing correctly the initial ontology. However, we discovered that RACER included some unnecessary and redundant statements, which, in many cases, were semantically overlapping. For example:

- For every role and concept, RACER included tags from the OILed namespace; in particular, RACER added the tags `oiled:creationDate` and `oiled:Creator`, which were not required nor included in the initial document.

- For every concept defined as domain or range, RACER used the `owl:UnionOf` operand, thus expressing these restrictions as singleton concept unions (including only the concept in particular).
- The definition of role domains and ranges, even in OWL, comes from the RDF(S) namespace (`rdfs:domain`, `rdfs:range`). RACER, even though it maintains these statements, it duplicates them with equivalent expressions, which relate to the DL-like style of expressing this kind of restrictions. These equivalent statements involve number and value restrictions and can be represented in OWL.

This process resulted in transforming the initial 60KB file to a 478KB OWL document. We therefore opted for the manual transcription of the RDF(S) document, during which common expressions between RDF(S) and OWL were preserved (e.g. `rdfs:subClassOf` and `rdf:resource`), while we replaced some namespace prefixes and updated the terminology used (e.g. `owl:Class` instead of `rdfs:Class` and `owl:ObjectProperty` or `owl:DataTypeProperty` instead of `rdf:Property`). In this manner the CRM syntactical transformation phase was completed, resulting in a 63KB document, named cidoc_crm_v3.4.owl.

3.2 Semantic Augmentation and Profiling

The second phase of CRM upgrading process included its semantic augmentation with OWL-specific structures up to the OWL DL level, so as to enable a satisfactory level of reasoning, as well as its completion with some concrete instances.

This has been conducted in two steps: first, we added expressions that pertain to the model itself, so as to better capture intended meaning of properties and classes by taking advantage of OWL vocabulary. Second, added further subclasses and *semantic constraints* on them that actually profile the model for the specific case of paintings and painters in general. As an application scenario we have chosen to model facts from the life and work of the Dutch painter Piet Mondrian. Let’s examine these steps in detail:

Core Refinement

In this step, we do not add any new classes or entities that extend the CRM. Instead, we try to better approximate the core model’s conceptualization by using OWL statements that allow for its more precise implementation. In particular:

- We modeled minimum and maximum cardinality restrictions by using unqualified number restrictions (`owl:minCardinality`, `owl:maxCardinality`).
- We modeled inverse roles, using the `owl:inverseOf` operand.
- We included a symmetric role example, using the `rdf:type= “&owl;Symmetric”` statement.

To a certain extend, adding cardinality constraints to properties may be considered a profiling act, since the model clearly specifies that these quantifiers are provided only for semantic clarification. Nevertheless, by doing this we achieve the shift of intended meaning from inside text notes and annotations to a semantic commitment. Please also note that RDF(S) being CRM's favoured implementation, there is no way to express such constraints. For the purpose of our work, we have not exhaustively quantified the CRM properties, but applied constraints to some ones, used and instantiated in our Mondrian example.

Clearly, the additions above actually refine the core model, either if this intended in its specification or not. Profiling in this way therefore achieves to expand the *intentional knowledge* of the schema using constructs and means provided only in a Semantic Web infrastructure.

Profiling for the Application

During this step, we create some specific CRM concept and roles instances pertaining to our particular application. We also include axiom and fact declarations that only OWL allows to be expressed, as well as new roles and concepts making use of this expressiveness.

- We added the classes: "Painting" as subclass of CRM's "Visual_Item", "Painting_Event", a subclass of "Creation_Event" and "Painter" a subclass of "Person".
- We added a data type property "hasURL" as a sub-property of "has_current_location".
- We semantically characterized above concepts based on existential and universal quantification, by using the `owl:hasValue`, `owl:someValuesFrom` and `owl:allValuesFrom` expressions, which ultimately enable more complex inferences.

For example (see also Section 4):

```
<owl:Class rdf:ID="Painter">
 <rdfs:subClassOf rdf:resource="&crm;E21.Person"/>
 <owl:equivalentClass>
  <owl:Restriction>
     <owl:onProperty rdf:resource="&crm;P14B.performed"/>
     <owl:someValuesFrom rdf:resource="#Painting_Event"/>
    </owl:Restriction>
 </owl:equivalentClass>
</owl:Class>
```

This fragment defines a new class, namely "Painter" and states that a "Painter" is any "Person" that has "performed" at least one "Painting_Event". Then we can instantiate Mondrian as follows:

```
<crm:E21.Person rdf:ID="Mondrian">
 <crm:P14B.performed>
  <Painting_Event rdf:ID="Mondrian's Composition"/>
 </crm:P14B.performed>
</crm:E21.Person>
```

Given the "Painter" class definition it is straightforward for a Semantic Web reasoner to infer that "Mondrian" is indeed a painter.

This is another direction of semantic profiling: We added new elements bearing their own namespace, but then we semantically entangled them with each other and with the model's own definitions, thus imposing *semantic refinements* for our own specific case.

One of the main concerns when developing an application profile is to ensure the source schema is not affected and its general applicability maintained. To achieve this, in addition to namespaces, OWL provides an explicit inclusion mechanism through the `<owl:imports>` statement.Therefore, although these extensions could have been integrated in the initial document, we chose to include them in a new file, namely mondrian.owl. In this way we preserve the original model and we also show Semantic Web capabilities for ontology integration and distributed knowledge discovery.

Both documents were made available on the Internet through the Tomcat server. Inclusion of cidoc_crm_v3.4.owl axioms was possible simply by using the `<owl:imports>` directive in mondrian.owl. Therefore, loading mondrian.owl also loads all the axioms from cidoc_crm_v3.4.owl as well, as long as the latter is available on the Internet. In order to resolve potential ambiguities, different namespaces were defined for each document. In order to refer to statements from the imported ontology, the `crm` prefix is used, whereas for the new statements the default prefix (#) is used[1].

4 Results

In the following we present the results from some experimental inference actions conducted on the CRM augmented OWL form using our KDI. The KDI is a web application, providing intelligent query submission services on Web ontology documents and is detailed in [7]. The KDI uses a Description Logics inference engine (RACER) as a reasoning back-end. To actually conduct our inferences we follow the methodology for Semantic Web Knowledge Discovery documented in [8].

Inferences performed can be divided in two categories: *Positive* inferences where, based on the concept and role axioms as well as the ontology facts we conclude new, not explicitly expressed facts and *negative* inferences where, based on the ontology axioms and facts we detect unsatisfiability conditions on concepts and instances.

For every example we give the OWL fragment where the inference is based on, and we graphically depict the reasoning process in terms of DL formalism. To save space, instead of full namespaces we use the prefix `&crm;` for entities originating from the cidoc_crm_v3.4.owl document, as well as the default prefix "#" for entities coming from the mondrian.owl document (which includes the former).

Relationships and assertions that hold in these results are also depicted graphically according to the following legend: a box denotes an instance, a circle stands for a concept, a headed arrow between instances i_1, i_2 a relation $R<i_1,i_2>$, an arrow between concepts a subsumption relationship towards the direction of the arrow and an arrow between an instance and a concept denotes a membership ("isA") relationship.

[1] Both documents are available under http://ippocrates.hpclab.ceid.upatras.gr:8998/SWapp/

4.1 Positive Inferences

The following code is a fragment from mondrian.owl stating that a “Painting_Event” is in fact a “Creation_Event” that “has_created” “Painting” objects only:

```
<owl:Class rdf:ID="Painting_Event">
  <rdfs:subClassOf
  rdf:resource="&crm;E65.Creation_Event"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty
      rdf:resource="&crm;P94F.has_created"/>
      <owl:allValuesFrom rdf:resource="#Painting"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<Painting_Event rdf:ID="Creation of Mondrian's
composition">
  <crm:P94F.has_created rdf:resource="#Mondrian's
  composition"/>
</Painting_Event>
```

The above fragment is graphically depicted in the left part of Figure 3.

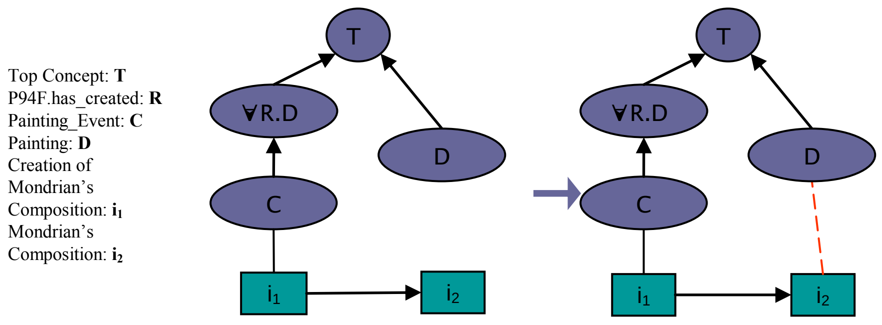


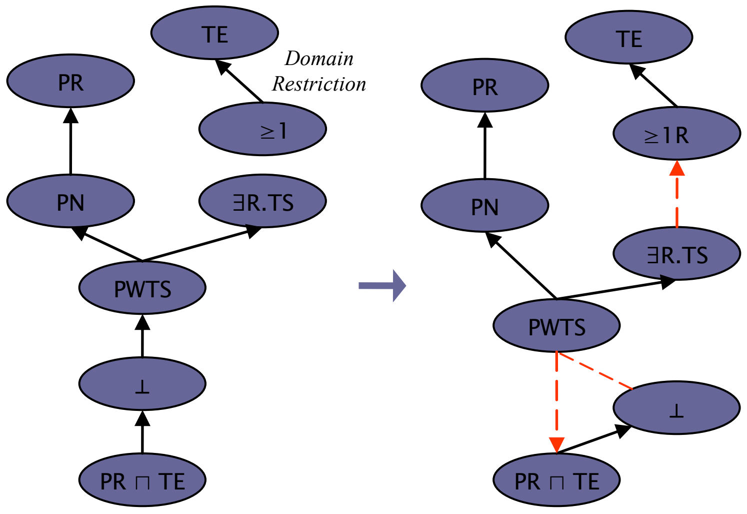
Fig. 3. Inference example using value restriction.

Creation of Mondrian’s Composition” (***i****1*) is an explicitly stated “Painting_Event” that “has_created” (***R***) “Mondrian’s composition” (***i****2*). Now, asking the KDI to infer “what is a painting?” it infers that i_2 is indeed a painting (right part of Figure 4), correctly interpreting the value restriction on role R.

As simple as it may seem, this is indeed a very powerful inference. Without the value restriction on role “has_created”, the “Mondrian’s composition” is just an instance of the world, i.e. it can be a book, a chair, a man or a time-period. It is just because of this restriction, *apparent only in OWL*, and thus made possible to express o*nly after our transformation* that “Mondrian’s composition” is discovered to be a “painting” and not anything else. In the next example however, “Mondrian’s composition is clearly stated to be a “painting” from the beginning.

4.2 Negative Inferences

In CRM, temporal events may have a time-span. Naturally, a “Person” cannot have a time-span, unless it is also a “Temporal Entity”. In the following we state that “Persons” and “Temporal Entities” are disjoint concepts and we attempt to define the class of “Painters with time-span”.



Bottom Concept: ⊥
P4F.has_time-span: **R**
E2. Temporal_Entity: **TE**
E21.Person: **PR**
E52.Time-Span: **TS**
Painter: **PN**
Painter_with_time-span: **PWTS**

Fig. 4. Detecting unsatisfiable concepts.

```
<owl:ObjectProperty rdf:ID="P4F.has_time-span">
  <rdfs:domain rdf:resource="#E2.Temporal_Entity"/>
</owl:ObjectProperty>
<owl:Class rdf:about="&crm;E2.Temporal_Entity">
  <owl:disjointWith rdf:resource="&crm;E21.Person"/>
</owl:Class>
<owl:Class rdf:about="#Painter">
  <rdfs:subClassOf rdf:resource="&crm;E21.Person"/>
</owl:Class>
<owl:Class rdf:ID="Painter_with_time-span">
  <rdfs:subClassOf rdf:resource="#Painter"/>
  <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty rdf:resource="&crm;P4F.has_time-
      span"/>
      <owl:someValuesFrom rdf:resource="&crm;E52.Time-
      Span"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

The above fragment is graphically depicted in Figure 4. A "Painter with time-span" is defined as a "Painter" (known subclass of "Person") that "has time-span" some "Time-Span" instances. However, individuals that "have time-span" are required to belong to the "Temporal Entity" class, as dictated by the corresponding domain restriction. Therefore, apart from being a "Person", a "Painter with time-span" must also be a "Temporal Entity". On the other hand "Persons" and "Temporal Entities" are disjoint, so their intersection represents the bottom (always empty) concept. Thus, a "Painter with Time-Span" can never exist, as its class is inferred to be equivalent to the bottom concept. The KDI correctly detects the unsatisfiability of this class by pointing it out with red colour in the taxonomy.

5 Conclusions and Future Work

In this paper we attempted to deploy a working platform upon which we experimented with the application of Semantic Web techniques and ideas on the cultural heritage domain. Concurrently, we suggested a practice that can easily be followed in any other domain of interest.

First we have showed the Semantic Web capabilities for knowledge discovery with web ontologies. We conducted and presented a series of successful experimental results possible only after aligning our ontological model to the Semantic Web standards. A side product of this process is the strengthening of the argument that OWL and its most expressive decidable subset, OWL DL, may be recommended for modeling domain metadata and be fruitful in that way.

At the same time, we have documented a procedure for knowledge acquisition which, having the CIDOC-CRM as a starting point, can be likewise applied in any other knowledge domain. To ensure feasibility and re-productivity we developed and utilized suitable technical means for this, namely the KDI, as a proof-of-concept.

Doing so, we elaborated a novel technique for creating metadata application profiles, by taking advantage of the Semantic Web toolbox. This technique, involves semantic enrichment of the metadata model and then deepening of its structures and definitions in accordance to specific needs.

A possible combination of semantic profiling with traditional metadata profiling practices like namespace inclusion and merging may be worth to examine as future work. The combination for example of a CRM profile with a flat metadata schema (e.g. Dublin Core) should allow for the interchangeable use of both their element sets, provided this is done in a semantically consistent and productive manner, i.e. simple metadata elements are not treated naively as annotations.

To this end, of particular interest is looking into the upcoming OWL 1.1[2] specification and especially its concept of *punning* as a meta-modeling principle, based on which a name definition may have variable semantic interpretation depending on the ontological context.

References

1. Crofts, N., Doerr, M. and Gill, T. The CIDOC Conceptual Reference Model: A standard for communicating cultural contents. Cultivate Interactive, 9 (2003) http://www.cultivate-int.org/issue9/chios/
2. Cuenca Grau, B., Horrocks, I., Parsia, B., Patel-Schneider, P. and Sattler, U. Next Steps for OWL. In Proc. of the OWL Experiences and Directions Workshop (OWLED'06) (2006)
3. Duval, E., Hodgins, W., Sutton, S., Weibel, S.L. Metadata Principles and Practicalities. D-Lib Magazine, 8 (2002) http://www.dlib.org/ dlib/april02/weibel/04weibel.html
4. Guarino, N. Formal Ontology and Information Systems. In N. Guarino (ed.): Formal Ontology in Information Systems. Proc. of FOIS'98. IOS Press (1998) 3-15
5. Haarslev, V., Möller, R.: Racer: A Core Inference Engine for the Semantic Web. In Proc. of the 2nd International Workshop on Evaluation of Ontology-based Tools (EON2003) 27-36
6. Heery R. and Patel, M. Mixing and Matching Metadata Schemas. Ariadne, 25 (2000) http://www.ariadne.ac.uk/issue25/app-profiles/intro.html
7. Koutsomitropoulos, D. A., Fragakis, M. F., Papatheodorou, T. S. Discovering Knowledge in Web Ontologies: A Methodology and Prototype Implementation. In Proc. of SEMANTICS 2006. OCG (2006) 151-164
8. Koutsomitropoulos, D. A., Fragakis, M. F., Papatheodorou, T. S. A Methodology for Conducting Knowledge Discovery on the Semantic Web. In S. Sirmakessis (ed.) Adaptive and Personalized Semantic Web. Studies In Computational Intelligence, 14. Springer (2006) 95-105
9. Powell, A. and Johnston P. Guidelines for implementing Dublin Core in XML. DCMI Recommendation (2003) http://dublincore.org/documents/dc-xml-guidelines/