A Revised Data Model for the ISO Data Category Registry

MARC KEMPS-SNIJDERS, MENZO WINDHOUWER,
PETER WITTENBURG, SUE ELLEN WRIGHT

Introduction

ISO Technical Committee 37, Terminology and other language and content resources, is developing a revised version of its Data Category Registry (DCR)**.** Data categories are defined as ”the result of the specification of a gived data field”, and in TC 37 practice they comprise field names, definitions, and constraints, including the enumeration of permissible values for strictly constrained data categories. The DCR has demonstrated its value as a proof of concept and has been populated with numerous data categories (DCs), but users are clamoring for an upgrade with improved interface features and fully developed functionalities. The current implementation of the DCR is called Syntax (Ide and Romary, 2004). The new revised DCR has been christened ISOcat (Kemps-Snijders, Windhouwer et al., 2008). While designing ISOcat the opportunity was taken to address several issues with the current DCR data model.

The DCIF Model

The current ISO 12620 draft (ISO DIS 12620, 2007) describes a Data Category Interchange Format (DCIF) model which is based on the metamodel of ISO/IEC 11179-3 and is conformant with the terminological

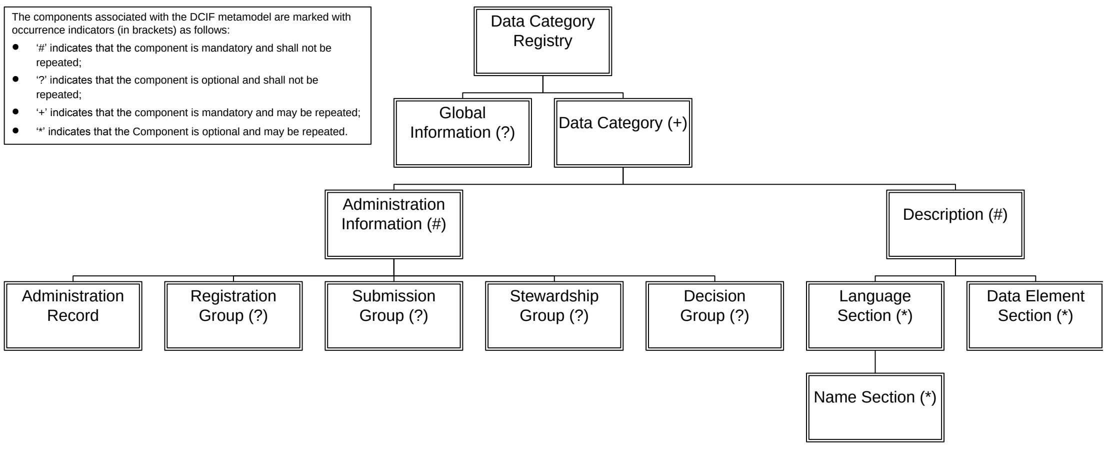

The components associated with the DCIF metamodel are marked with occurrence indicators (in brackets) as follows:
• '#' indicates that the component is mandatory and shall not be repeated;
• '?' indicates that the component is optional and shall not be repeated;
• '+' indicates that the component is mandatory and may be repeated;
• '*' indicates that the Component is optional and may be repeated.
Data Category Registry
Global Information (?)
Data Category (+)
Administration Information (#)
Description (#)
Administration Record
Registration Group (?)
Submission Group (?)
Stewardship Group (?)
Decision Group (?)
Language Section (*)
Data Element Section (*)
Name Section (*)

Figure 1 The DCIF model

metamodel described in ISO 16642 (ISO 16642, 2003) (see Figure 1), with minor modifications to accomodate for differences between data category specifications and terminological entries. In the current 12620 draft DCIF is further expressed in GMT as an interchange format in line with ISO 16642 (ISO 16642, 2003) methodology. The draft further describes all the attributes associated with the various components. There is no separate description of any data model other than the interchange format, and so the DCIF model thus also functions as the core data model of the DCR.

A Revised Data Model

Experience with the Syntax implementation of this model, ongoing work in TC37 and design of the ISOcat implementation revealed several difficulties with the current model, leading to revision of the data model. Instead of the hierarchical component structure of the DCIF model, it was decided to use Unified Modeling Language (UML) (Object Management Group, 2004) class diagrams to represent the data model. These diagrams are semantically richer and able to express more of the constraints, certainly in combination with the Object Constraint Language (OCL) (Object Management Group, 2006), which are currently only expressed in prose. In the Appendix the full class diagram is shown, throughout the paper smaller parts of the diagram are used for illustrational purposes.

It should be noted that this data model is not the same as the definition of an interchange format. Optimally ISO 12620 will describe a semantically rich data model *and* an interchange format, i.e. a DCIF, which provides a standard for serialization of instances of that data model. The following sections will focus on difficulties and propose solutions.

Distinction between Simple and Complex Data Categories

ISO 12620 describes two basic types of DCs: complex and simple. Only complex DCs have a conceptual domain, which can be open or closed, i.e. a finite list of values represented by simple DCs. The DCIF model does not provide an explicit distinction between simple and complex DCs. For closed (complex) DCs this is not a direct problem, because the conceptual domain will contain a list of admissible simple DCs. The presence of this list indicates a closed DC. In cases where there is an open conceptual domain, i.e. there are no restrictions on the admissible values, or when a closed conceptual domain is non-enumerated, the differentiation with respect to a simple DC is lost. In all of these cases, the conceptual domain will be empty, thus making distinction between them impossible. It could be argued that when a conceptual domain has not been specified at all in the DCIF, it should indicate a simple DC. This leaves the problem of distinguishing between closed and open complex DCs, where it could be agreed that a closed DC with a non-enumerated conceptual domain should not be allowed. However, in the currently active implementation these details have not been implemented, thus making the distinction impossible.

As an alternative we have chosen to model the distinction between simple and complex DCs explicitly. As Figure 2 shows, the object oriented nature of UML allows us to create a base class, Data Category, with two subclasses, one for the Complex and the other for Simple DC types. The association between the Conceptual Domain and the Complex DC classes now makes explicit that simple DCs do not have a conceptual domain. Contrary to Complex DCs, Simple DCs can be part of a value hierarchy by participating in an ’is a’ association. This allows one, for example, to declare a */pronoun/* to be a */noun/*. This type of association is not permitted between Complex DCs since these would essentially describe concept hierarchies which fall outside the scope of the DCR.

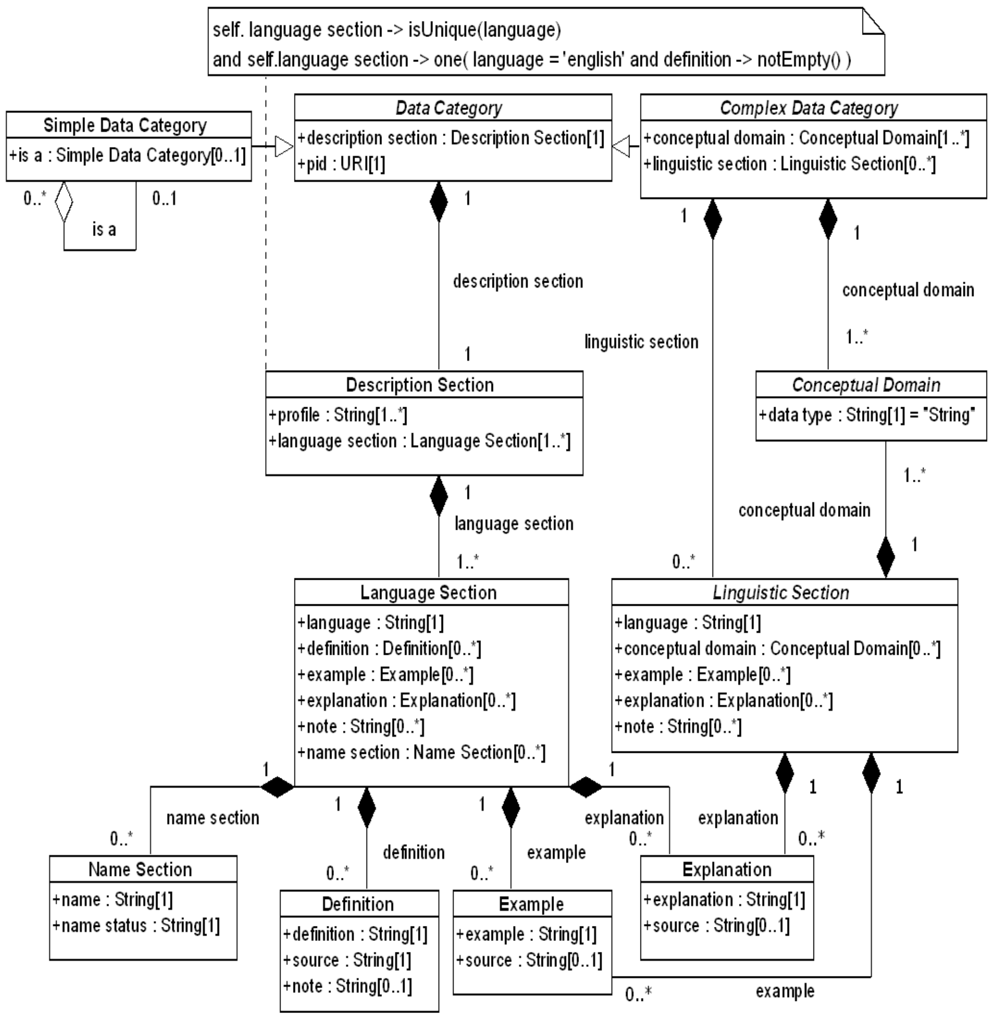


Figure 2 Classes related to DC types, along with object language and working language

Object and Working Languages

“The Language Section describes the data category within the context of a given object language. If need be, information can be provided on the applicability of the concept in a given language by associating it with a specific conceptual domain (subsetting the set for the Descriptive component).” “The Name Section is part of the Language Section and shall be used to record a possible name for the data category in a specific language.” (ISO DIS 12620, 2007) The Language Section thus provides information on the data category in the context of a specific language. That language can play the role of an object language or a working language. The mix of fields for these two distinct roles leads to confusion. To make a proper distinction between object language and working language in a concrete use case requires some careful maneuvering through the current DCR. Assume a German user who wants to annotate French texts. To retrieve the necessary information for ‘gender’, first the language section for German needs to be retrieved for displaying */gender/* as *‘Genus’*. Next, the French language section needs to be retrieved for extracting the conceptual domain for French, i.e. */masculine/* and */feminine/* being the subset for French of the full Value Domain, /*masculine*/, /*feminine*/ and /*neuter*/. Finally, for */masculine/* and */feminine/* the German translations need to be retrieved for display to the user. This means accessing the German language sections for /*masculine*/ and /*feminine*/ to yield ‘*männlich*’ and ‘*weiblich*’ as their respective German equivalents. As can be seen, the language in the language section is used multiple times in either the context of an object language or a working language.

Our solution for this confusion starts by dividing the fields related to object and working language by creating two classes: Language Section and Linguistic Section (see also Figure 2). The Language Section and its cluster of related classes contains the descriptive information for a given working language. The new Linguistic Section class subsets the conceptual domain for a given object language. As Simple DCs do not have conceptual domains, the Linguistic Section is only associated with Complex DCs.

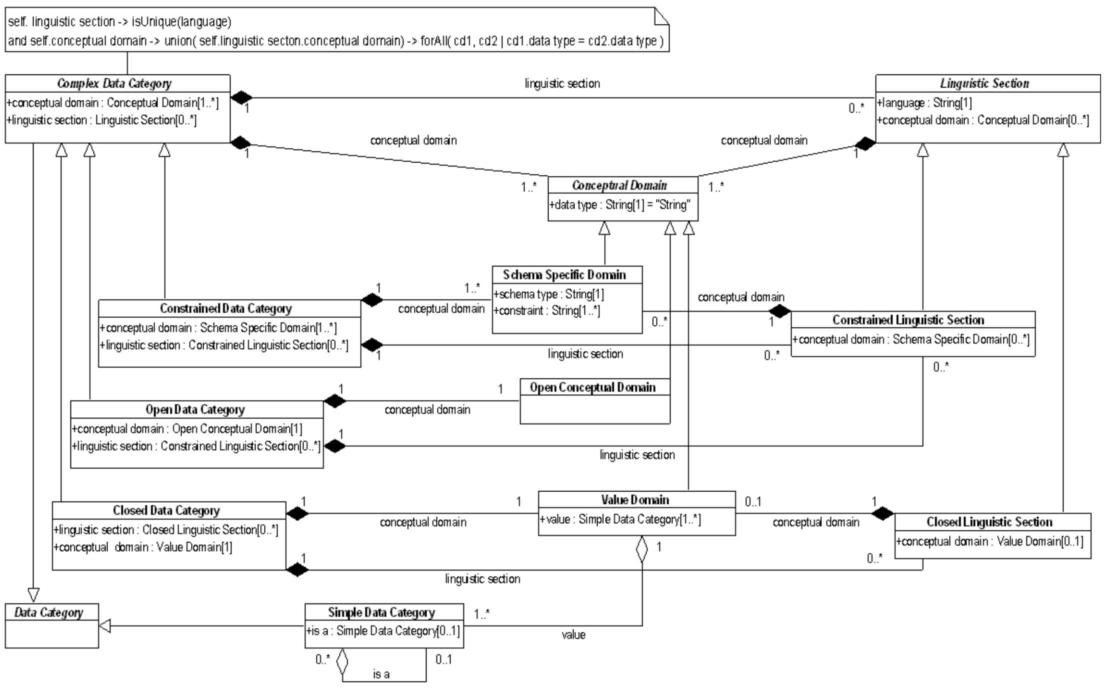
Duplication of the English Description

The aim of the Description DCIF component is to “provide descriptive information applicable to the data category” (ISO DIS 12620, 2007). The working language of this component is English, as one of the restrictions for submitting a new data category for standardization is that there should be at least one English */definition/* in the Description component. Translations in other languages of the same descriptive information can be provided in one or more Language Sections. However, for DCR clients (human or machine) this means that there are two places to look for the same type of information: for English look in the Description component; for all other languages look for a specific Language Section. English has become an exception, which will need special handling by clients. A solution to make sure that all information can be found in one place is to duplicate the English information from the Description component in an English Language Section. If this has to be done manually it will be error prone, i.e. a likely source of data inconsistencies.

In Figure 2 shows the solution in the revised data model is shown. The Description Section class contains a set of Language Sections. An additional semantic constraint, expressed in OCL, states that in this set there has to be one Language Section for the English language which also has a non-empty definition. All descriptive information is now stored in one place without need for duplication.

Types of Conceptual Domains

ISO 12620 currently defines two types of DCs with a conceptual domain: open and closed DCs. An open DC is defined as “complex data category whose conceptual domain is not restricted to a set of values”. An additional note states: “It is always possible to restrict an open data category to a specific data type (e.g. String, Integer, Date, etc.).” However, the DCIF model does not contain any mechanisms for specifying these data type constraints. A closed DC is defined as “complex data category whose



self. linguistic section -> isUnique(language)
and self.conceptual domain -> union(self.linguistic secton.conceptual domain) -> forAll(cd1, cd2 | cd1.data type = cd2.data type)
Complex Data Category
+conceptual domain : Conceptual Domain[1..*]
+linguistic section : Linguistic Section[0..*]
linguistic section
1
0..*
Linguistic Section
+language : String[1]
+conceptual domain : Conceptual Domain[0..*]
conceptual domain
1
conceptual domain
1
Conceptual Domain
+data type : String[1] = "String"
1..*
1..*
Schema Specific Domain
+schema type : String[1]
+constraint : String[1..*]
1
1..*
conceptual domain
Constrained Data Category
+conceptual domain : Schema Specific Domain[1..*]
+linguistic section : Constrained Linguistic Section[0..*]
conceptual domain
0..*
1
Constrained Linguistic Section
+conceptual domain : Schema Specific Domain[0..*]
linguistic section
1
0..*
0..*
Open Conceptual Domain
1
1
conceptual domain
Open Data Category
+conceptual domain : Open Conceptual Domain[1]
+linguistic section : Constrained Linguistic Section[0..*]
1
linguistic section
Closed Data Category
+linguistic section : Closed Linguistic Section[0..*]
+conceptual domain : Value Domain[1]
1
conceptual domain
1
Value Domain
+value : Simple Data Category[1..*]
0..1
conceptual domain
1
Closed Linguistic Section
+conceptual domain : Value Domain[0..1]
1
1
0..*
linguistic section
Data Category
Simple Data Category
+is a : Simple Data Category[0..1]
1..*
value
0..*
0..1
is a

Figure 3 Classes related to conceptual domains

conceptual domain is restricted to a set of identified simple data categories making up its value domain" (ISO DIS 12620, 2007).

Next to these two basic types, current users of the DCR indicate that they would also like to add arbitrary constraints to DCs, e.g. 'only dates later then the January 1st 2008 are allowed'. A complication for these constraints is the language in which they are expressed, e.g. W3C XML Schema, Semantic Web Rule Language (SWRL) or OCL (Fallside and Walmsley, 2004; Horrocks, Patel-Schneider et al., 2004; Object Management Group, 2006). The DCR should not prescribe a certain constraint language, but instead allow them all.

Once more we use subclassing to create classes for these three types of complex DCs: Constrained DC, Open DC and Closed DC (see Figure 3). Each of them is associated with a specific subclass of Conceptual Domain. Conceptual Domain itself contains the attribute type which indicates the data type of the DC, where the default type is String. A Schema Specific Domain can now add multiple constraints to this data type. To return to the previous constraint example: the DC data type could be Date, the schema type Schematron (ISO 19757, 2006) and the actual constraint:

```
<sch:assert test="self::*[. ge xs:date('2008-1-1Z')]">
  Only allow dates greater or equal then 2008-1-1.
</sch:assert>
```

Notice that the Constrained DC class allows links to multiple Schema Specific Domains. Each of these Schema Specific Domains would contain the same constraint but expressed in different constraint languages. Validation that the different Schema Specific Domains indeed express the same constraint would be a task performed external to the DCR, otherwise the DCR implementation would need to be able to interpret all possible constraint languages, which is neither reasonable nor feasible. Therefore during the standardization process for DCs, special attention has to be paid to each of the schema-specific constraints that are submitted.

The subclass Open Conceptual Domain presents the opportunity to overwrite the open DC's default data type. Closed Conceptual Domain contains the enumeration of simple DCs which make up the value domain of the Closed DC.

For a given object language the Linguistic Section allows users to subset the conceptual domain. As the data model now supports three different types of conceptual domains, the Linguistic Section is also subclassed so it can express subsetting using the proper Conceptual Domain subclass.

Sharing Simple Data Categories

DCs which have been entered into the Syntax DCR show many duplicates for simple DCs. Users create these duplicates to accommodate conceptual domains in their private workspaces. To prevent these duplicates, closed DCs should be able to share simple DCs between their conceptual domains. The Value Domain class shown in Figure 3 already allows arbitrary references to Simple DCs, and does not prevent in any way the sharing of them between multiple conceptual domains. However, for some corner cases this model is not expressive enough. DCs can be part of multiple profiles, which are stored in the Description Section class (see Figure 2). The current model assumes this globally stored information is enough to reconstruct the set of complex data categories and their conceptual domains for each profile. But when two or more complex DCs share a simple DC in different profiles, the global nature of the profile information can result in the accidental association of the simple DC with the wrong profile-specific value domain(s).

The following example demonstrates this kind of inappropriate behavior. The target is to store the following profile specific DC selections:

- profile: p1
    - complex DC: */Ca/*
        - simple DC: */Sa/*
        - simple DC: */Sb/*
    - complex DC: */Cb/*
        - simple DC: */Sa/*
- profile: p2
    - complex DC: */Ca/*
        - simple DC: */Sb/*
        - simple DC: */Sc/*
    - complex DC: */Cb/*
        - simple DC: */Sb/*

The conceptual domain for complex DC /*Ca*/ should contain simple DCs /*Sa*/, /*Sb*/ and /*Sc*/, resulting in the union of all values across all profiles. For /*Cb*/ the conceptual domain should be comprised of /*Sa*/ and /*Sb*/. In the original DCR model each of the simple DCs declares the profile to which they belong. Thus:

- /*Sa*/ belongs to profile p1;
- /*Sb*/ belongs to the profiles p1 and p2;
- /*Sc*/ belongs to profile p2.

Using this global profile information the conceptual domains of the complex DC for a specific profile are reconstructed. When trying to reconstruct /*Cb*/ for profile p1 from this information, the system will need to assess which simple DCs are present in profile p1 and appear in /*Cb*/'s conceptual domain. Here both /*Sa*/ and /*Sb*/ satisfy that criterion resulting in an (incorrect) representation of the conceptual domain for /*Cb*/ as containing /*Sa*/ **and** /*Sb*/. This is due to the fact that /*Sb*/ is shared by /*Ca*/ and /*Cb*/ in different profiles, i.e. p1 and p2.

Our fix for this behavior is to localize the profile information for simple DCs in the value domain (see Figure 4). The conceptual domain of a Closed DC now uses a set of Profile Value Domains, one for each profile. The Closed Linguistic Section can still use the 'old' Value Domain class, which means that the actual value domain of a DC, in the context of a specific profile and object language, consists of the intersection between the two value ranges of the Value Domains involved. We could now repeat the example and see that in this revised model we can now faithfully store and reconstruct the profile specific data category selections.

It should be noted that Simple DCs still contain the global profile information, and that we have only added the local information to the conceptual domain. The global info makes it possible for the user interface of the DCR implementation to easily offer a picklist of possible simple data categories which can be used to construct the value domain of a closed DC.

Conclusion

The design, implementation and use of the TC37 DCR has been fruitful as a breeding ground for ideas in the terminology community. However, to achieve wider adoption its implementation has to become fully accessible. But, as this paper has shown, the current DCR data model was in need of some revisions, leading to the revised data model presented here. Adopting this data model will enhance the stability and usability of the new DCR implementation, ISOcat.

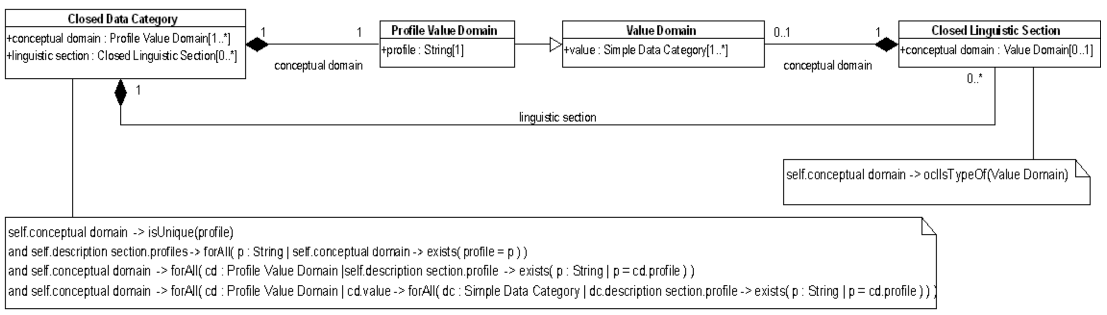

Closed Data Category
+conceptual domain : Profile Value Domain[1..*]
+linguistic section : Closed Linguistic Section[0..*]
1
1
conceptual domain
Profile Value Domain
+profile : String[1]
Value Domain
+value : Simple Data Category[1..*]
0..1
1
conceptual domain
Closed Linguistic Section
+conceptual domain : Value Domain[0..1]
1
0..*
linguistic section
self.conceptual domain -> oclIsTypeOf(Value Domain)
self.conceptual domain -> isUnique(profile)
and self.description section.profiles -> forAll(p : String | self.conceptual domain -> exists(profile = p))
and self.conceptual domain -> forAll(cd : Profile Value Domain |self.description section.profile -> exists(p : String | p = cd.profile))
and self.conceptual domain -> forAll(cd : Profile Value Domain | cd.value -> forAll(dc : Simple Data Category | dc.description section.profile -> exists(p : String | p = cd.profile)))

Figure 4 Classes related to profile-specific value domains

Acknowledgements

The authors are grateful to the TC37 community, especially contributors to ISO 12620, for their ongoing input on the DCR data model.

Bibliography

Fallside, D. C. and P. Walmsley (2004). W3C XML Schema, W3C.

Horrocks, I., P. F. Patel-Schneider, et al. (2004). SWRL: A Semantic Web Rule Language Combining OWL and RuleML.

Ide, N. and L. Romary (2004). *A Registry of Standard Data Categories for Linguistic Annotation*. International conference on Language Resources and Evaluation, Lisbon, Portugal.

ISO 16642 (2003). Computer applications in terminology — TMF (Terminological Markup Framework). International Organization for Standardization (ISO).

ISO 19757 (2006). Information technology — Document Schema Definition Languages (DSDL) — Part 3: Rule-based validation — Schematron, International Organization for Standardization (ISO).

ISO DIS 12620 (2007). Terminology and other language resources - Data categories - Specification of data categories and management of a Data Category Registry for language resources, International Organization for Standardization (ISO).

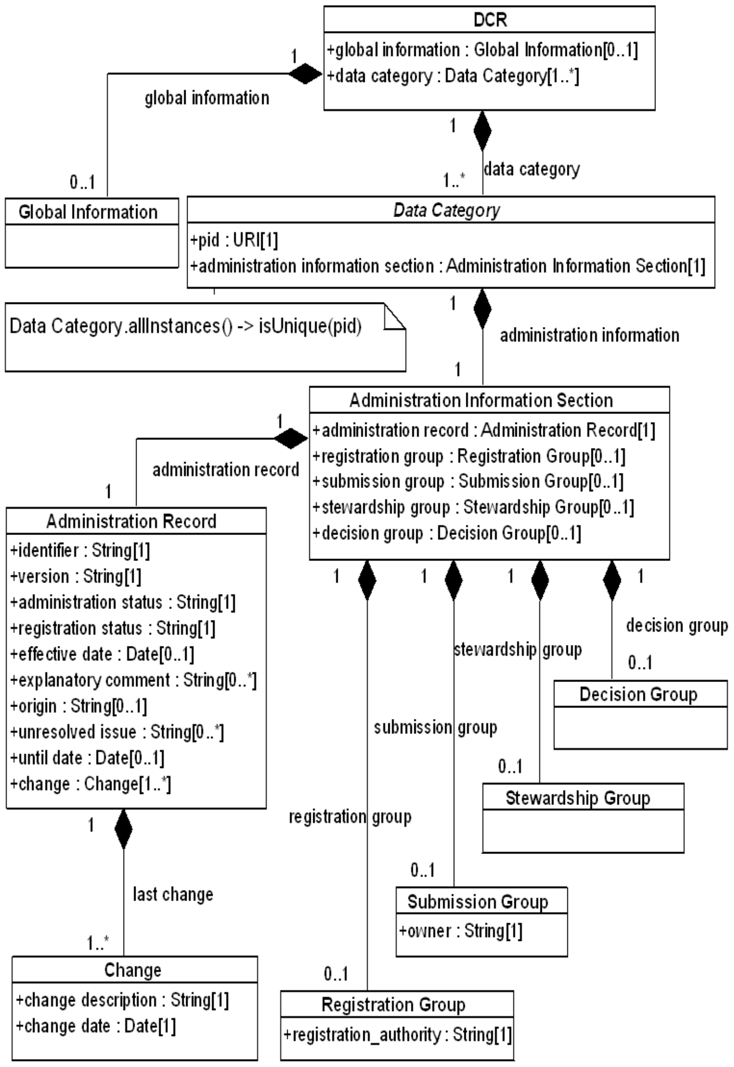
Kemps-Snijders, M., M. A. Windhouwer, et al. (2008). ISOcat: Corralling Data Categories in the Wild. International Conference on Language Resources and Evaluation. Marrakech, Marocco.

Object Management Group (2004). Unified Modeling Language Specification, Version 2.0.

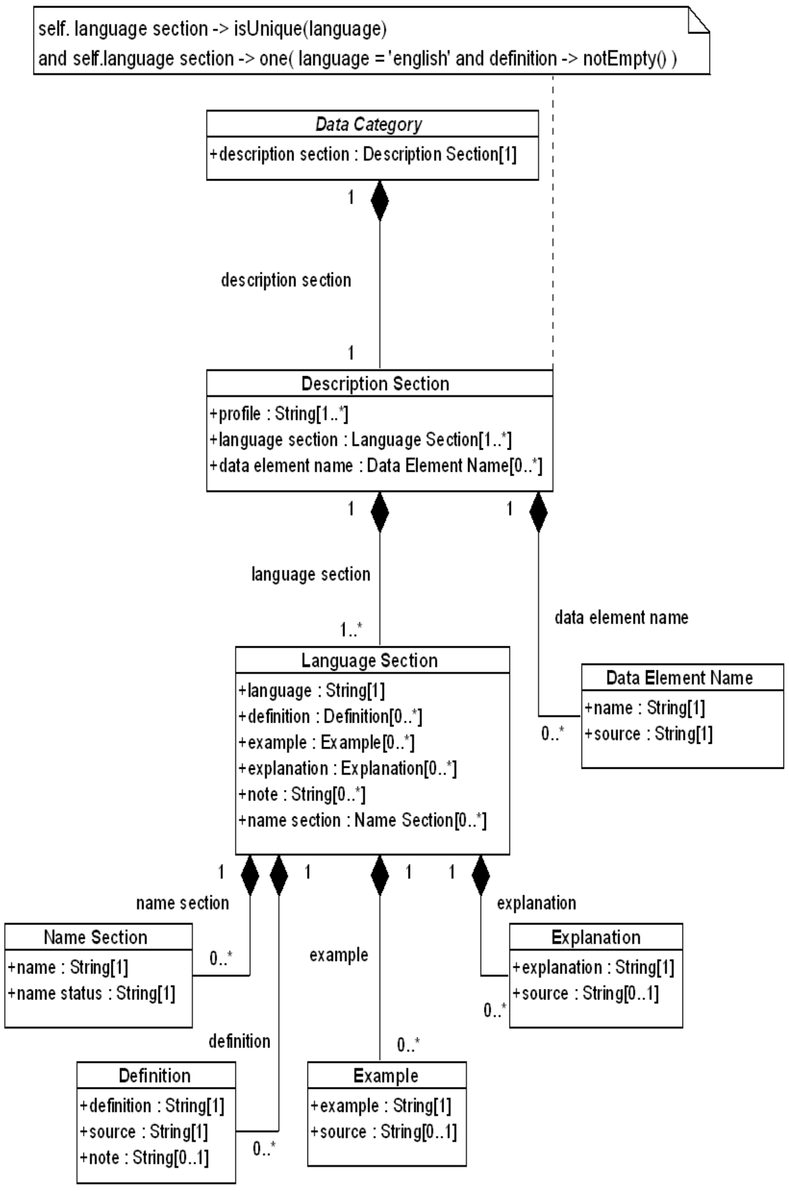
Object Management Group (2006). Object Constraint Language Specification, version 2.0.

Appendix: Revised Data Model for the ISO DCR

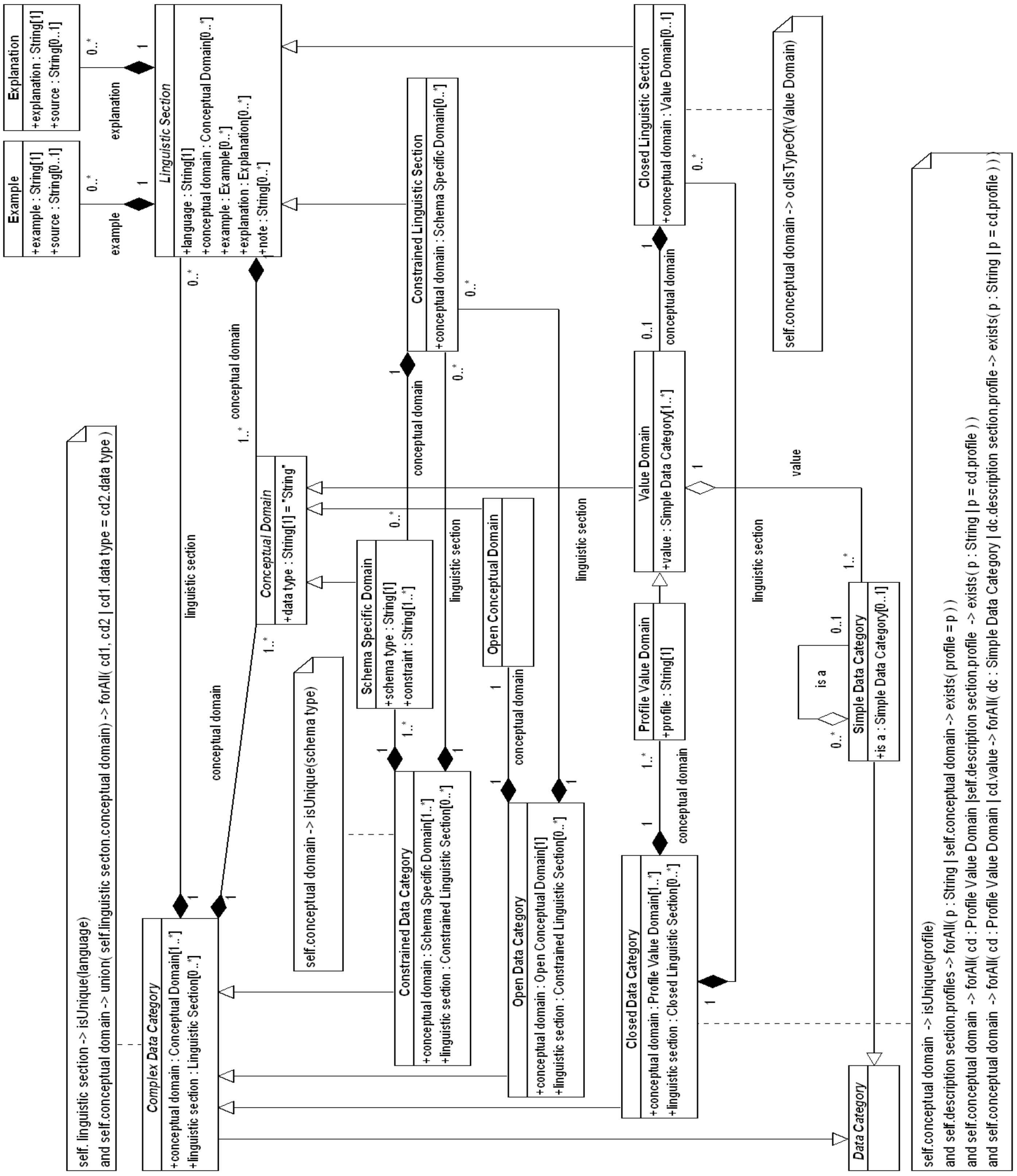
Due to space limitations and the size of the revised data model, the complete model is split up in three parts. The Data Category class appears on all three parts, and thus forms the linking pin connecting the three views.


DCR
+global information : Global Information[0..1]
+data category : Data Category[1..*]
global information
1
0..1
1
data category
1..*
Global Information
Data Category
+pid : URI[1]
+administration information section : Administration Information Section[1]
Data Category.allInstances() -> isUnique(pid)
administration information
1
1
Administration Information Section
+administration record : Administration Record[1]
+registration group : Registration Group[0..1]
+submission group : Submission Group[0..1]
+stewardship group : Stewardship Group[0..1]
+decision group : Decision Group[0..1]
administration record
1
1
Administration Record
+identifier : String[1]
+version : String[1]
+administration status : String[1]
+registration status : String[1]
+effective date : Date[0..1]
+explanatory comment : String[0..*]
+origin : String[0..1]
+unresolved issue : String[0..*]
+until date : Date[0..1]
+change : Change[1..*]
1
last change
1..*
Change
+change description : String[1]
+change date : Date[1]
decision group
0..1
Decision Group
stewardship group
0..1
Stewardship Group
submission group
0..1
Submission Group
+owner : String[1]
registration group
0..1
Registration Group
+registration_authority : String[1]

1 Administrative information



2 Language section


Explanation
+explanation : String[1]
+source : String[0..1]
Example
+example : String[1]
+source : String[0..1]
example
explanation
Linguistic Section
+language : String[1]
+conceptual domain : Conceptual Domain[0..*]
+example : Example[0..*]
+explanation : Explanation[0..*]
+note : String[0..*]
Constrained Linguistic Section
+conceptual domain : Schema Specific Domain[0..*]
Closed Linguistic Section
+conceptual domain : Value Domain[0..1]
Conceptual Domain
+data type : String[1] = "String"
Schema Specific Domain
+schema type : String[1]
+constraint : String[1..*]
Open Conceptual Domain
Value Domain
+value : Simple Data Category[1..*]
Profile Value Domain
+profile : String[1]
Simple Data Category
+is a : Simple Data Category[0..1]
is a
value
conceptual domain
linguistic section
Complex Data Category
+conceptual domain : Conceptual Domain[1..*]
+linguistic section : Linguistic Section[0..*]
Constrained Data Category
+conceptual domain : Schema Specific Domain[1..*]
+linguistic section : Constrained Linguistic Section[0..*]
Open Data Category
+conceptual domain : Open Conceptual Domain[1]
+linguistic section : Constrained Linguistic Section[0..*]
Closed Data Category
+conceptual domain : Profile Value Domain[1..*]
+linguistic section : Closed Linguistic Section[0..*]
Data Category

```
self. linguistic section -> isUnique(language)
and self.conceptual domain -> union( self.linguistic secton.conceptual domain) -> forAll( cd1, cd2 | cd1.data type = cd2.data type )
```

```
self.conceptual domain -> isUnique(schema type)
```

```
self.conceptual domain -> oclIsTypeOf(Value Domain)
```

```
self.conceptual domain -> isUnique(profile)
and self.description section.profiles -> forAll( p : String | self.conceptual domain -> exists( profile = p ) )
and self.conceptual domain -> forAll( cd : Profile Value Domain |self.description section.profile -> exists( p : String | p = cd.profile ) )
and self.conceptual domain -> forAll( cd : Profile Value Domain | cd.value -> forAll( dc : Simple Data Category | dc.description section.profile -> exists( p : String | p = cd.profile ) ) )
```

3 Conceptual domain and linguistic section