AN IMPLEMENTATION OF THE CIDOC CONCEPTUAL REFERENCE MODEL (4.2.4) IN OWL-DL

Guenther Goerz; Bernhard Schiemann; Martin Oischinger
Department of Computer Sciences (8), University of Erlangen-Nuremberg,
Haberstrasse 2, D-91058 Erlangen.
Phone: +49 9131 852-8701, Fax: +49 9131 852-8986
eMail: goerz@informatik.uni-erlangen.de
URL: http://www8.informatik.uni-erlangen.de/inf8/en/goerz.html

ABSTRACT

CIDOC's CRM has been introduced as a formal reference ontology with a particular focus on cultural heritage documentation. Meanwhile acknowledged as an ISO standard (21127:2006), one of its main goals is to facilitate interoperability between data and database schemata. In order to take full advantage of CRM's benefits, an implementation in a modern knowledge representation language has been a desideratum for quite a while.

Suitable candidates for very expressive, albeit decidable knowledge representation languages are Description Logics. A dialect of the Web Ontology Language, OWL-DL, based on XML and RDF/RDFS(FA) within the Semantic Web language hierarchy, is equivalent to a very expressive description logic.

To satisfy the needs arising from practical applications, we have developed an implementation of the most recent CRM version in OWL-DL. We will discuss problems of formalization of the scope notes by a few typical examples, and, furthermore, point out some problems in the design of the CRM which may lead to future improvements.

As a practical example, a comprehensive XML database containing documentation of paintings and sculptures as supplied by the German National Museum in Nuremberg (GNM), has been turned into a domain ontology for which CRM/OWL serves as a foundational reference ontology.

1 CRM AS A REFERENCE ONTOLOGY

The CIDOC CRM has been developed as a reference ontology [10] with a particular focus on cultural heritage information and documentation. The document "Definition of the CIDOC Conceptual Reference Model" [4], whose most recent version (4.2.4) has been released in January 2008, which describes 87 classes in a class hierarchy and 148 properties and their inverses, is regarded as the authoritative reference. The semantics is described by scope notes in textual form, augmented by best practice hints and examples.

The origins of the CRM can be seen in attempts to achieve interoperability between various cultural heritage databases in the 1990s by defining mappings between their data schemata, which were not successful in the long run. There are many reasons for this failure, among which the most important ones were unresolved intriguing semantic problems and, of course, combinatorial growth of the number of mappings. Therefore, to define a generic reference ontology to which different data models could be mapped seemed to be a more appropriate solution to the data integration problem.

Taking the claim of a (formal) reference ontology seriously, the CRM is much more than just another data model, but offers instead the opportunity for a transition from a data model to semantics. I.e., the CRM is more than a labelled structure in the form of an associative network, but it can be (re-) constructed methodologically by means a well understood set of epistemological expression types within a (background) theory of validity and truth.

As for the methodological aspect, a few remarks may suffice (for details cf. [9]). Taking up a pragmatic and language-critical viewpoint, we will start with — in the understanding of the respective community — systematic, meaningful, and methodologically justified action in a certain domain of discourse. Communication is embedded in cooperation, and this is where we get to validation criteria. In a "top-down" fashion, we reach the semantic level by appropriate abstractions w.r.t. to particular features of the situational context. I.e., now we deal with validity criteria and truth conditions for propositions, with meaning and reference in general. But we have to be aware that the notion of a "valid inference" is grounded in the pragmatic level. The construction of well-formed (meaningful) expressions of a language — for knowledge representation in general, some kind of labeled structures — is defined on the syntactic level. For logical consequence, there is a strict correspondence between (semantic) entailment and syntactic derivation (in some logic calculus).

Coming back to the CRM, we have to be aware that the CRM document is primarily natural language text; it attempts to define terms as precisely as possible in its scope notes and by means of examples. As with any natural language text, there will always be a range of interpretation as opposed to a formal, mathematical specification with clear (and unambiguous) definitions and a well-defined semantics. As the ongoing update process shows — the publication of version 4.2.5 is in preparation —, there have been many improvements and clarifications, and there will be more in the future.

In our opinion, in the long run the only opportunity to resolve the problems due to underspecification and vagueness is to aim at a formalization in some logical language such that automatic inference techniques can be applied.

As a formal reference ontology, the CRM can serve for several purposes, e.g.:

- as a generic background ontology for application modelling;
- as a tool for interoperability and data integration either by preprocessing data bases (data transformation) or at access time (by inference);
- for processing complex queries which require inference;
- to check consistency and coherence of extensions to the CRM.

2 WHY AN IMPLEMENTATION IN A COMPUTATIONAL LOGIC (LANGUAGE) ?

A critical investigation of the scope notes in the CRM document shows that the description of intricate semantic problems in common language is not only error-prone, but also in danger of vagueness and a certain degree of ambiguity — despite all effort in the precision of argumentation. Therefore, a clarification via translation into a logic-based language can be achieved; it also offers an opportunity to uncover methodological problems. In fact, with a few exceptions which reach beyond the realm of standard logic, as e.g. defaults, most of the issues which cannot be covered in a formal way in general are related to semantic issues in the domains and applications to be modeled.

To enable a computational system to use the represented knowledge it must be equipped with some kind of an inference mechanism, i.e., we have to address the computational aspects of logic. Three levels are to be considered: Given a formal logical language with its syntax and semantics and the subsequent expressive power, first of all the reasoning problem has to be specified. Investigations of the reasoning problem are aimed at its decidability and computational complexity. Finally, a problem solving procedure, i.e. a specific implementation solving the reasoning problem, has to be provided, and the problems to be addressed at this level are its soundness and completeness and the practical aspects of complexity. Of course, an ideal computational logic should be expressive, and have decidable reasoning problems for which sound complete and efficient reasoning procedures are available. Unfortunately, as one might expect, the field of logic is not an ideal world at all. But even the question for decidability cannot be answered positively for full first-order logic.

After several decades of research in knowledge representation languages, so called *description logics* are generally regarded as the optimum compromise between the scylla of expressiveness and the charybdis of decidability and computational complexity.

They are attractive as languages for formal ontologies not only due to specific language constructs designed for this purpose, but also because their inference problems are decidable and very efficient sound and complete reasoners are available for them [2]. So, description logics (DLs) are structured fragments of classical first-order logic providing concepts (classes), arranged in subsumption hierarchies with inheritance, properties (roles) and individuals (instances). Knowledge representation is object-oriented and done at the predicate level; there are no variables in the formalism. For structured descriptions, a restricted set of epistemologically adequate language constructs is provided to express complex relational structures of objects. Of particular importance is the distinction between conceptual (terminological, "T-Box") knowledge and knowledge about individuals, i.e. concept instances (assertional, "A-Box"). For reasoning, automatic classification to determine the subsumption — i.e., universal (material) implication — lattice plays a central role.

Constructions of conceptual models are begun with a small set of primitive concepts and roles. Further concepts are defined by composite expressions which express sufficient and necessary conditions. Compositional operators are negation (complement), conjunction, disjunction, and value and existential restrictions for roles. One of the most recent description logics, the (Semantic) Web Ontology Language OWL-DL (cf. [1, 3, 8]), adds further language constructs such as qualified number restrictions, "General Inclusion Axioms", nominals (i.e. classes with a singleton extension), transitive and inverse properties, property hierarchies, and data types. Therefore OWL-DL is currently the optimal choice for an ontology language.

From a syntactic point of view, OWL-DL belongs to the family of XML languages (cf. http://xml.coverpages.org/xml.html, accessed 31 May 2008). For the Semantic Web, a hierarchy of standardized representation languages based on XML has been proposed: The basic layer is given by XML plus namespaces plus data types (XMLSchema). The second layer consists of RDF (Resource Description Framework), a language to express associative triples (subject-predicate-object) which can be combined to associative networks as directed labeled graphs. RDF offers as modelling primitives instance-of, subclass and properties with range, domain, and cardinality restrictions. This layer is enriched by RDFS (RDFSchema), which provides a limited modelling vocabulary and allows to organize it in a typed hierarchy with facilities for the definition of classes and subclasses, and of roles and role hierarchies, but there is no commitment to an inference mechanism. The (description) logic layer, built on top of that, is exactly the place where OWL-DL is located as an extension to RDFS(FA), a sub-language of RDFS with a first-order style semantics. Extensions to incorporate rules, e.g. SWRL, which further increase its expressive power, are currently under development.

Inference engines ("reasoners") for Description Logics can perform a variety of logical deductions. Given a set of concepts, properties, and individuals, a "knowledge base", they can check for

- *Concept satisfiability*, i.e., whether a newly defined concept consistent with the knowledge base such that the extended knowledge base has (still) a model, as

well as *satisfiability* of the knowledge base as a whole;

- *Subsumption*, i.e., compute the proper place for a newly defined concept in the concept hierarchy;
- *Proper instantiation* , i.e. whether a given individual belongs to the class it claims to belong to;
- *Realization*, i.e., compute the class a given individual belongs to and *retrieval*, i.e., determine the instances of a given class.

From a technical point of view all of these kinds of inferences can be reduced to satifiability which facilitates their practical implementation (cf. [5, 8]).

Because of the inherent verbosity of OWL graphical editors such as Protégé (http://protege.stanford.edu/, accessed 31 May 2008) have become very popular. They cannot only be used to define formal ontologies and to generate concept instances, but also to visualize concept hierarchies by menas of dynamic graphs. A particular convenient feature is that they can be coupled with a DL reasoner such as Racer (http://www.racer-systems.com/, accessed 31 May 2008) in a client-server architecture, where the reasoner can immediately execute consistency checks, perform automatic classification of new concepts and instances, and answer queries.

3 PRINCIPLES OF THE IMPLEMENTATION OF THE CRM STANDARD

The Erlangen OWL-DL implementation of the CIDOC CRM (version 4.2.4) has been designed as close as possible to the specifications in the CRM document. Therefore, the CRM document should serve as the primary reference for the implementation (cf. [4]; the most recent version is available via the CIDOC CRM website http://cidoc.ics.forth.gr/, accessed 31 May 2008).

The current version of CRM/OWL-DL has been implemented with the help of Protégé; the code together with a Protégé "project file" and a short document describing the specifics of the implementation can be retrieved from the CIDOC CRM website. The screenshot in fig. 1 gives a general overview of the CRM class hierarchy. In fig. 2 the class *E73 Information Object* has been selected and is shown with its associated properties in the "conditions" pane; alternatively, a property-centered view could be chosen as well. Finally, a graphical display of the concept hierarchy surrounding the selected class is given in fig. 3.

Using Protégé or a similar graphical editor does not only increase readability of OWL-DL code considerably, but also makes handling it a great deal simpler due to a variety of built-in operations. This becomes immediately clear by comparing the Protégé display

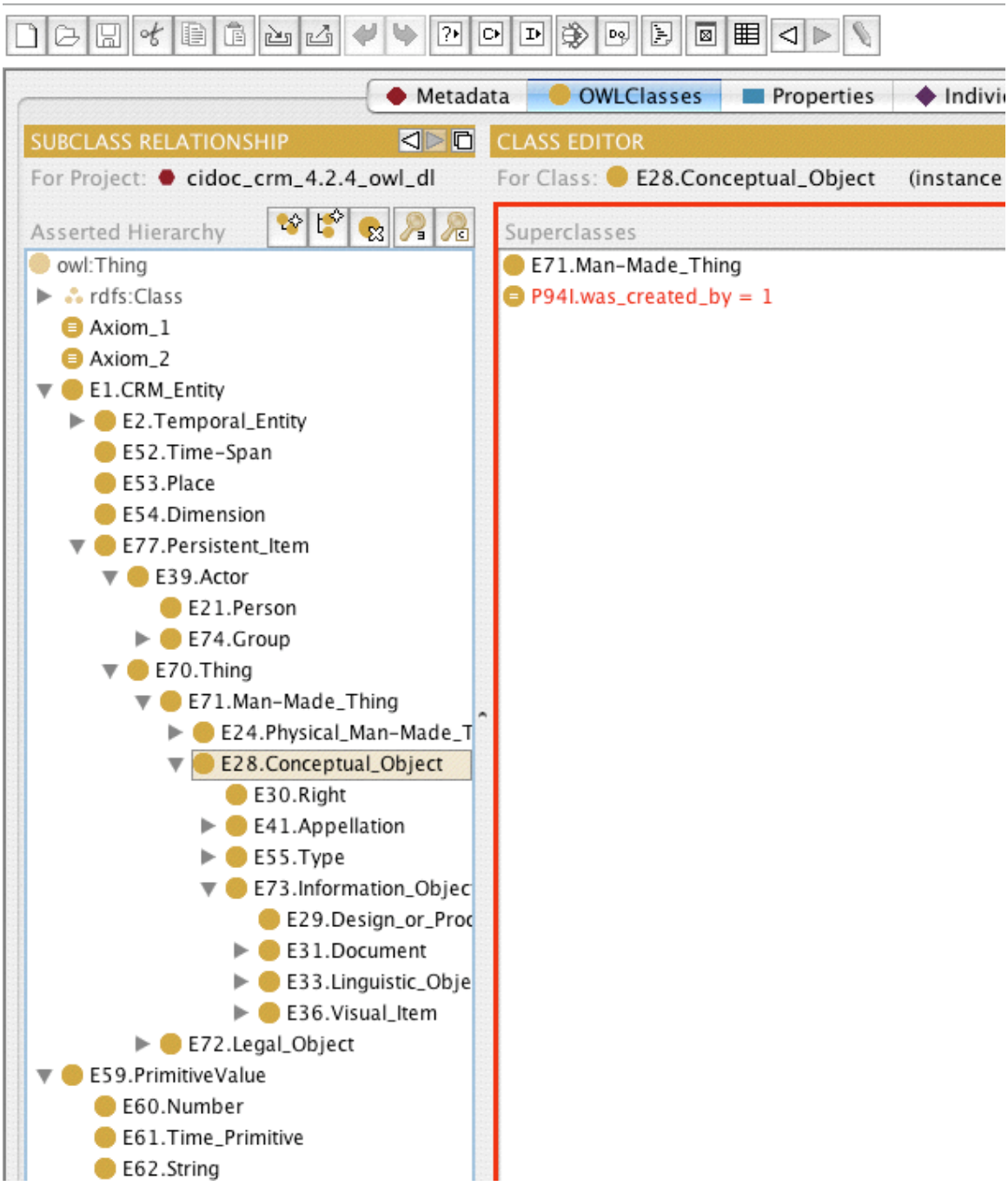

Figure 1: CRM 4.2.4 in OWL-DL: Partial view of the class hierarchy

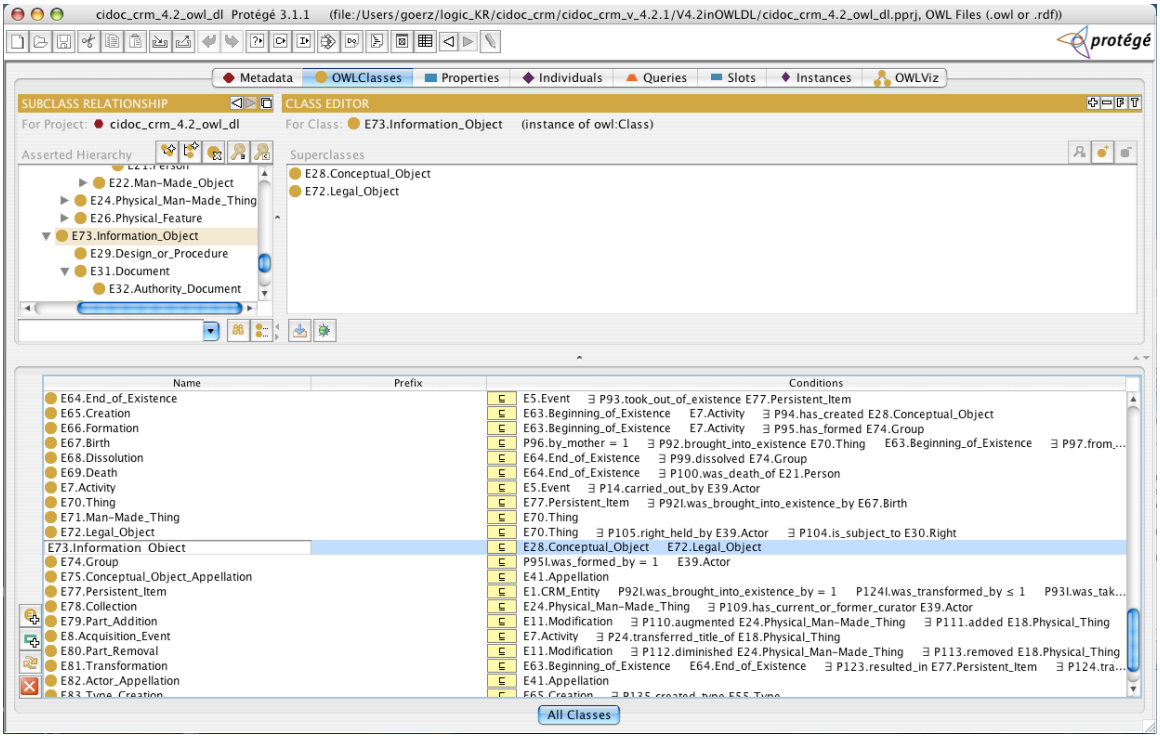

Figure 2: CRM 4.2.4 in OWL-DL: E73 Information Object and properties

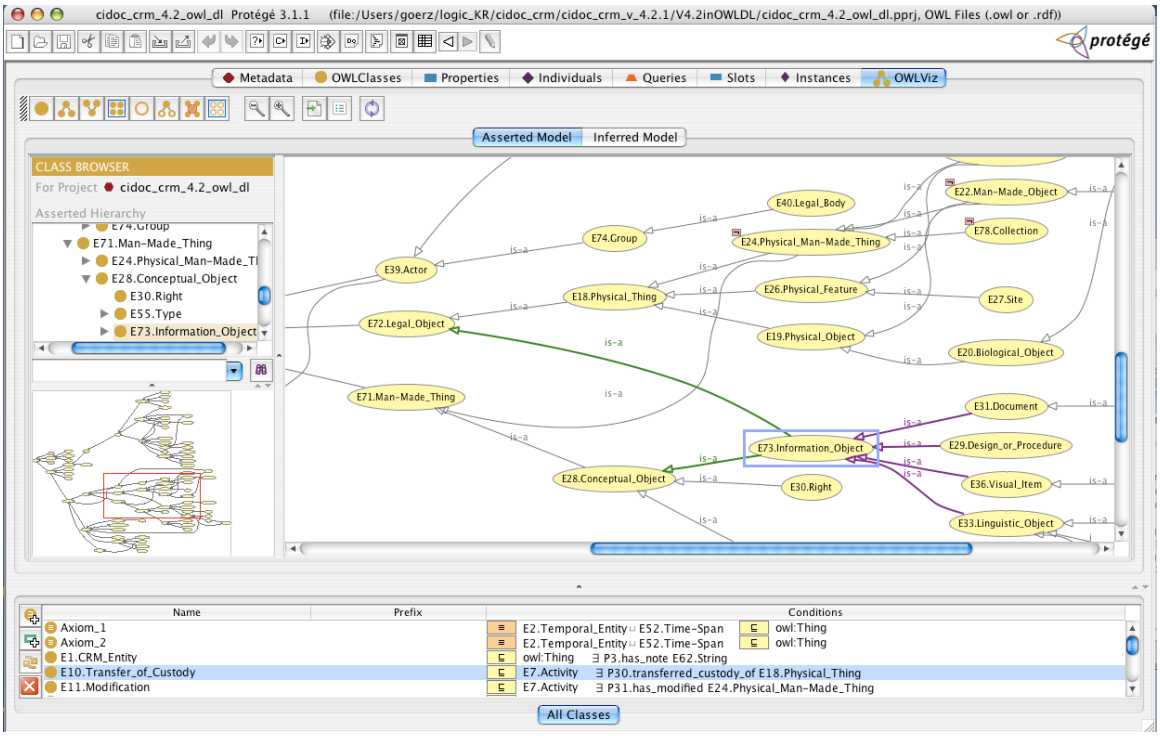

Figure 3: CRM 4.2.4 in OWL-DL: Partial graphical view of the class hierarchy

of *E1 CRM Entity* and *P3 has note* in fig. 4 with the following section of the source code:

```
<owl:Class rdf:about="#E1.CRM_Entity">
  <rdfs:subClassOf>
    <rdf:Description rdf:about="#Thing">
    </rdf:Description>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:someValuesFrom>
        <owl:Class rdf:ID="E62.String"/>
      </owl:someValuesFrom>
      <owl:onProperty>
        <owl:InverseFunctionalProperty rdf:ID="P3.has_note"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:InverseFunctionalProperty rdf:about="#P3.has_note">
  <rdf:type rdf:resource="#ObjectProperty"/>
  <rdfs:domain rdf:resource="#E1.CRM_Entity"/>
  <rdfs:range rdf:resource="#E62.String"/>
</owl:InverseFunctionalProperty>
```

The property-centered view is illustrated with the example of the properties of *E77 Persistent Item* in fig. 5. Inverse properties always carry an "I" at the end of their identifier.

Whatever is underspecified or unspecified in the CRM document has been left open in the OWL-DL implementation as well. E.g., the scope note of *P48 has preferred identifier (is preferred identifier of)* says: "Use of this property requires an external mechanism for assigning temporal validity to the respective CRM instance." What this external mechanism would be remains unspecified; therefore, the OWL-DL implementation leaves the matter open, too. Nevertheless, there are some features which could not be implemented or have not been implemented for certain reasons.

Two characteristic examples — out of 14 — of differences in the implementation are:

E60 Number: "…Identifiers in continua may be combined with numbers expressing distances to yield new identifiers, e.g., 1924-01-31 + 2 days = 1924-02-02." The implementation of *E60 Number* defines the concept by means of integer and float numbers. Therefore, arithmetic expressions cannot be instances of *E60 Number*; arithmetic expressions cannot be evaluated and it is not possible in a logical language without

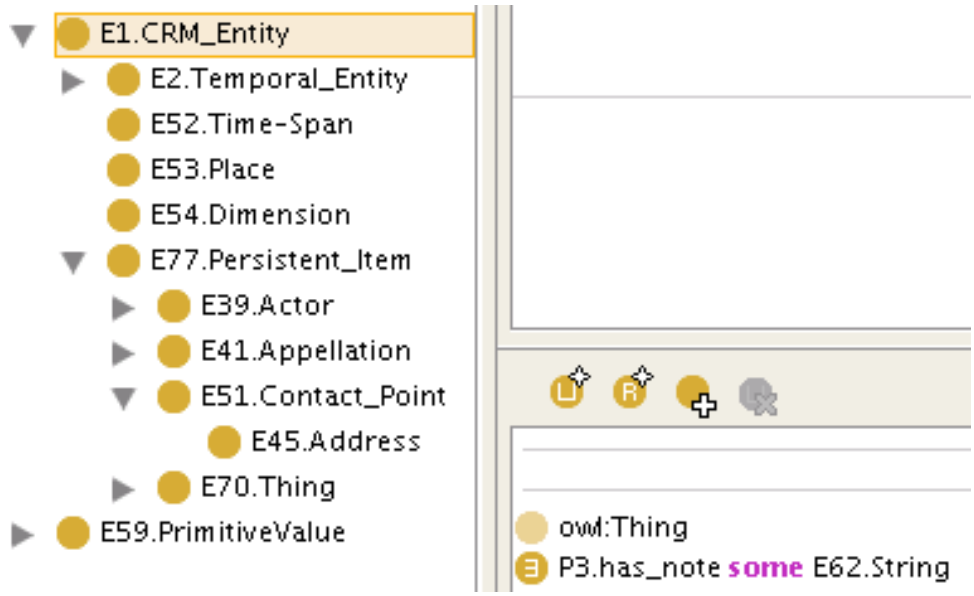

Figure 4: CRM 4.2.4 in OWL-DL: E1 CRM Entity

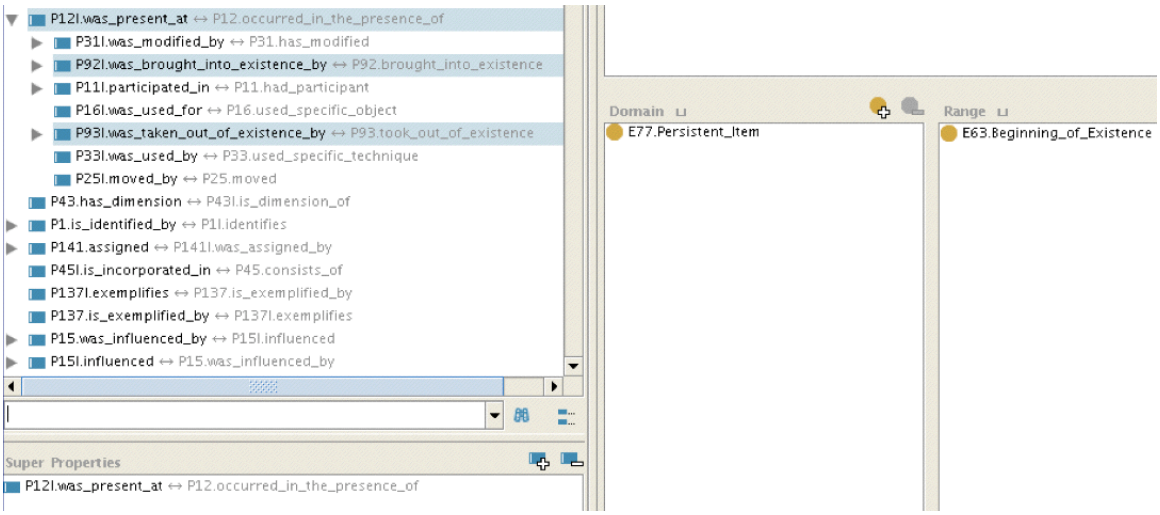

Figure 5: CRM 4.2.4 in OWL-DL: Properties of E77 Persistent Item

equality to express "3+1 = 4". If there is a need to deal with arithmetic expressions, a different representation must be used, e.g. *E73 Information Object*.

P10 falls within (contains): "...The difference with *P9 consists of (forms part of)* is subtle. Unlike *P9 consists of (forms part of)*, *P10 falls within (contains)* does not imply any logical connection between the two periods and it may refer to a period of a completely different type." This property has been implemented as a regular OWL-DL object property which expresses a logical relation between two *E4 Period*s.

Many scope notes contain comments related to the application of the CRM, in many cases by giving recommendations or referring to best practice in documentation. Comments pertaining to the use of the CRM do not affect its implementation directly, but could give reason for (optional) constraints in future versions.

A particular problem comes with the so called "types" in CRM. The use of the term "type" in CRM is explained in the introduction of the CRM document (p. xi); readers should be aware that the term "type" has a different meaning in computer science. In the CRM document (ver. 4.2.4), the class *E55 Type* is described as a metaclass. For the sake of decidability, OWL-DL does not provide means to represent metaclasses; metaclasses are higher-order logic constructs — and probably hard to comprehend for practitioners anyway.

Therefore, *E55 Type* has been implemented as a class which — for the purpose of reasoning on the conceptual level — may serve as an interface to external concepts of formal domain ontologies (or thesauri) as subclasses or as constants. In fact, at least two different representations are possible:

- The usual way to attach concepts of a domain ontology to the CRM is direct subclassing, e.g., the (application domain) class *Artist* as a subclass of *E21 Person*. So, "Vincent van Gogh" would be an instance of *Artist* and inherit all properties of *E21 Person*. In that case to represent *Artist* also as a *subclass* of *E55 Type* would lead to contradictions.
- Instead, a constant "Artist" may be used; in general, it will be a term of a domain-specific thesaurus. Such constants ("individuals") are admitted in T-Boxes by means of the "one-of" OWL-DL language construct, i.e. an enumeration datatype. They correspond to classes with singleton extensions. So, we could represent "Vincent van Gogh" as an immediate instance of *E21 Person* and relate it by *P2 has type* to *E55 Type* with value "Artist". In this case, of course, the constants cannot have instances in turn.

Both representations are not mutually exclusive; in our example the **name** of the class *Artist* (case 1) could additionally be used as a constant which is assigned as a value to *E55 Type* (case 2), but then it is up to the user to guarantee for semantic integrity. In the second case the intention expressed in the CRM document is supported that is shall be possible to deal with domain concepts — such as *Artist* — as objects of discourse.

Which of these representations will be chosen for a particular application will depend on the intended use of the domain model.

Because of this difference to the CRM document, all special properties with ".1" in its name have not yet been implemented; e.g., *P14.1 in the role of: E55 Type* connects an *E7 Activity* with an *E55 Type* to express a more detailed description of some other property (*P14 carried out by*). Methodologically, *P14.1* is **not** a subproperty of *P14*, but it describes the role of an *E21 Person* (*P14*-)related to an *E7 Activity* in more detail, thus providing a further explanation. In our example: The *E7 Activity* "Painting" *P14 carried out by: E21 Person* "Vincent van Gogh" is further described by *P14.1 in the role of: E55 Type* "Artist". So, *P14.1* addresses the "compound" *E21 Person + E7 Activity*, or, more precisely, *E21 Person* factorized by *E7 Activity* — an operation similar to "Currying" in functional and logic programming. One possibility to model such a factorized class would be to introduce a new subclass, in our example a subclass *Person/Activity* of *E21 Person*. Of course, an analogous subclass *Activity/Person* of *E7 Activity* could be introduced as well, if required. But this issue is still to be discussed and may be included in a future version of CRM/OWL-DL.

In general, all "short cuts" described in the CRM document have not (yet) been included because there is no unique meaning of these abbreviations. E.g., the scope note of *P8* says: *"P8 took place on or within (witnessed)* is a short-cut of a path defining a E53 Place with respect to the geometry of an object. Cf. E46 Section Definition." Obviously, the property *P8 took place on or within* is used as a placeholder for the connection between an *E4 Period* and the description of a geometry for an *E53 Place*. Unfortunately, there is no definition of a geometry at *E53 Place* such that it remains unclear how it should be implemented. Another kind is described in the scope note of *E36 Visual Item*: "The property *P62 depicts (is depicted by)* between E24 Physical Man-Made Thing and depicted subjects (E1 CRM Entity) can be regarded as a short-cut of the more fully developed path from E24 Physical Man-Made Thing through *P65 shows visual item (is shown by)*, E36 Visual Item, *P138 represents (has representation)* to E1 CRM Entity, which in addition captures the optical features of the depiction." This note describes a sequence of properties between three different CRM classes: the representing media (e.g. photographic paper), the visual item (e.g. a photograph itself) and the concept which represents the depicted object. If only the property *P62 depicts* would be used, it would have mutable semantics which could not be determined from the property itself but from the connected concepts. In favor of clear semantics the whole path should be composed of different properties. Both examples show that CRM short cuts describe quite different abbreviations and it remains to be clarified how they should be modelled. Furthermore, it still to be discussed whether — and which — short cuts should be inferred automatically or be introduced manually.

4 LIMITATIONS

At very few places the CRM document stipulates representation language features which go beyond the expressiveness of first-order logic. In such cases, an immediate respresentation in OWL-DL is not possible, but there may be work-arounds.

The following example, pertaining to *E11 Modification*, can only be represented adequately in the framework of a temporal logic: *P31 has modified (was modified by)* "If a modification is applied to a non-man-made object, it is regarded as an E22 Man-Made Object from that time onwards." This property describes a transition between two class affiliations of an instance in time. Such a transition — an instance first being an A and then becoming a B — cannot be represented in OWL-DL.

Another kind of limitations is related to data types, where data type properties are defined to be inverse functional. For the reason of decidability, this is not allowed in OWL-DL; instead the following "container" representation provides a solution. To stay within OWL-DL, it is required that entities point to *E59 Primitive Value* or its suclasses via inverse functional object properties, subclasses of *E59 Primitive Value* have data type properties which in turn point to XSD data types of XML Schema.

Finally, OWL-DL cannot express the definition of new properties (relations) in the way familiar from logic programming. As an example, consider the following definition of the `has-uncle` relation (capital letters represent variables):

$$\texttt{has-uncle(X, Z)} \Leftarrow \texttt{has-parent(X, Y)} \wedge \texttt{has-brother(Y, Z)}$$

For this purpose, SWRL, a rule extension language to OWL, has been proposed [7], by which OWL-DL is augmented with a limited form of logic programming (Datalog rules).

5 EXTENSIONS: AN APPLICATION

As a practical example, a comprehensive XML database containing documentation of sculptures as supplied by the German National Museum in Nuremberg (GNM; http://www.gnm.de/, accessed 31 May 2008), has been turned into a domain ontology for which CRM/OWL-DL serves as a foundational reference ontology.

GNM's document management system (DMS) is built upon a relational database system. Among the documents it holds are two large functional groups, the inventories of paintings up to 1800 and of sculptures up to 1800. Both inventories are available in XML format with associated DTDs for object and administrative metadata. The DTDs have been taken as the starting point for the construction of the GNM domain ontology, which in turn has been linked to the CRM as its reference ontology. Thus, the GNM-DMS ontology has been implemented as an extension to the CRM, where

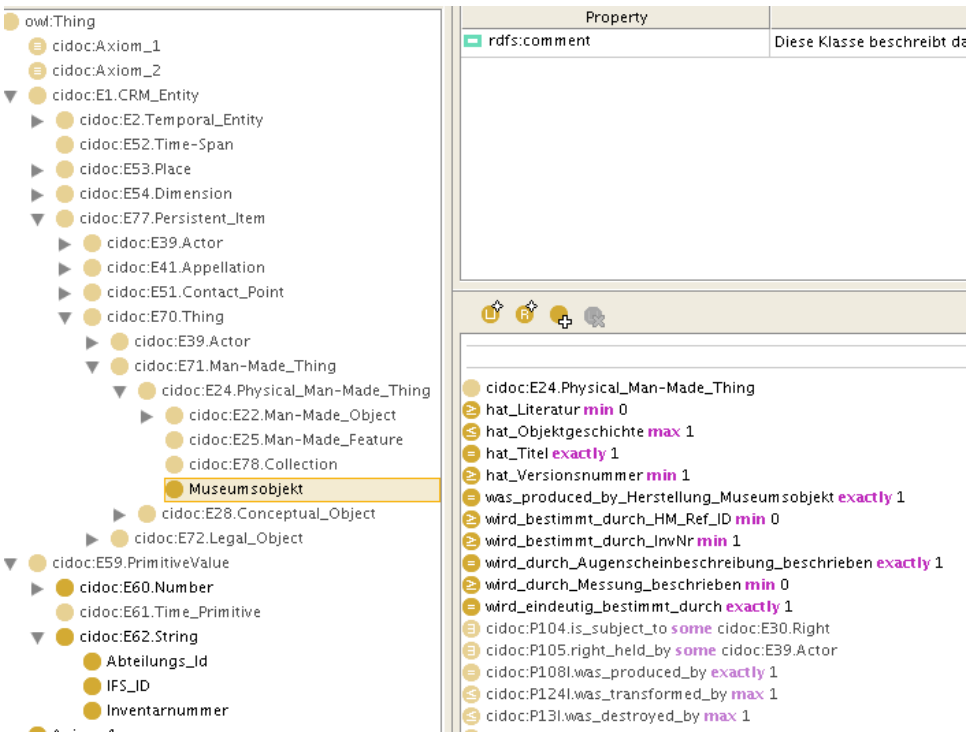

Figure 6: The GNM domain ontology as an extension to the CRM 4.2.4 in OWL-DL

the fundamental “interface class” is **Museumsobjekt** (cf. fig. 6 with domain specific properties — in German).

Particular descriptions of paintings and sculptures can be directly inspected in Protégé in the “individuals” (instances) pane. Of course, Protégé could also be used for the input of new object descriptions (instances). Despite a few technical problems with Protégé’s ontology import function the construction of the domain ontology was quite straightforward and could be finished successfully — including data transformation — within a few weeks. But this is just the first step, because the DMS data have to be augmented by deep indexing in terms of CRM’s event orientation as well as linking to various authority files for proper names, place names, etc. This holds in particular for all free text fields which are still represented as strings by now.

6 CONCLUSION

We have successfully achieved an implementation of the CRM in OWL-DL which covers the given specification as far as it can be formalized in logical terms, and given evidence of its usefulness in a practical application. Further improvements and extensions, e.g. towards a meta-CRM to be able to deal with generics and simple cases of default reasoning, are already taken up (cf. [6]).

ACKNOWLEDGEMENT

The authors are grateful to Siegfried Krause and Georg Hohmann of the GNM, Karl-Heinz Lampe of the ZFMK Bonn, and CIDOC's CRM SIG group for help and valuable discussions.

References

[1] Antoniou, G.; van Harmelen, F.: *A Semantic Web Primer*, The MIT Press, Cambridge, MA., 2004.

[2] Baader, F.; Calvanese, D.; McGuinness, D. (Hrsgb.): *The Description Logic Handbook : Theory, Implementation, and Applications*, Cambridge Univ. Press, Cambridge etc., 1. publ.. Ausg., 2003.

[3] Bechhofer, S. et al.: *OWL Web Ontology Language Reference*, W3C (World Wide Web Consortium), Geneva, February 2004, W3C Recommendation 10 February 2004.

[4] Crofts, N.; Doerr, M.; Gill, T.; Stephen, S.; Stiff, M.: *Definition of the CIDOC Conceptual Reference Model. Version 4.2*, The International Committee for Documentation of the International Council of Museums (ICOM-CIDOC), Paris, June 2005.

[5] Donini, F.; Lenzerini, M.; Nardi, D.; Schaerf, A.: *Reasoning in Description Logics*, in Brewka, G. (Hrsgb.): *Foundations of Knowledge Representation*, CSLI Publications, Stanford, CA, 1996, S. 191–236.

[6] Görz, G.: *"Generics and Defaults". Zum technischen Umgang mit Begriffssystemen, Standardannahmen und Ausnahmen*, in *Methodisches Denken im Kontext*, mentis, Paderborn, 2007, S. 383–401, Festschrift für Christian Thiel zum 70. Geburtstag.

[7] Horrocks, I. et al.: *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*, W3C (World Wide Web Consortium), Geneva, May 2004, W3C Member Submission 21 May 2004.

[8] Horrocks, I.; Sattler, U.: *A Tableaux Decision Procedure for SHOIQ*, in *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence IJCAI-05, Edinburgh, UK*, Morgan Kaufmann, San Francisco, 2005, S. 448–453.

[9] Kamlah, W.; Lorenzen, P.: *Logical Propaedeutic – Pre-School of Reasonable Discourse*, University Press of America, Lanham, MD, 1984.

[10] Menzel, C.: *Ontology Theory*, in Euzenat, J.; Gomez-Perez, A.; Nicola, G.; Stuckenschmidt, H. (Hrsgb.): *Ontologies and Semantic Interoperability, Proc. ECAI-02 Workshop*, Bd. 64 von *CEUR-WS*, ECCAI, Lyon, 2002, S. 7.