A model driven approach for the development of metadata editors, applicability to the annotation of geographic information resources

J. Nogueras-Iso, M.Á. Latre, R. Béjar, P.R. Muro-Medrano, F.J. Zarazaga-Soria

PII: S0169-023X(12)00084-5
DOI: doi: 10.1016/j.datak.2012.09.001
Reference: DATAK 1394

To appear in: *Data & Knowledge Engineering*

Received date: 28 July 2010
Revised date: 10 September 2012
Accepted date: 10 September 2012

Please cite this article as: J. Nogueras-Iso, M.Á. Latre, R. Béjar, P.R. Muro-Medrano, F.J. Zarazaga-Soria, A model driven approach for the development of metadata editors, applicability to the annotation of geographic information resources, *Data & Knowledge Engineering* (2012), doi: 10.1016/j.datak.2012.09.001

A model driven approach for the development of metadata editors, applicability to the annotation of geographic information resources

J. Nogueras-Iso, M. Á. Latre, R. Béjar, P. R. Muro-Medrano, F. J. Zarazaga-Soria

Computer Science and Systems Engineering Department
Universidad de Zaragoza
María de Luna 1, E-50018 Zaragoza, Spain

Abstract

Metadata are a key element for the development of information infrastructures because they facilitate the semantic description of contents and services. However, the diversity and heterogeneity of metadata standards have become a barrier for the generation of these metadata. Many metadata editors are not useful anymore because they do not support the latest version of metadata standards or the new profiles arisen in the market. Thus, this work proposes a model driven approach for the development of metadata editors, more focused on the generic treatment of metadata models than on the development of specific edition forms for a reduced set of metadata standards. This approach has been tested in the context of spatial data infrastructures for the development of an Open Source tool called CatMDEdit. Additionally, the approach could be also applied to improve the efficiency of any metadata editor using a metamodeling development strategy.

Key words: Spatial Data Infrastructures, SDI, Metadata, Annotation, Model Driven Engineering, MDE, Model Driven Architecture, MDA, Metamodeling, SKOS

Email addresses: jnog@unizar.es (J. Nogueras-Iso), latre@unizar.es (M. Á. Latre), rbejar@unizar.es (R. Béjar), prmuro@unizar.es (P. R. Muro-Medrano), javy@unizar.es (F. J. Zarazaga-Soria).

[1] This work has been partially supported by the Spanish Government through the project TIN2007-65341, the National Geographic Institute of Spain (IGN), and GeoSpatiumLab S.L.

1 Introduction

Spatial Data Infrastructures (SDI) are special types of information infrastructures consisting of the relevant base collection of technologies, policies and institutional arrangements that facilitate the availability of and access to spatial data. Traditionally, spatial data (also known as geographic information) were the core component of Geographic Information Systems (GIS), which is the term commonly used to refer to the software packages that allow to capture, store, check, integrate, manipulate, analyze and display them. However, the potential of spatial data as an instrument to facilitate decision-making and resource management in diverse areas (e.g., natural resources, facilities, cadaster or agriculture) of government or private sectors has led to the evolution of GIS into the broader concept of SDI [1].

One of the main elements for the success in the development of SDI or any other type of information infrastructure is the appropriate annotation of resources to be accessed and distributed by means of metadata. Metadata constitute the mechanism to characterize data and services (e.g., descriptions of the content, quality, condition, authorship and any other features) in order to enable other users and applications to make use of such data and services. However, due to the heterogeneity of contents in information infrastructures, it is not possible to consider a unique metadata model or schema.

The diversity of metadata standards has been a critical issue for the development of SDIs. During the last fifteen years, standardization bodies have proposed different metadata standards such as the *Content Standard for Digital Geospatial Metadata (CSDGM)* [2] or *ISO 19115 Geographic Information – Metadata* [3]. Additionally, apart from the standards, it is also common to find application profiles and extensions of these standards. The Dublin Core Metadata Initiative [4] defines a *Metadata Application Profile* as a declaration of the elements (either selected from the standard, or new elements) that an organization or user community employs in their metadata, and how these elements have been customized to a particular application domain. Within the SDI context, there are multiple examples of metadata profiles for the description of remote sensing data [5,6], environmental data [7,8], or the customization of general metadata standards such as Dublin Core [9].

In parallel to the definition of this wide range of geographic metadata standards, there has been an increasing need for desktop or web-based metadata editors able to manage these complex standards (hundreds of elements with different data types organized in hierarchical entities) and providing, among other functionalities, an internationalized interface, online help, validation of standard conformance, and serialization to XML or other semi-structured formats. Through different surveys [10,47], it can be verified that the development

of these metadata editors has been highly promoted during the last years: about 50 tools have been published by different organizations and software companies. At the beginning, these tools were developed to support a unique metadata model. However, these first tools based on fixed structures became rapidly out-of-date. In contrast, nowadays software developers are sensitive to this problem of flexibility. Instead of implementing a particular metadata model, most of current editors enable as well the configuration of the model using some kind of schema language.

From a software engineering perspective, we could say that annotation tools in the SDI context have moved towards a higher level of abstraction: the focus is now on the management of metamodels. According to ISO/IEC 11179-3 [11], a metamodel provides a mechanism for understanding the precise structure and components of the specified models, which are needed for the successful sharing of the models by users and/or software facilities. Although the available documentation of most editors does not reveal a conscious interest on metamodels, the key component of these tools is the mechanism to describe the different metadata standards that can be used later to customize dynamically the software for editing metadata in conformance to these standards.

The need for flexibility and the consequent increase of model abstraction is leading the development of metadata editors to a stage that, although not directly intended, is close to the software engineering paradigm of Model Driven Engineering (MDE). MDE focuses on models as the primary artifact in the development process, with transformations as the primary operation on models, used to map information from one model to another [12]. However, in order to avoid ad-hoc and un-efficient implementations of this MDE paradigm, it is important to follow a systematic and acknowledged methodology such as the one defined by the Model Driven Architecture (MDA), the OMG instantiation of MDE [13,19]. Although in the SDI context MDA has been considered for interoperability issues and data access services [14,15,16], it has received little attention for the development of metadata editors. The objective of this work is to provide the guidelines and framework to accomplish the development of metadata editors according to MDA in order to facilitate their rapid development, decreasing the development effort and, at the same time, augmenting their efficiency and flexibility. This paper presents the different domain specific languages and transformations required to define and transform the models involved in the development of a metadata editor: the Platform Independent Model (PIM) for the representation of supported metadata schemas; the Platform Specific Models (PSM) for the specification of a set of GUI (Graphical User Interface) edition forms, and the representation of controlled vocabularies; and the transformation of PSM models into code.

The rest of this paper is structured as follows. Section 2 presents our proposal to apply an MDA approach in the development of metadata editors. Following this proposal, Section 3 describes how this proposal has been used in CatMDEDit, an open-source metadata editor, for the support of different

metadata standards and profiles. Then, Section 4 describes the related work in the development of annotation tools from a metamodeling perspective, and discusses the benefits from applying an MDA development approach. Finally, Section 5 draws some conclusions and outlines future work.

2 Development of metadata editors according to an MDA approach

2.1 Overview

As stated by Djurić et al. [17], if we look back to the history of software development, we can see a notable increase of models abstraction. The activity of modeling is now more separated from the specification of the details of the underlying platform. This evolution allows domain experts to focus on defining reusable models of the real world, alleviating them from acquiring the knowledge about specific computer systems. Two examples of this evolution are the methodologies known as Model Integrated Computing (MIC) [18] and Model Driven Engineering (MDE). MIC was a precursor methodology for generating application programs automatically from multi-aspect models. Nowadays, MDE encompasses the software modeling methodologies which focus on creating and exploiting domain models (through successive transformations), rather than on the concepts of a specific computing platform.

Model Driven Architecture (MDA) is the OMG instantiation of MDE [19]. MDA separates the specification of system functionalities from the specification of this functionality on a specific technology platform. MDA defines three viewpoints on a system: a computation independent viewpoint, a platform independent viewpoint, and a platform specific viewpoint. Each viewpoint is represented by means of a viewpoint model (also called view). A Computation Independent Model (CIM) is a view of a system that does not show details of the structure of systems. A CIM is sometimes called a domain model and focuses on representing the environment and the requirements of the system. A Platform Independent Model (PIM) is a view of a system that focuses on the operation of a system while hiding the details necessary for a particular platform. It shows the part of the system specification that does not change from one platform to another, i.e. it is the specification of a system for a technology-neutral virtual machine. Finally, a Platform Specific Model (PSM) combines the specification in the PIM with the details that specify how that system uses a particular type of platform.

A crucial issue to allow the transformation between models in MDA is the appropriate definition of the domain specific languages, which are used in turn to build models. A domain specific language consists of three elements: an abstract syntax to define the concepts of the language, a concrete syntax to provide a graphical or textual notation, and a description of the semantics of the language [20]. The abstract syntax of a domain specific language is defined

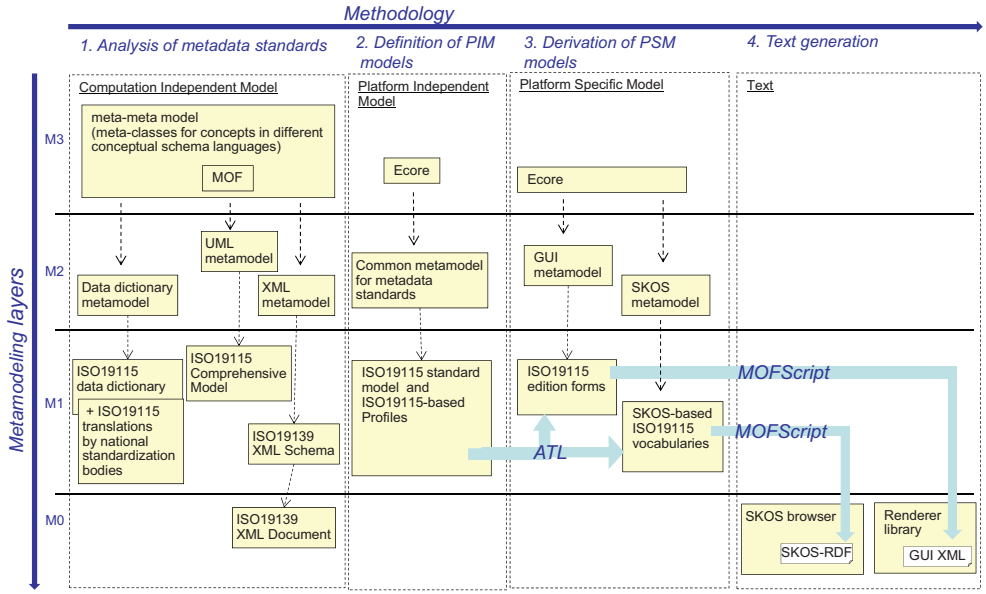


Figure 1. MDA approach for the development of metadata editors

by means of a metamodel, which can be considered as an explicit description (constructs and rules) of the way to build a domain-specific model. Metamodeling allows strict and agile automatic processing of models and metamodels. Models in the MDA approach are based on the four-layer metamodeling pattern [21]: a meta-metamodel (M3) layer that provides concepts to define metaclasses of arbitrary conceptual schema languages (e.g., *Class* or *Association* for UML); a metamodel (M2) layer that defines the concepts used in a conceptual schema language (e.g., the concepts used in UML language); a model (M1) layer that contains models of the real world, which are represented by concepts defined in the corresponding metamodel at M2; and an instance (M0) layer that contains the things from the real world.

Fig. 1 presents our approach for adopting an MDA methodology in the development of metadata editors. In this approach, the following steps are considered:

- Analysis of metadata standards in terms of their original conceptual schema languages: The definition of a geographic metadata standard can be understood as the CIM model that originates the transformation between models towards the development of the metadata editor for such standard. For instance, Fig. 1 shows ISO 19115 from the perspective of the four-layer meta-modeling architectural pattern.
- Definition of PIM models in terms of a common metamodel: The previous CIM models are transformed into a PIM model, whose metamodel takes into consideration features that may arise in different metadata standards such as ISO 19115, Dublin Core and other ones.

- Generation of derived PSM models: The previous PIM models are transformed into PSM models that describe, in an abstract way, the GUI of the metadata edition forms. Additionally, the controlled vocabularies (e.g., codelists or enumerations) are transformed into the SKOS language [22], a domain specific language proposed by W3C for the standardized representation of knowledge organization systems such as thesauri, taxonomies and other controlled vocabularies.
- Text generation: As a final step, the previous PSM models are transformed into code, i.e. either software or other artifacts that can be interpreted and executed on a specific platform. As a possible alternative for the transformation PSM GUI-based models, we propose the serialization of the GUI-based models into an XML serialization, which can be parsed by a library to generate dynamically the edition forms implemented, for example, as a Java desktop application. With respect to the SKOS representation of vocabularies, their transformation into an SKOS-RDF encoding is proposed.

These steps are detailed in next subsections. With respect to the implementation of this approach, we have used the tools provided by the Eclipse Modeling Framework (EMF) [23], which is currently used in many model driven approaches [16,24]. EMF is an open source Java implementation of a core subset of the Meta-Object Facility (MOF) specification [13], a specification promoted by OMG to create an MDA framework for constructing and managing technology neutral metamodels. The MOF-like core metamodel in EMF is called Ecore. Additionally, EMF provides different tools for the transformation between models. In the case of the transformation from PIM to PSM models, the Atlas Transformation Language (ATL) [25] has been selected. ATL is a hybrid language that combines declarative and imperative constructs for the definition of transformation rules from source to target models. Finally, the transformation of PSM models to code (text serialization) has been designed using the MOFScript language, one of the most extended languages for model-to-text transformation [26]. For the sake of space only selected excerpts of the ATL and MOFScripts transformations will be shown in the following subsections. For further details, a web site with the full code of the metadata models in Ecore format, and the ATL/MOFScript transformations is provided [2], together with some examples of the input and output models of these transformations.

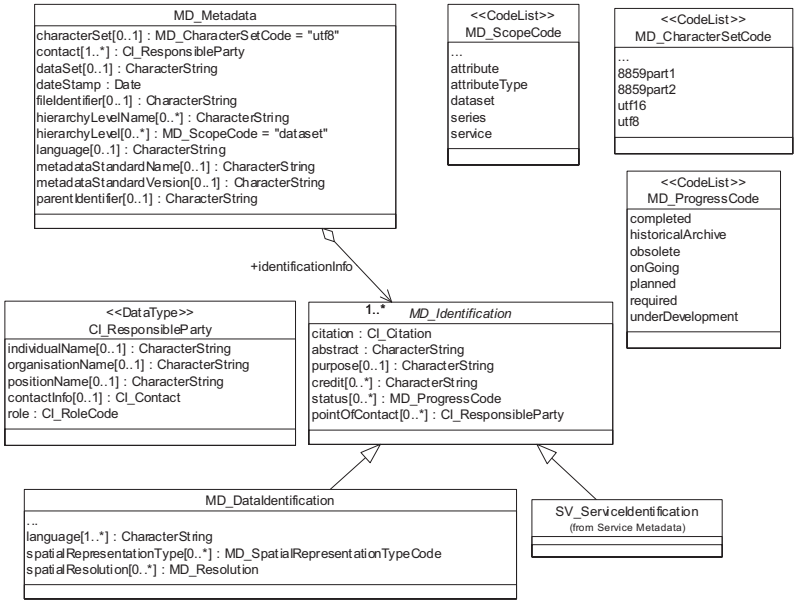
2.2 Analysis of metadata standards (CIM models)

Nowadays, there are three main families of metadata standards and profiles to describe geographic information resources: Content Standard for Digital

[2] `http://catmdedit.sourceforge.net/mda`

Geospatial Metadata (CSDGM) [2], ISO 19115 Geographic Information – Metadata [3] and Dublin Core [4]. CSDGM is the oldest proposal. Released in 1994 with the sponsorship of the U.S. Federal Geographic Data Committee, it is still used and integrated in many GIS tools. ISO 19115 is the metadata standard that it is currently world-wide accepted and has been adopted by the great majority of national standardization bodies and thematic communities in the domain of geographic information. Finally, although Dublin Core is a general-purpose metadata standard, it is also widely used in the domain of geographic information to establish minimum discovery mechanisms and to promote interoperability [27].

By its inherent nature, the definition of a metadata standard or a metadata profile can be considered as a CIM model of an annotation tool because it does not show details of the structure of the system, it just shows the model of the content that must be generated through this system. In order to illustrate the features of these metadata standards and their correspondence with the four layer metamodeling perspective, this section describes the case of ISO 19115.



Data dictionary:

	Name/Role name	Short Name	Definition	Obligation/ Condition	Maximum occurrence	Data type	Domain
1	MD_Metadata	Metadata	root entity which defines metadata about a resource or resources	M	1	Class	Lines 2-22
2	fileIdentifier	mdFileID	Unique identifier for this metadata file	O	1	CharacterString	Free text
...	...	...	...	...	...	...	...

Figure 2. Definition of class *MD_Metadata* in ISO 19115

The conceptual schema language used for expressing the comprehensive model of ISO 19115 metadata and their profiles is UML. Additionally, several stereotypes have been defined for expressing special features of metadata models. Some of these stereotypes are the following: *DataType*, a descriptor of a set

of values that lack identity; *Enumeration*, a data type whose instances form a list of named literal values; *CodeList*, a more open enumeration where the list of values can be extended if necessary; or *Union*, a set of alternative classes/types that can be used without the need to create a common supertype/class. For instance, Fig. 2 shows a UML diagram with the definition of the class *MD_Metadata*, which is the root class in ISO 19115 for describing a geographic resource. The figure also shows a containment relationship with the *MD_Identification* class and its derived classes. Additionally, UML static diagrams are accompanied with a data dictionary in tabular form, whose rows define UML classes, attributes and relationships by means of seven attributes (*Name/Role name*, *Short name/code*, *Definition*, *Obligation/Condition*, *Maximum occurrence*, *Data type*, and *Domain*). Fig. 2 includes two rows of the data dictionary describing the *MD_Metadata* class and one of its attributes. This example will be used to illustrate the MDA approach along this work.

The XML encoding of this metadata standard is specified through the ISO 19139 technical specification [28], which establishes the rules to map the UML structure of the standard into an XML serialization through a series of XML Schemas. Trying to establish the parallelism among all these documents for the definition of the standard and the four layer meta-modeling perspective (see Fig. 1), the UML metamodel and the XML metamodel would be at M2; ISO 19115 standard document and the associated ISO 19139 XML Schemas would be at M1; and ISO 19139-compliant XML metadata files would be at M0.

The other two metadata standards, CSDGM and Dublin Core, are described by means of other conceptual schema languages and artifacts: BNF (Backus-Naur-Form) and a DTD (Document Type Definition) for the syntax and XML encoding in the case of CSDGM; and the DCMI abstract model document [29] and RDF Schemas for specifying the constructs and RDF encoding of Dublin Core. Although the mechanisms to express these standards are different, a parallelism can be established between UML classes and BNF production rules or RDF Schema descriptions.

2.3 Definition of PIM models in terms of a common metamodel

Currently, the definition of the syntax and semantics of any of the metadata standards analyzed in Section 2.2 is distributed in a set of heterogeneous documents. Some of these documents are UML models or schemas in conformance with a schema language (e.g., XML-Schema or DTD), but in other cases these documents are just plain text documents (e.g., documents with description of elements organized in different sections, or data dictionaries not provided in tabular format that must be manually extracted). However, in order to

apply MDA for the automated development of metadata editors we need a machine-readable definition of metadata standards and in one unique and compact document. For instance, analyzing the case of ISO 19115, we could have considered that the UML profile established in ISO 19103[3] could be a plausible candidate as the domain specific language (DSL) for PIM models. Nevertheless, there are three main reasons not to recommend it. First, some problems found in the models of ISO 19100 series (which includes ISO 19115) like the disregarding of the UML specification or some conflicting data types are an obstacle for the direct integration of these models in an MDA development [14]. Second, the automatic serialization of metadata in XML format cannot be directly inferred from the ISO 19115 model. Issues such as the URI for XML namespaces or XML attributes are not explicitly included in the UML model. Third, the ISO 19115 UML model does not include multilingual information about labels, definitions, conditionality or examples of metadata elements, something that is an essential for a multilingual metadata editor with on-line help. Therefore, due to these weaknesses, we decided to propose a new DSL to define PIM models. Additionally, as many metadata standards and profiles are needed in this context of geographic information, our DSL is flexible enough to support at least the family of metadata profiles derived from ISO 19115, CSDGM and Dublin Core standards.

Fig. 3 shows the proposed common metamodel for the PIM models, i.e. the abstract syntax of the DSL. This metamodel has been built by means of the Ecore metamodeling language. The *Package* class is an auxiliary containment class that consists of a series of related standards (represented by the *Standard* class) and their derived metadata profiles (represented by the *Profile* class). The *Term* class represents any of the components that may be found in a metadata profile, e.g. an entity, an element, or a controlled vocabulary. In addition, it is also possible to customize the standard elements within the context of a specific profile. These possible modifications are stored in the intermediate class called *TermInProfile*.

All these classes have some attributes and peculiarities that should be noted:

- The *Standard* class contains an attribute called *namespaceURI* that holds a unique identifier of the standard, which is used for the namespace of the standard elements in an XML encoding.
- The *Term* class contains the following attributes: *identifier*, identifier of the term within a standard; *name*, name of the term as used in the encoding of a metadata instance; and a one-to-many *property* association with the *Property* class to store additional information (i.e., *label*, *definition*, *example*,

[3] Technical specification that proposes a conceptual schema language to be used in the models defined by the ISO TC211 technical committee for geographic information in the ISO 19100 series standards.

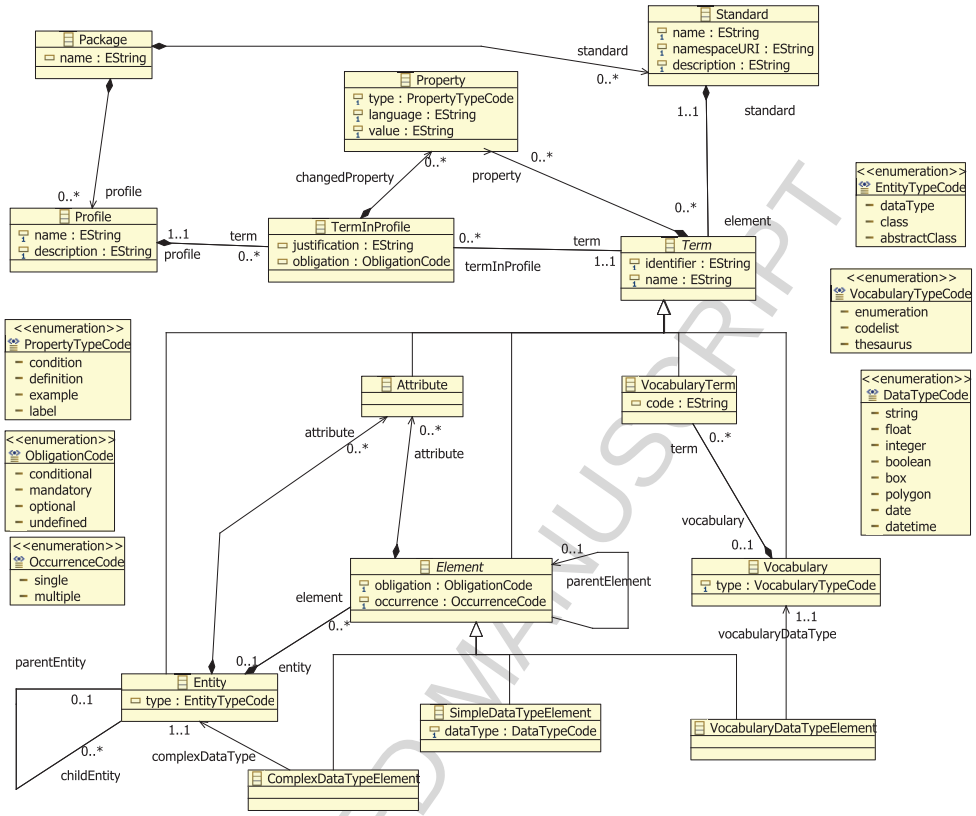


Figure 3. Metamodel of PIM models

Table 1
Mapping of concepts between CIM and PIM metamodels

ISO 19115 metamodel		PIM metamodel	
Class		Entity	type=class
AbstractClass			type=abstractClass
DataType			type=dataType
Union			*an artificial Entity with type=abstractClass is created*
Enumeration		Vocabulary	type=enumeration
CodeList			type=codelist
(thesauri are not explicitly represented in the standard, but they are recommended in the data dictionary for filling several elements whose data type is CharacterString)			type=thesaurus
Attribute	*data type is well-known*	SimpleDataTypeElement	
	data type is an enumeration or codelist	VocabularyDataTypeElement	
	data type is user-defined	ComplexDataTypeElement	
Association role			
(member of a codelist or enumeration)		VocabularyTerm	

or *condition*) in multiple languages.

- The *Term* class can be specialized into different classes:
  - *Entity*: it represents a set of metadata elements describing the same aspect of data. The kind of entity is indicated by the attribute *type*, which may be: *dataType* (entity that represents a data type), *class* (entity that represents a class of the standard), and *abstractClass* (entity that represents an abstract class).
  - *Element*: it represents an element that belongs to an *Entity* and has ad-

Table 2
Mapping between CIM data dictionary attributes and the PIM metamodel

ISO 19115 Data Dictionary		PIM metamodel	
row type	**attribute**	**class**	**attribute/association**
any row	Row number	Term	identifier
	Name/Role name		name
			property *(type=label)*
	Definition		property *(type=definition)*
attribute/ association role	Obligation	Element	obligation
	Condition		property *(type=condition)*
	Maximum occurrence		occurrence
attribute *(DataType =string, integer, … simple data type)*	Data Type	SimpleDataElement	dataType
	Domain		property *(type=example)*
attribute *(Data Type = enumeration or codelist class)*	Domain	VocabularyDataTypeElement	vocabularyDataType
attribute *(Data Type = datatype class)*	Domain	ComplexDataElement	complexDataType
association *role*	Domain		
member of codelist or enumeration	Code	VocabularyTerm	code

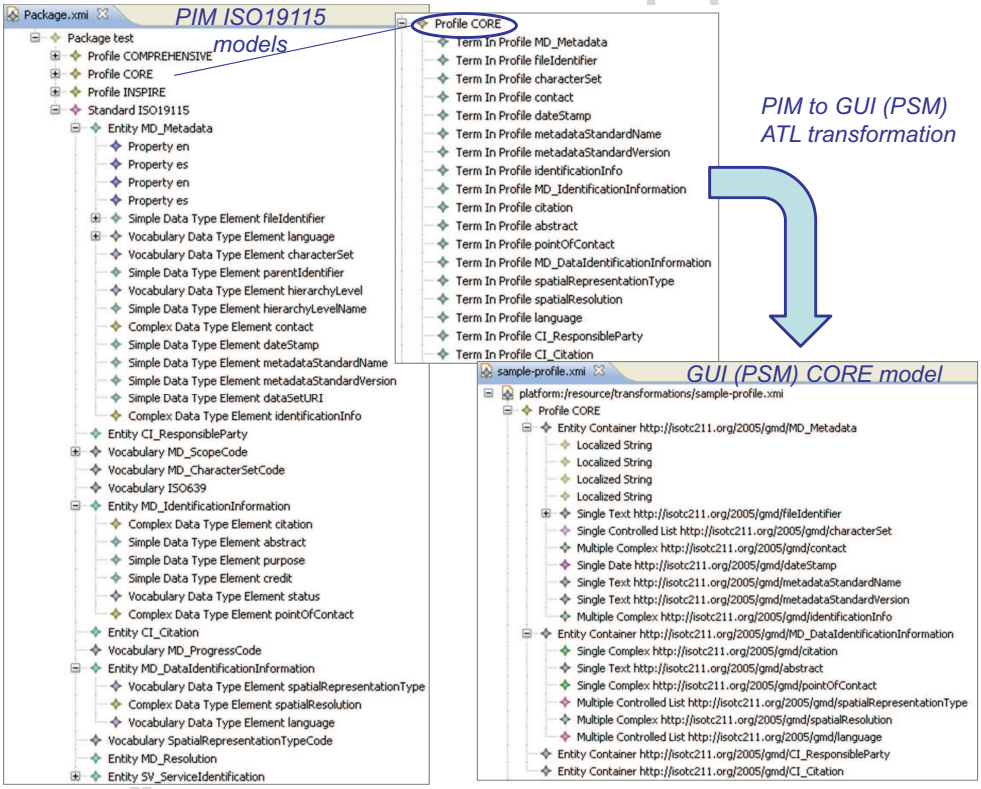


Figure 4. ISO 19115 *MD_Metadata* class in terms of the PIM and PSM metamodels

ditional attributes: the *obligation* attribute identifies whether the element is *mandatory*, *optional*, or *conditional* (i.e., mandatory under certain conditions expressed in a *property* association); and the *occurrence* attribute indicates the multiplicity of an element (*single* or *multiple*). The *Element* class has also different subclasses according to the data type of the information stored in a metadata record: *SimpleDataTypeElement* for well-known data types such as strings, numeric types, dates or geometries (see the *dataType* attribute); *ComplexDataTypeElement* for data types defined by means of another *Entity*; and *VocabularyDataTypeElement* for values that belong to a controlled vocabulary (see the *Vocabulary* class).

  - *Attribute*: it represents an additional element to characterize an *Entity* or an *Element*. This type of term is very common in the XML representation of metadata standards. Although they are not explicitly mentioned in the documents defining the metadata standards, they are frequently required in their XML encoding.
  - *Vocabulary*: it represents a controlled vocabulary. The *type* attribute indicates the specific type of controlled vocabulary: *enumeration* (a list of named literal values), *codelist* (a more open enumeration with a flexible list of named literal values), or *thesaurus*.
  - *VocabularyTerm*: it represents an element of a controlled vocabulary that may be identified with a specific *code*.
- The *TermInProfile* class allows the redefinition of some of the features of a term contained in a metadata application profile. On the one hand, the *obligation* attribute can be used to impose a more restrictive obligation value of an element with respect to the original one in the metadata standard (*obligation* attribute in *Element* class). On the other hand, the *changedProperty* relation can be used to override additional information such as labels or definitions.

Tables 1 and 2 describe the mapping between CIM and PIM metamodels, i.e. the transformation rules between CIM and PIM models. Table 1 shows the association between the concepts of ISO 19115 UML and the PIM metamodel. Table 2 shows how the data dictionary attributes of ISO 19115 are directly included in the concepts of the PIM metamodel. Fig. 4 (left side) shows an excerpt of the ISO 19115 model in terms of the PIM metamodel, which has been edited with the EMF sample reflective Ecore model editor. According to the aforementioned CIM to PIM mapping, this excerpt corresponds with the UML diagram already shown in Fig. 2. However, since CIM models can be expressed in very different conceptual schema languages (UML, BNF, XML Schemas, RDF Schemas) and accompanying documents (with data dictionaries, additional information, etc.), it is not possible to propose a unique method for the transformation of CIM models into PIM models. Anyway, programs to parse, for instance, the ISO 19115 data dictionaries (in the form of tabular files) could be easily designed to define the ISO 19115 metadata standard, and its derived metadata profiles, in terms of a concrete syntax such as the TCS textual notation [30].

2.4 Generation of derived PSM models

The next step in the MDA methodology is the transformation of PIM models into PSM models, which are closer to the final implementation of a metadata editor for a specific platform. An important decision in this step is to generate two distinct PSM models from the source PIM model. The first model en-

ables the definition of the edition forms for the different entities of metadata records. The second model is focused on the representation of controlled vocabularies (codelists, enumerations or thesauri) that are imposed by metadata standards. Since there are well-known domain specific languages to represent and manage these vocabularies, our proposal is to reuse them instead of integrating these vocabularies within the language for defining edition forms. Additionally, this decision increases the flexibility of metadata editors for the semantic annotation of resources with well-known knowledge sources already available in SKOS format [31].

2.4.1 A PSM model for edition forms

Before deciding which would be the most appropriate domain specific language for the definition of the edition forms that should enable the update of a metadata record in conformance to a specific profile, a set of functional requirements were established:

- The edition interface of a metadata profile should consist of a synchronized set of edition forms, each of them focused on a particular entity of the profile.
- An edition form should only show those elements that belong to the metadata profile.
- An edition form should avoid the complexity of inheritance hierarchies between entities. An edition form should facilitate the edition of both the own elements of the entity and the elements inherited from its super-entities (i.e. the *parentEntity* association in the PIM metamodel).
- An edition form should avoid the recursive hierarchical relations between entities and sub-entities, i.e. the *complexDataType* association of *ComplexDataElements* that belong to an entity in the PIM metamodel (see Fig. 3). In the case of editing a *ComplexDataElement*, a subordinate edition form should be opened.
- An edition form should provide enough internationalized information to understand the meaning and features of elements.
- An edition form should be able to parse and generate the XML encoding of an entity.

Taking into account these requirements, we considered first the adoption of existent GUI description languages like XUL [4] , SwingML (Swing Markup Language) [32] or SwiXML [5] . All these languages allow the possibility of defining a user interface by means of a wide range of GUI widgets (e.g., panels, buttons, labels, text-fields, list, trees, tables, etc.). However, there were two main

[4] Mozilla XML-based user interface language, `https://developer.mozilla.org/En/XUL`

[5] `http://www.swixml.org/`

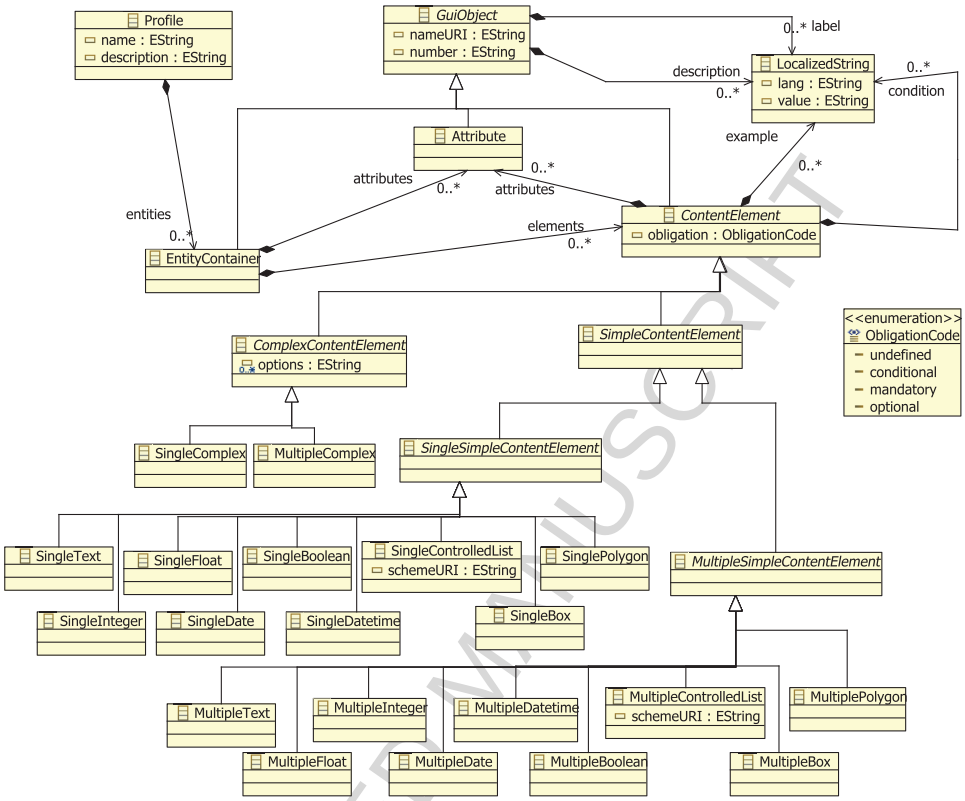


Figure 5. GUI-based PSM metamodel

problems for the adoption of these languages. On the one hand, there is a big semantic difference between the high-level concepts of the PIM model and the low-level GUI widgets provided by these GUI languages. For instance, one could consider that a single data element of a PIM model could be easily transformed into a set of widgets provided by these languages consisting of: labels to display the name, the definition and other additional information; and a text area for introducing its value. Nevertheless, richer GUI constructs are needed to take into account the obligation of an element, introduce appropriate masks according to the data type (e.g., masks for dates, numbers or geometries), or provide internationalized information [6]. On the other hand, the edition forms should embed enough information to deal with the XML encoding of an entity. Although the metadata editor does not need to show the XML tags (and their attributes) to the final user, this information should be stored in some kind of special hidden GUI artifacts, not directly supported by these languages.

[6] The Java internationalization methodology or other alternative solutions could be adopted to internationalize edition forms. However, the generation of external property files would increase the complexity of the transformation from PIM models.

Table 3
Mapping of concepts between PIM and GUI-based PSM metamodels

PIM metamodel		GUI (PSM) metamodel	
class	**attribute/association**	**class**	**attribute/association**
Profile		Profile	
Term	name	GUIObject	nameURI *(computed from the source model Term and the namespaceURI of the standard to which it belongs)*
	identifier		number
	property (*type=label*)		label *(can be redefined in profile)*
	property *(type=definition)*		definition *(can be redefined in profile)*
Entity *(type=class)*	attribute	EntityContainer	attributes
	element		elements *(including elements from the source Entity and its superclasses)*
Attribute		Attribute	
Element	obligation	ContentElement	obligation *(can be redefined in profile)*
	property *(type=condition)*		condition *(can be redefined in profile)*
	property *(type=example)*		example *(can be redefined in profile)*
SimpleDataType Element	*(occurrence=single and datatype=string)* *(occurrence=multiple and datatype=string)* *(occurrence=single and datatype=XX)* *(occurrence=multiple and datatype=XX)*	SingleText	
		MultipleText	
		Single*XX*	
		Multiple*XX*	
VocabularyDataType Element *(occurrence=single)*	vocabularyDataType	SingleControlled List	schemeURI *(derived URI from source Vocabulary)*
VocabularyDataType Element *(occurrence=multiple)*	vocabularyDataType	MultipleControlled List	schemeURI *(derived URI from source Vocabulary)*
ComplexDataType Element	complexDataType	ComplexContent Element	options *(derived nameURI of source entities connected with the complexDataType association)*
ComplexDataType Element	*(occurrence=single)*	SingleComplex	
	(occurrence=multiple)	MultipleComplex	

Therefore, due to the difficulties to adapt these languages for the required functional requirements, the final decision was to define a new domain specific language. Fig. 5 shows the metamodel of the proposed language. In this metamodel, a metadata profile (*Profile* class) can be edited by means of a series of *EntityContainer* objects, which are GUI objects representing the edition form of an entity. In turn, these *EntityContainer* objects contain *ContentElement* objects, which represent a GUI widget for the edition of a metadata element. This proposed language fulfills the required functional requirements, but still provides enough flexibility to transform a user interface based on this language into a final code implementation that may operate on different platforms (desktop or web applications). The objective of this language is to provide an intermediate step for the final serialization into an XML encoding, which can be parsed later by a specialized graphical library in charge of rendering the edition forms. The *ContentElement* class is the superclass of a hierarchy of subclasses according to the data type and multiplicity of elements. The aim of this inheritance hierarchy is to facilitate as much as possible the model-to-text transformations.

Table 3 shows the mapping between the concepts in the PIM and GUI-based PSM model. This mapping has been implemented using the ATL language. For the sake of space Fig. 6 only presents the rule to transform *Entities* of the PIM model into *EntityContainers* of the PSM model. Among other things, this rule makes use of helper functions to identify its internationalized labels (*getLabels* function), which may be overridden by a specific profile, and link this entity with its elements. The *getAllElements* function obtains all elements

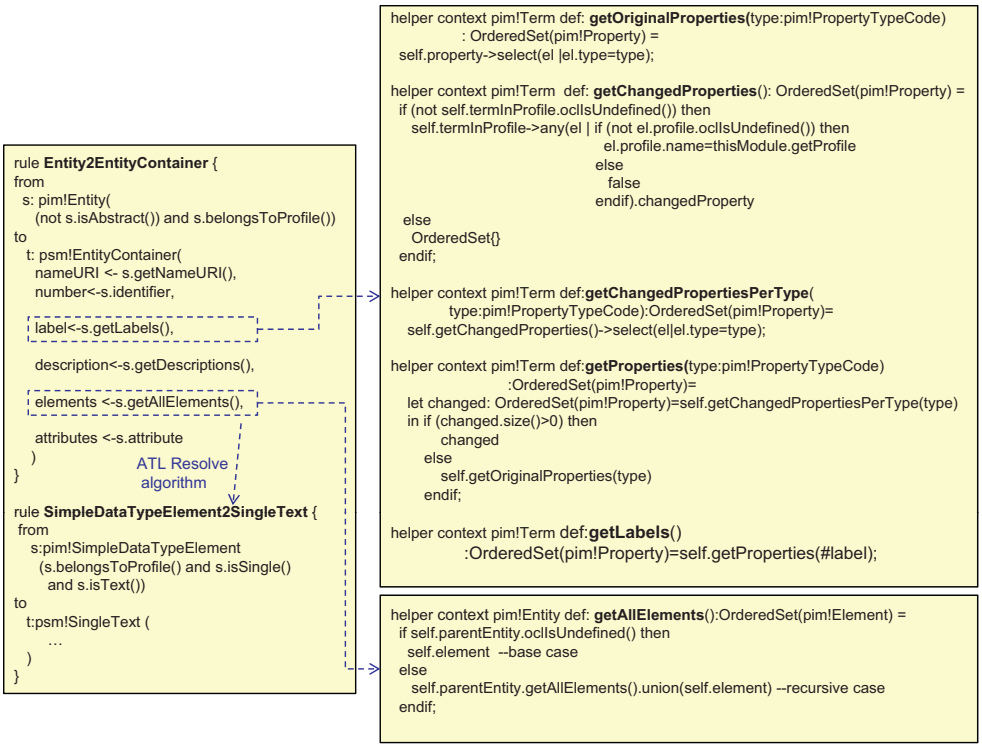


Figure 6. *Entity* to *EntityContainer* mapping rule

in the inheritance hierarchy, which are matched later to its appropriate *EntityContainer* through the ATL resolve algorithm. Fig. 4 (right side) shows an excerpt of the ISO 19115 *MD_Metadata* entity in terms of the PSM model (using the EMF sample reflective Ecore model editor), which has been derived from its corresponding PIM model. This figure exemplifies some effects of the mapping rules: only the elements of the profile are present in the PSM model (e.g., the *hierarchyLevel* element of *MD_Metadata* is not present in the Core profile); the *EntityContainer* objects include elements from super-entities (*MD_DataIdentification* includes the elements from *MD_Identification*); every *EntityContainer* and *ContentElement* has its own URI.

2.4.2 Using SKOS for the representation of controlled vocabularies

SKOS (Simple Knowledge Organization System) is a family of formal RDF-based languages for representing controlled structured vocabularies, including thesauri, classification schemes, taxonomies and subject-heading systems [22]. SKOS is currently developed within the W3C framework and has been widely accepted by the research community and the industrial sector since the initial proposal of this language in the SWAD-Europe research project. This wide acceptance was mainly due to the lack of standardized exchange formats in this field [33]. For instance, until 2011 ISO norms for monolingual and mul-

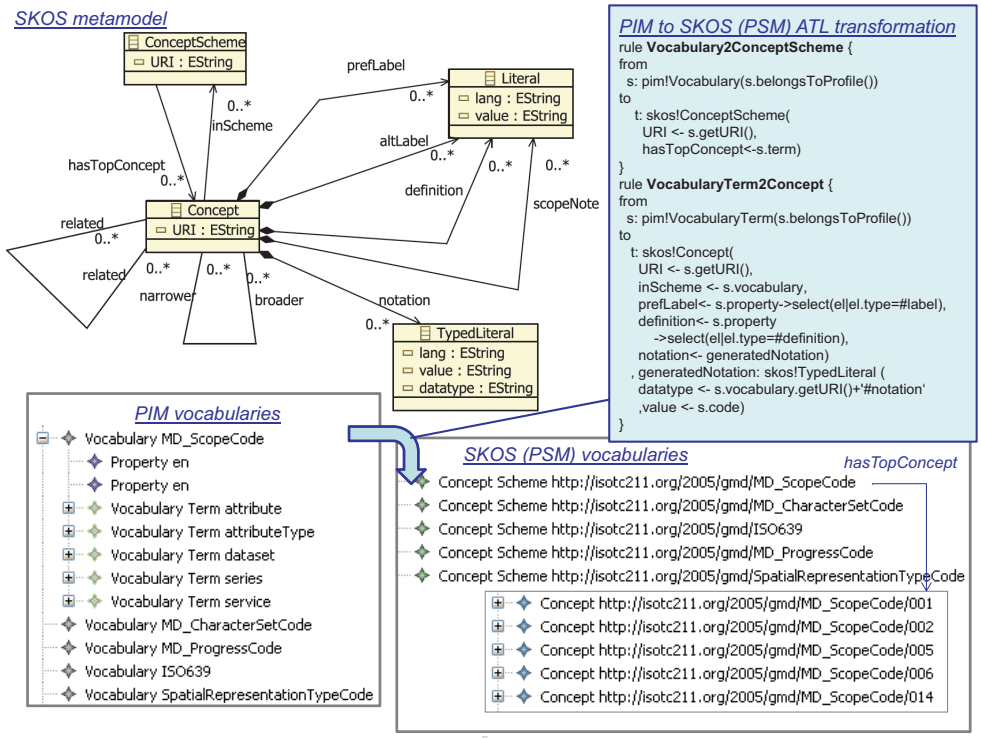


Figure 7. Metamodel of the SKOS language, and transformation of PIM vocabularies into SKOS

tilingual thesauri (ISO 2788 and ISO 5964) did not include a standardized representation format. Currently, these norms already include a XML-based representation format. However, in contrast to RDF-based languages such as SKOS, this XML representation limits its applicability for the publication of controlled vocabularies using Semantic Web technologies. Therefore, given the wide acceptance of this SKOS language for publishing well-acknowledged vocabularies (e.g., GEMET, AGROVOC, and other well-known thesauri used for the annotation of SDI resources are available on the Web in SKOS format), and the availability of multiple software libraries [7] for managing SKOS vocabularies, we decided to select the SKOS language as the domain specific language for representing controlled vocabularies in metadata editors.

Fig. 7 shows the main elements of the metamodel of the SKOS language. A controlled vocabulary consists of a set of concepts (represented by the *Concept* class) grouped in a concept scheme (*ConceptScheme* class). Each concept is linked to its textual representation in different languages (*Literal* class) by means of *prefLabel* and *altLabel* associations, which denote their preferred and alternative textual representations respectively. Additionally, each concept may hold definitions, scope notes, or alternative notations (e.g., ISO 19115 provides a numeric code for each member of a code list or enumeration

[7] `http://www.w3.org/2004/02/skos/wiki/Tools`

that could be represented in the SKOS by means of a typed literal linked by a *notation* association). Finally, it is possible to establish different semantic relations among concepts: the *related* association is used to denote any type of connection between two concepts; the *broader* and *narrower* associations are used to establish hierarchical relations (i.e., one concept is more general than another).

Fig. 7 also shows an example of controlled vocabularies defined as PIM models and how they can be converted by means of ATL transformations into the SKOS language. The figure shows the main transformation rules that map *Vocabularies* to *ConceptSchemes* (*Vocabulary2ConceptScheme* rule), and *VocabularyTerms* to *Concepts* (*VocabularyTerm2Concept* rule).

2.5 Text generation

The last step proposed in the MDA methodology is the transformation of PSM models into textual representations that can be parsed by software libraries to generate the metadata edition forms, in the case of GUI models, and browse the terms from controlled vocabularies, in the case of SKOS models.

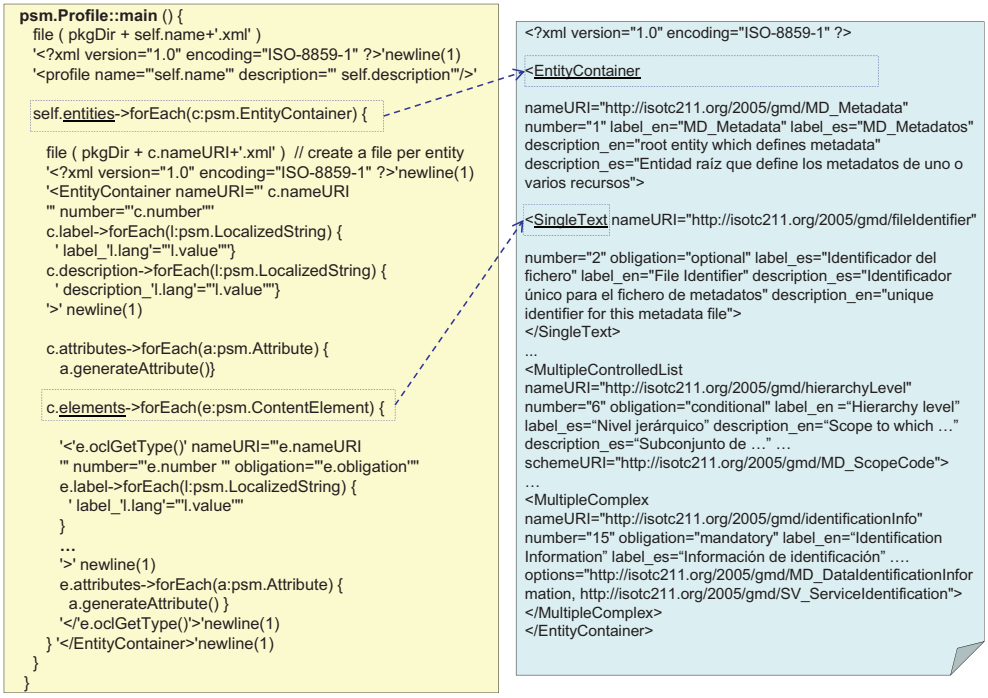


Figure 8. MOFScript transformation from GUI model to XML

An excerpt of the main MOFScript rule for the transformation of a GUI model into a set of XML files (describing the edition forms) is shown in Fig. 8. The transformation is quite immediate as each element from the GUI model has a unique correspondence with a fragment of text. The main contribution

of this MOFScript code is the addition of the low-level details of the XML encoding: it creates separate files for the description of the profile, and each entity contained in the profile. An example of the generated text for the ISO 19115 *MD_Metadata* entity is shown on the right-hand side.

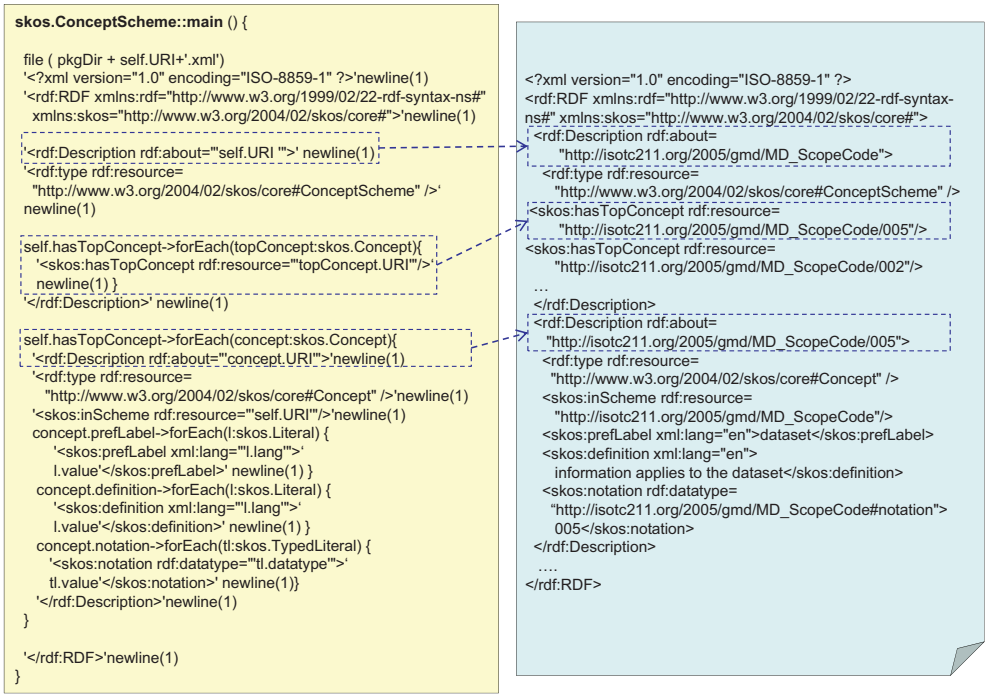


Figure 9. MOFScript transformation from SKOS model to SKOS-RDF

An equivalent MOFScript rule is used to transform the SKOS vocabularies into SKOS-RDF files (see Fig. 9). It generates a separate file for the SKOS-RDF encoding of each *ConceptScheme*. An example of the generated RDF for the *MD_ScopeCode* codelist is shown on the right-hand side of Fig. 9.

Based on this XML and SKOS-RDF encodings, Fig. 10 shows the design of a prototype implementation for a software library that is able to parse and render the graphical components of the edition forms to be integrated in a Java desktop application using the graphical components of the Java Swing library. The *EditionFormRenderer* class is in charge of the dynamic generation of the edition forms according to their XML description. It returns an instance of the *EntityContainer* class, which is a panel rendering the *EntityContainer* class of the PSM metamodel (i.e., the edition form for the root entity in the metadata standard). Apart from providing methods to parse and generate the XML encoding of a metadata entity, the *EntityContainer* class holds a containment association with the *ContentElement* class, which is in turn the Java implementation of the corresponding class in the PSM metamodel. This class and its subclasses are in charge of creating the appropriate GUI widgets for rendering the values and associated information of metadata elements. For

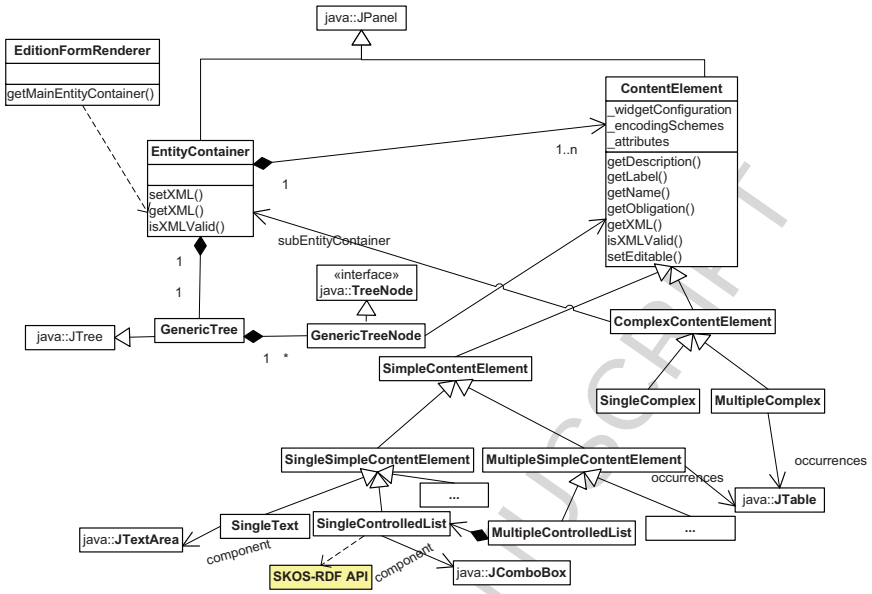


Figure 10. Design of an edition form renderer prototype

instance, the *SingleText* class uses a *JTextArea* component to facilitate the edition of a metadata element filled with free text. In the case of multiple elements, the classes for its GUI edition (e.g., *MultipleComplex*, or derived classes from *MultipleSimpleContentElement*) hold a reference to a *JTable* component to manage multiple occurrences of widgets. Additionally, the classes in charge of editing controlled vocabularies (e.g., *SingleControlledList*) use an SKOS-RDF API to retrieve codelists and enumerations represented in SKOS. This SKOS-RDF API may wrap the access to a generic library for RDF management such as Jena or Sesame, or SKOS-specific libraries such as ThManager [33].

Fig. 11 displays a screenshot of this prototype implementation, enabling the edition of a metadata record that describes a "Natura 2000 sites" dataset. In particular, it shows the edition form of the ISO 19115 *MD_Metadata* entity. A tree is used on the left part of the editing window to browse the elements of an entity through the hierarchical structure of the metadata standard. This is the graphical view of the tree structure, which is modeled with the *GenericTree* and *GenericTreeNode* classes connected with the *EntityContainer* class in Fig. 10. On the right-hand side of the window in Fig. 11, there is a dedicated panel to edit the *hierarchyLevel* element, which must be filled with a value from a controlled vocabulary. Apart from providing a *JTable* component to manage multiple occurrences and a *JComboBox* for editing each occurrence, this panel also provides additional internationalized information such as a definition, special conditions for obligation or examples. Fig. 11 also shows the XML encoding that would be parsed or generated for this metadata element (*getXML* and *setXML* methods of *ContentElement* class in Fig. 10).

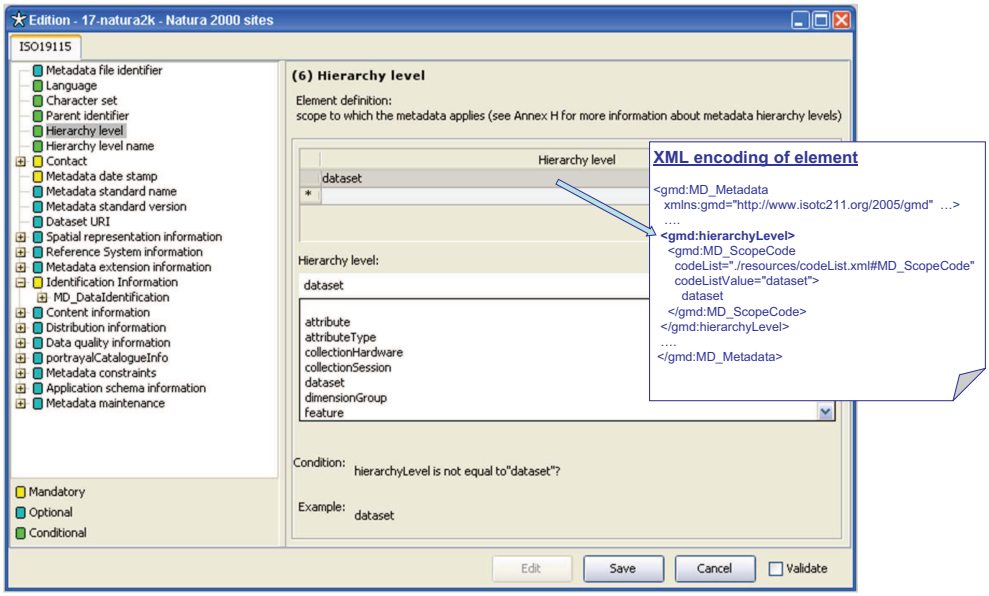


Figure 11. Edition of the *hierarchyLevel* metadata element of ISO 19115

3 Testing the MDA approach in CatMDEdit

CatMDEdit is a Java-based Open Source tool for the semantic annotation of geographical information. Initial versions of this tool were directly implemented around the logical model of an extended version of the CSDGM metadata standard. However, with the increasing requirements for supporting new arising metadata standards and profiles, it was soon acknowledged that this development approach was costly and error prone. Since version 3.7, the development team decided to adopt the MDA approach that has been presented in this paper. The XML describing the GUI and the SKOS-based vocabularies (i.e., the output of our MDA approach), together with the software components to manage them, became the essential core of the architecture of this metadata editor.

Fig. 12 shows an architectural view of CatMDEdit, which follows the *module* viewtype of the *'Views and Beyond'* proposal defined by Clemens et al. [34]. Viewtypes are definitions of the element and relationship types that can be used in a certain view. A *module* viewtype partitions the system in code units (modules) with certain responsibilities. Additionally, viewtypes can have a set of styles, which are specializations of the element and relationship types, and constraints on their use. In particular, focusing on the main features of the application and those modules related to the MDA approach (highlighted in yellow), Fig. 12 provides a hybrid style *module* view with UML notation that combines the following styles: *decomposition*, which shows the structure of modules and submodules; *use*, which indicates functional dependency relations

among modules; and *generalization*, which indicates specialization relations among modules. There are three main modules in the application: a *UserInterface* module aggregating the submodules providing the main functions of the application with GUI interaction; a *KernelLibrary* module providing core support to the *UserInterface* submodules; and a *DataAccess* module in charge of the accessing the data, metadata and vocabularies, which are managed or produced by the application.

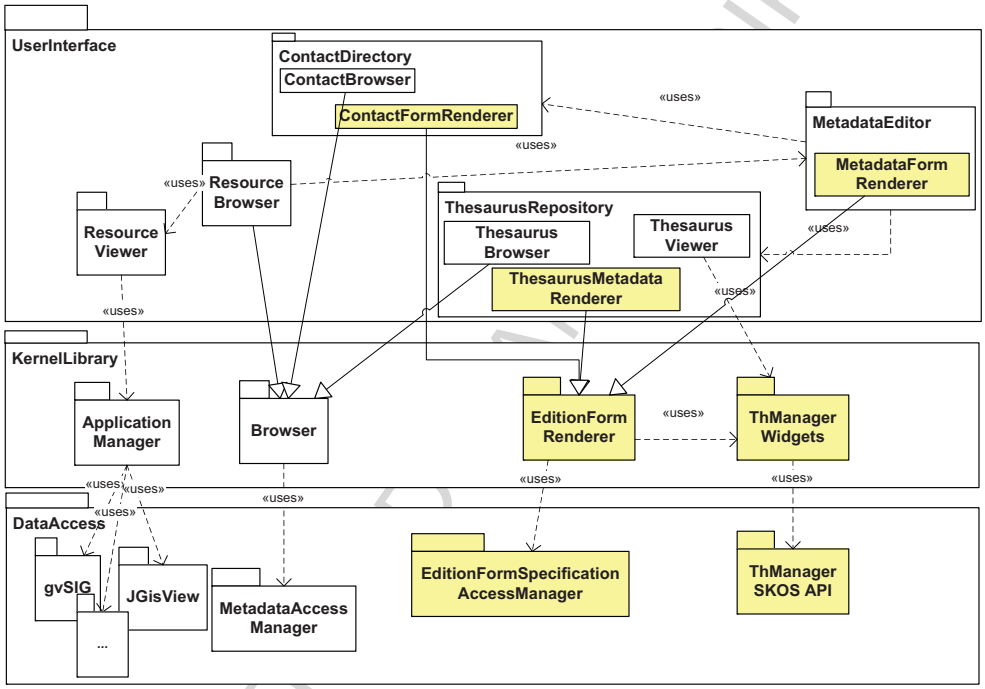


Figure 12. Decomposition-uses-generalization view of CatMDEdit architecture

The *Resource Browser* submodule contains the code of the main browser that enables the navigation of resources organized in different repositories. It is an specialization of the generic *Browser* submodule in the *KerneLibrary*, which uses the *MetadataAccessManager* to have access to metadata repositories. Additionally, it invokes the rest of functions available in the application: a resource viewer, and a metadata editor. The resource viewer (*ResourceViewer* submodule) facilitates the connection with external applications (e.g., GIS tools such as *JGISView* or *gvSIG* represented as submodules in *DataAccess* module) through an application manager (*ApplicationManager* submodule) [35]. The metadata editor (*MetadataEditor* submodule) facilitates the edition of the metadata describing a resource thanks to the integration of a *MetadataFormRenderer*, which facilitates the dynamic configuration of metadata edition forms according to the metadata standard/profile followed by each metadata record.

The *MetadataFormRenderer* submodule is a specialization of the *Edition-*

Table 4
Metadata profiles for the annotation of Geographic Information resources

Profile	Standard	Description	Encoding	Ent	Ele	Prop	Languages
ISO 19115 - Comprehensive	ISO 19115	ISO 19115 comprehensive metadata model as defined in annexes A and B of the ISO 19115 document [3].	ISO 19139	126	345	4702	en,es,fr,pl,pt
ISO 19115 - Core	ISO 19115	Core metadata for geographic datasets as defined in Section 6.5 of ISO 19115 document [3].	ISO 19139	45	115	1318	en,es,fr,pl,pt
WISE	ISO 19115	Metadata profile customized to meet the guidelines for metadata in the implementation of the Water Framework Directive and the development of the "Water Information System for Europe"(WISE). This profile [36] was one of the deliverables of the SDIGER project [37], the first pilot project to test the feasibility of the INSPIRE directive.	ISO 19139	71	156	1856	en,es,fr,pl,pt
INSPIRE-Data	ISO 19115	Metadata profile for describing data customized to meet the requirements set up in the INSPIRE directive [38] through the implementing rules for metadata and their correspondence with the standard ISO 19115 [39].	ISO 19139	57	70	1185	en,es,fr,pl,pt
EURADIN-Featuretype	ISO 19115	Metadata profile created within the scope of the EURADIN project (`https://www.euradin.eu/`), a European project funded by the 7th Research Framework Programme whose main aim is to harmonize specifications about addresses data. This metadata profile for describing feature types is based on the guidelines for metadata in the draft INSPIRE data specifications for addresses.	ISO 19139	48	85	1070	en,es,fr,pl,pt
EURADIN-Dataset	ISO 19115	Metadata profile created within the scope of the EURADIN project (see row above). This metadata profile for describing datasets is based on the guidelines for metadata in the draft INSPIRE data specifications for addresses.	ISO 19139	54	108	1387	en,es,fr,pl,pt
NEM	ISO 19115	Metadata profile recommended by the Spanish National Geographical High Board as a minimum subset of ISO 19115 [40].	ISO 19139	73	156	2621	en,es,fr,pl,pt
PNOA	ISO 19115	Metadata profile created towards the description of datasets produced under the Spanish National Plan for Aerial Ortophotography [41].	ISO 19139	84	142	1952	en,es,fr,pl,pt
SIOSE	ISO 19115	Metadata profile created towards the description of datasets produced for the Spanish Programme for collecting information about land use [42].	ISO 19139	63	115	1719	en,es,fr,pl,pt
DC:CSW	Dublin Core	Application profile that includes all the elements recommended as returnable properties in the CSW protocol binding of OGC Catalog Specifications [27].	RDF	12	37	217	en,es,pl,pt
DC:SDIGER	Dublin Core	Application profile for geographical data mining [43] proposed within the context of the SDIGER project (see row above about WISE). This profile is based on the Dublin Core Spatial Application Profile, a CWA proposed by CEN [44].	RDF	1	54	275	en,es,fr,pl,pt

FormRenderer submodule of *KernelLibrary*, which follows the design already explained in Section 2.5. It renders the GUI of the edition forms defined by means of XML configuration files, which are accessed through the *EditionFormSpecificationAccessManager*. This submodule also uses submodules of *ThManager* [33], an implementation of the technology required for accessing and displaying controlled vocabularies in SKOS-RDF format. Finally, it must be noted that the *MetadataEditor* submodule makes use of other two submodules: *ContactDirectory*, which facilitates an agenda of reusable contact information (e.g., providers or contributors); and *ThesaurusRepository*, which gives access to a browser of available thesauri to select keywords/topics for the classification of resources. These two submodules apply the same pattern used in resource metadata for browsing and rendering the edition forms. Both submodules integrate specializations of the generic *Browser* and *EditionFormRenderer* submodules to navigate and edit contact information and thesaurus metadata in conformance with Dublin Core metadata profiles.

Once we have introduced the architecture of the tool, Table 4 shows the 11 metadata profiles for the annotation of geographic information resources (derived from ISO 19115 and Dublin Core), whose edition forms and vocabularies have been generated using the MDA approach and integrated in CatMDEdit. The table columns present: the name of the profile; the standard from which it derives; a brief description; the encoding rules applied for serialization; the number of entities (*Ent*), elements (*Ele*) and multilingual information (*Prop*) found in the PIM metadata model; and the different languages supported. Additionally, although CatMDEdit is primarily intended for editing geographic metadata, it can be also customized to support new standards and metadata profiles according to different user needs. Table 5 shows a list of 9 metadata profiles that have been included in CatMDEdit for the annotation of other types of resources such as services, feature catalogs, thesauri, contacts or scientific related resources.

In order to validate the benefits from adopting this MDA approach, we have compared it with the development approach of the initial versions of this tool. The inclusion of a new metadata standard in these initial versions implied the development of a new logical model to support the persistence of metadata according to this new standard and a specific GUI to support the edition of metadata records according to this profile. For this comparison between approaches, the following dimensions have been considered: development effort, legibility, modularity, ease of maintenance, and extensibility (similar dimensions have been considered in other MDA works for comparison [46]).

With respect to the development effort, using the old approach the implementation of the ISO 19115 comprehensive profile, which includes all entities in the ISO 19115 standard (126 entities and 345 elements, see Table 4), involved the programming of a minimum number of 252 Java classes. Each standard entity

Table 5
Metadata profiles for the annotation of other types of resources

Profile	Stand.	Description	Encod	Ent	Ele	Prop	Lang
INSPIRE-Services	ISO 19115/ 19119	Metadata profile for describing services that has been customized to meet the requirements set up in the INSPIRE directive [38] through the implementing rules for metadata and their correspondence with the standards ISO 19115 and ISO 19119 [39].	ISO 19139	58	75	1161	en,es, fr,pl, pt
ISO 19110	ISO 19110	Metadata profile based on the model proposed in ISO 19110 for feature catalogs [45], i.e. catalogs describing spatial application schemas.	ISO 19139	23	78	598	en,es, fr,pl, pt
DC: Thesaurus	Dublin Core	Application profile for describing thesauri and other knowledge organization systems (see [33]).	RDF	1	32	132	en,es, fr
DC:Photo	Dublin Core	Application profile for describing georeferenced photographs.	RDF	18	53	256	en,es, pl,pt
DC:Group	Dublin Core	Application profile for describing research groups. This profile is used on the web site of the IAAA research group (`http://iaaa.unizar.es`).	RDF	9	27	328	en,es, pl,pt
DC: Organization	Dublin Core	Application profile used in CatMDEdit for describing contact information of organizations. It includes some elements from the FOAF (Friend Of A Friend) Dublin Core metadata profile.	RDF	9	32	400	en,es, pl,pt
DC:Person	Dublin Core	Application profile used in CatMDEdit for describing contact information of individuals. It includes some elements from the FOAF (Friend Of A Friend) Dublin Core metadata profile.	RDF	12	33	414	en,es, pl,pt
DC:Project	Dublin Core	Application profile for describing scientific projects. This profile is used on the web site of the IAAA research group (`http://iaaa.unizar.es/showProjectList.do?cid=proyectosIAAA.EN`).	RDF	9	27	210	en,es, pl,pt
DC: Publication	Dublin Core	Application profile for describing scientific publications. This profile includes elements from BibTex and other typical fields used in the library context. This profile is used on the web site of the IAAA research group (`http://iaaa.unizar.es/showPublicationList.do?cid=publicacionesIAAA.EN`).	RDF	14	35	242	en,es, pl,pt

required two classes: one class taking part in the logical model of the application and in charge of handling the XML encoding of an entity and its elements; and, at least, another class (usually extending a Java panel) in charge of the graphical view of this entity (i.e., creation of graphical components, layout organization, coordination with the logical model, internationalization, etc.). For a metadata profile including a big number of entities, this development process was error-prone as it required a long phase of testing to verify the correct XML encoding or the no omission of any element. In contrast, using the MDA methodology, the inclusion of a new metadata profile involves no

programming at all. The developer (metadata profile designer) just focuses on introducing the features of this new standard using the domain specific language proposed for the PIM models. Additionally, if the information of the base standard has been previously introduced for other profiles, the developer only needs to indicate which entities and elements belong to this new profile.

About legibility, in the old approach the structure of the metadata standard was scattered in multiple separate classes, both for the logical model and the graphical view. Therefore, in the case of a metadata profile with more than 100 entities, it was difficult to understand the code if the programmer had not followed a strict programming policy (e.g., arrange classes in different packages, or use the same name of entities and elements for Java classes and attributes). On the opposite side, the domain specific language for defining the PIM models is a high-level language, easy to understand because it incorporates the same concepts that are used in the documents describing the standards and metadata profiles.

Regarding modularity, when using the old approach, similar pieces of code were repeated in the classes of the logical model (e.g., the XML encoding of special data types like dates or geometries), or in the classes in charge of the graphical view (e.g., similar patterns to show elements with multiple occurrence, or similar masks of text fields to introduce the same datatypes). Any decision implying a modification in the look and feel or in the serialization of data types required multiple checks in different parts of the application. In contrast, with this new approach each element type is rendered and serialized by a unique software component (see hierarchy of *ContentElements* in Fig. 10).

With respect to the ease of maintenance, any modification introduced by a new version of a metadata standard (and its XML encoding) required new updates in the old approach software: new classes if new entities had been proposed; or the update of existing classes if some new elements had been added or removed. On the opposite side, using the new approach we just need to introduce the modifications in the PIM models and generate automatically a new version of XML files describing the edition forms. The software code does not require any update or compiling.

Finally, regarding extensibility and as already discussed, the support of new metadata profiles in the old approach required a hard effort of coding. In contrast, the incorporation of new metadata profiles based on supported standards is quite immediate: the metadata expert just needs to create a new PIM model picking the elements of the profile and modify, if necessary, some of their features (e.g., labels, obligation, or condition). But even in the worst case, integrating a metadata profile of a new standard not considered so far, the effort is focused on a very high-level level. A typical scenario introduced by the support of a new standard can be the definition of a new data type,

e.g. a time period or a 3-dimension geometry. A possible solution to solve this problem is the definition in the PIM model of a complex data type element (see *ComplexDataTypeElement* in Fig. 3), whose *complexDataType* is another entity aggregating simple data type elements (e.g., the begin date and the end date of a time period). However, in order to promote the ergonomics and ease of use of the metadata editor, the developers can consider the creation of new GUI widgets specialized on these data types. Although this last alternative requires a higher development effort, this effort is located in well-identified parts of our MDA approach: define a new data type in the *DataTypeCode* codelist of the PIM metamodel; define a new type of *ContentElement* in the GUI PSM metamodel; program a new GUI widget for the edition form renderer; and add small modifications in ATL and MOFScript transformation rules. Additionally, this effort is quickly recovered if new metadata profiles and standards have the same requirement.

4 Related work

Given the increasing importance of geographic metadata, numerous software packages (dedicated tools or plug-ins in GIS tools) have appeared during the last decade for the creation of metadata. The Wisconsin Land Information Clearinghouse (WiscLINC) [10] and the Federal Geographic Data Committee [47] provide detailed reviews of edition tools based on CSDGM (the old North American standard) and ISO 19115 metadata standards. Without being exhaustive, the first review, last updated in 2006, reports 24 metadata edition tools (21 compliant with CSDGM and 6 compliant with ISO 19115), 10 metadata utilities, and 4 metadata servers. The second review, made between 2007 and 2009, is focused on a thorough analysis of eleven ISO 19115 metadata editors.

The purpose of this section is not to make another review of metadata editors, but to analyze some of them from a metamodeling perspective and how this aspect may affect their flexibility to satisfy new requirements. Tables 6 and 7 review 13 metadata editors with respect to the metamodeling layer they are based on (see metamodeling layers in Section 2.1). Being an M1-layer based tool (6 tools in Table 6) means that the logical model of this tool has a direct correspondence with the standard it supports. On the contrary, being an M2-layer based tool (7 tools in Table 7) means that the tool is able to understand a metamodel and parse different metadata standards expressed in terms of this metamodel. Additionally, apart from showing the layer category, these tables show other features: distributor, platform (and interface), year of first and last known releases, standards supported, and an explanation for being included within the M1 or M2 category. The tables only report metadata editors about which we have clear evidence from their source code or from

Table 6
Metadata editors based on M1 layer

Tool	Distributor	Platform	First-last	Standards	Metamodeling layer explanation
ArcCatalog v9.3	ESRI	Desktop - Windows	1999-2009	partial ISO 19115, CSDGM	According to Vermeij [48], building a custom environment in ArcCatalog requires four components: a logical model for metadata content, an implementation schema, an editor, and one or more stylesheets. Custom editors must be deployed as COM (Component Object Model) components and registered under the "Metadata Editors" component category using the ArcGIS component manager. Programmed in Visual Basic or other COM compliant languages, editors must create a specific GUI to support the edition of new or existing metadata elements.
KaR Metadata Toolkit	British Geological Survey	Desktop - Windows	2004-2004	partial ISO 19115	Microsoft Access application using a relational database model where each table supports the persistence of a different class in the UML specification of ISO 19115.
MetaGenie v2.0	Association for Geographic Information, UK	Desktop - multiplatform	2004-2006	partial ISO 19115, UK Gemini	The desktop version of this Java-based tool enables the creation of metadata records in compliance with UK Gemini metadata model. Metadata records can be saved as XML files or within a database [49]. For the database storage, the UK Gemini model is implemented as a relational schema with a fixed structure.
Metalite 1.7.5	U.S. Geological Survey, United Nations Environment Program	Desktop - Windows	1998-2000	partial CSDGM	Visual basic application using an Access database as storage device. A core set of CSDGM elements is implemented as columns of a single table.
MetaMaker	National Biological Information Infrastructure	Desktop - Windows	1999-1999	CSDGM, CSDGM_NBII	Microsoft Access application using a relational database model where each table represents a section (or subsection) of the CSDGM metadata standard. Appendix E of the user manual [50] describes the names, types, and sizes of fields in the MetaMaker database.
Tkme 2.9.20	U.S. Geological Survey	Desktop - Windows	1998-2006	CSDGM	The metadata elements of CSDGM are hard-coded within the C source code of this application [51].

literature about their design approach. This is the reason to exclude three tools from the study (MetaD, Preludio, terraCatalog) that are included in the review of ISO metadata editors [47].

As it can be observed in Table 6, only one M1-layer based tool (ArcCatalog) from the CSDGM era has been able to evolve and support partially the complexity of ISO 19115. Obviously, one of the reasons for this lack of maintenance may be the change of company objectives in a ten-year frame. But probably, the most important reason can be found in the approval of ISO

Table 7
Metadata editors based on M2 layer

Tool	Distributor	Platform	First-last	Standards	Metamodeling layer explanation
CatMDEdit 4.6.5	IAAA, GeoSpatium-Lab	Desktop - multi-platform	2001-2010	ISO 19115, Dublin Core	Although initial versions followed an M1 approach for the support of CSDGM and ISO19115, since version 3.7 (2005) the tool incorporates an MDA approach for the development of edition interfaces.
Geonetwork v2.6.4	FAO-UN, WFP-UN, UNEP	Web-based	2005-2011	ISO 19115, ISO 19119, CSDGM, Dublin Core	The configuration of a new metadata standard in this Web-based tool using the J2EE technologies requires the inclusion of an XML Schema for the syntax, and different XML and XSL files to adjust the edition interface, the internationalization to different languages and the indexing and searching capabilities [52, Section 4.10].
ISO Metadata Editor v4.1	Instituto Nacional de Técnica Aeroespacial	Desktop - multi-platform	2005-2007	ISO 19115	This Java based tool enables the configuration of metadata standards and profiles, whose syntax is stored as CSV (Comma Separated Value) files [53]. The edition interface for a new metadata record is guided by these configuration files.
M3Cat v1.6	Intelec Geomatics Inc.	Web-based	2001-2007	ISO 19115, CSDGM, CSDGM_NBII, GILS	M3Cat uses the concept of metamodel to provide support for multiple standards [54]. The definitions of metadata elements are stored in a RDBMS (Access, Oracle or Postgress). The Web-based interface of the tool (programmed with ASP) creates dynamically the edition forms following the standard structure.
MDWeb 2	IRD, LIRMM, Cemagref	Web-based	2006-2009	ISO 19115	The design and implementation of this Web-based (PHP) tool draws on the concept of metamodeling [55]. The genericity of the metadata database resides in the originality of the relational schema that describes a structure at a meta level coupled with a standard structure.
MetaManager 4	Compusult	Web-based	1998-2004	CSDGM, ISO 19115	Because one of the initial objectives of MetaManager was to allow data suppliers to publish their internal and ad-hoc metadata holdings as a standardized node in compliance with the Z3950 protocol for discovery and information retrieval, this tool uses a metamodeling approach to define the elements of a metadata standard and the mappings to the relational model used for ad-hoc metadata storage [56]. Developed in C++, the standard definition and the mappings can be stored using Access, SQL Server or Oracle as RDBMS.
Pangloss product suite	Drexel University	Desktop - multi-platform	2004-2004	ISO 19115, any OWL ontology	Pangloss is a suite of Java products that facilitates the development of metadata expressed in OWL (Web Ontology Language) [57]. The Pangloss metadata instance tool edits metadata instances based on OWL Metadata Schemas.

19115 as international standard in 2003 and the convergence of national initiatives such as CSDGM towards this international standard [58]. The high costs involved in adapting old metadata editors to the complexity of this international standard (more than 400 elements organized in hierarchical and recursive structures) can explain the decision of abandoning the maintenance. The WiscLINC review [10] identified 23 historical tools (or utilities) compliant with CSDGM whose maintenance had been stopped. Even ArcCatalog, the M1-layer tool providing support for both CSDGM and ISO 19115, only offers a partial coverage of the ISO standard (since 2002, ArcCatalog covers the core set of ISO 19115 metadata elements for describing geographic datasets [59]). The logical model of this tool supports the CSDGM metadata elements, some ESRI-defined elements, and a partial set of ISO 19115. CSDGM, ESRI and partial ISO 19115 content coexist in parallel in the metadata XML documents maintained by ArcCatalog v9.3 [60].

In general, it can be stated that the life-time of the analyzed M1-layer tools is potentially shorter than the M2-layer ones (average life-time of 4.6 years in Table 6 versus 5.5 years in Table 7, which will probably increase in the following years). The higher costs involved in upgrading M1-layer based tools to satisfy new standard requirements can be observed even for those tools initially developed to support ISO 19115. KaR Metadata Toolkit is no longer available, and MetaGenie only covers a partial set of ISO 19115. The long history of draft versions of ISO 19115 until the final approval in 2003, the corrigendum of this standard in 2006, the appearance of different ISO 19115 profiles, and the continuous flow of XML encoding proposals before the publication of ISO 19139 have made the upgrading of M1-layer based tools extremely complicated.

As a conclusion, there is a clear tendency towards the implementation of metadata editors using the metamodeling concept. CatMDEdit, the tool described in Section 3, is not an exception and works directly at the M2 layer. As well as other M2-layer tools, its user manual explains how to customize the description of edition forms for a specific metadata profile. From this perspective, it does not differ too much from similar M2-layer tools. However, the real advantages of applying our proposed MDA approach are in the steps previous to the generation of the GUI description files. Thanks to this approach, a metadata designer can benefit from the tools and infrastructure provided by MDA. Using textual or graphical notations such as TCS [30] or GMF (Graphical Modeling Framework [8]), the metadata designer can define PIM models using a high-level domain specific language which is very close to the concepts expressed in the documents delivered by standardization bodies. Once these PIM models have been defined, it is possible to generate automatically the configuration files that are required by a specific metadata editor software. To our knowledge after reviewing the literature of tools in Table 7, none has applied

[8] `http://www.eclipse.org/modeling/gmp`

the MDA methodology to transform a high-level definition of the metadata standard into a set of low-level configuration files, which are usually hard to understand for a non-expert developer.

Additionally, the adaptation of the MDA approach proposed in Section 2 can be easily applied to any of the tools described in Table 7. Apart from having demonstrated its applicability for generating the configuration files (and SKOS vocabularies) required by CatMDEdit, the proposed framework (domain specific languages and transformations proposed in Section 2) can be adapted to generate the input required by other tools. In most cases, only the MOFScript transformation rules should be modified to generate an appropriate configuration. All the effort devoted to the definition of PIM models can be fully reused if we decided to utilize a new metadata editor software. Besides, using this approach, the internationalization of the metadata standard concepts (e.g, labels, definitions, or conditions in multiple languages) is managed at a high and centralized level instead of handling this issue at a low technical level (e.g., by means of multiple instances of external property files).

5 Conclusions

This paper has presented the guidelines to apply an MDA approach for the development of annotation tools, which can be customized to different metadata standards and profiles with minimum effort. Applying this approach, experts in metadata standards (without any special programming skills) can focus their efforts on the definition of new metadata models using a domain specific language (the one used to define PIM models), whose abstraction level is close to the way of expressing metadata standards in the documents provided by standardization bodies. After this high-level definition, the PIM models are automatically transformed into the PSM models and configuration files required by the specific metadata editor that has been chosen as the final platform for annotating resources.

The feasibility of these guidelines for applying MDA has been tested with CatMDEdit, an open source metadata editor supporting multiple metadata standards and profiles. Following this MDA approach, 11 metadata profiles derived from ISO 19115 and Dublin Core have been incorporated in CatMDEdit for the annotation of geographic information resources. Anyway, this approach is also extensible for the annotation of other types of resources. As it has been shown in Section 3, many other different metadata profiles have been integrated in the tool for the annotation of a wide range of resources such as photographs, services, thesauri or scientific publications. Additionally, using qualitative criteria (development effort, legibility, modularity, ease of maintenance and extensibility), it has been proven that the adoption of this MDA approach outperforms the initial versions of CatMDEdit, which did not follow this development philosophy. The initial versions of this tool, as well as

other M1-layer based tools, were implemented around a logical model (directly hardcoding the structure of the metadata standards) and were very sensitive to any upgrade of metadata standards and profiles.

Another issue that must be taken into account is that this MDA approach is not only applicable for the development of desktop applications like CatMDEdit. For instance, this MDA approach has been successfully applied for the development of a Web application for editing metadata for geographic web services [61]. The whole infrastructure for the processing of PIM and GUI-based PSM models was reused. The only thing created specifically was a renderer for Web edition forms, which was customized to the specific features of GWT (Google Web Toolkit) technology. A similar adaptation of the approach could be also applied to improve the efficiency of other metadata editors using a metamodeling perspective (i.e., M2-layer based tools where supported metadata standards can be configured). The domain specific languages and transformations presented in this paper would provide the users of these tools with the appropriate MDA infrastructure to focus the development effort on defining reusable models of metadata standards, and alleviating them from the technical details of configuration files.

As future work, we will work on the evaluation of different concrete syntax notations (visual and textual) for the definition of PIM models. In order to test these different syntaxes, we plan to integrate them in a new release of CatMDEdit. This would allow final users to define metadata standards and profiles in a more expressive way, and hide the details of model-to-model and model-to-text transformations (which would be executed automatically). Additionally, instead of using the Ecore metamodeling language for defining the PIM metamodel, we will study the use of other languages with more expressive power (e.g., PVS or Eiffel [62]), which could help us to check automatically the conformance to some special constraints of metadata profiles (e.g., a profile can only impose a more stringent obligation on existing metadata elements), not fully expressed with Ecore.

References

[1] D. Nebert (Ed.), Developing Spatial Data Infrastructures: The SDI Cookbook v.2.0, Global Spatial Data Infrastructure (GSDI), `http://www.gsdi.org`, 2004.

[2] Federal Geographic Data Committee, Content Standard for Digital Geospatial Metadata, version 2.0, Document FGDC-STD-001-1998, Metadata Ad Hoc Working Group (1998).

[3] International Organization for Standardization, Geographic information – Metadata, ISO 19115:2003 (2003).

[4] Dublin Core Metadata Initiative (DCMI) Usage Board, DCMI Metadata Terms, `http://dublincore.org/documents/2008/01/14/dcmi-terms/` (2008).

[5] International Organization for Standardization, Geographic information – Metadata – Part 2: Extensions for imagery and gridded data, ISO 19115-2:2009 (2009).

[6] Federal Geographic Data Committee, Content Standard for Digital Geospatial Metadata: Extensions for Remote Sensing Metadata, Public review draft, Standards Working Group (2002).

[7] W. Swoboda, F. Kruse, R. Nikolai, W. Kazakos, D. Nyhuis, H. Rousselle, The UDK Approach: the 4th Generation of an Environmental Data Catalogue Introduced in Austria and Germany, in: Proc. of 3rd IEEE META-DATA Conference, Bethesda, Maryland, 1999.

[8] M. Wilde, M. A. Lange, H. Pundt, N. Ostländer, K. Janowicz, An environmental metadata profile for the EU project MEDIS, in: 17th International Conference on Informatics for Environmental Protection (EnviroInfo), 2003.

[9] R. Tolosana-Calasanz, J. Nogueras-Iso, R. Béjar, P. R. Muro-Medrano, F. J. Zarazaga-Soria, Semantic interoperability based on Dublin Core hierarchical one-to-one mappings, International Journal of Metadata, Semantics and Ontologies (IJMS&O) 1 (3) (2006) 183–188.

[10] Wisconsin Land Information Clearinghouse, Metadata tools for geospatial data, `http://www.sco.wisc.edu/wisclinc/metatool/` (2006).

[11] Information technology – Metadata registries (MDR) – Part 3: Registry metamodel and basic attributes, ISO/IEC 11179-3:2003, International Organization for Standardization (2003).

[12] R. Grangel, C. Metral, A. Cutting-Decelle, J. Bourey, R. Young., Ontology based communications through model driven tools: feasibility of the MDA Approach in Urban Engineering, in: Ontologies for Urban Development, Vol. 61 of Studies in Computational Intelligence, Springer, 2007, pp. 181–196.

[13] Object Management Group (OMG), Meta Object Facility (MOF) Specification v1.4, OMG Document formal/02-04-03, `http://www.omg.org/cgi-bin/apps/doc?formal/02-04-03.pdf` (April 2002).

[14] T. Kutzner, A. Donaubauer, Critical remarks on the use of conceptual schemas in geospatial data modelling - a schema translation perspective, in: Bridging the Geographic Information Sciences, Lecture Notes in Geoinformation and Cartography, Springer, 2012, pp. 43–60.

[15] P. Staub, H. R. Gnägi, A. Morf, Semantic interoperability through the definition of conceptual model transformations, Transactions in GIS 12 (2) (2008) 193–207.

[16] C. Mayr, U. Zdun, S. Dustdar, View-based model-driven architecture for enhancing maintainability of data access services, Data & Knowledge Engineering 70 (9) (2011) 794–819.

[17] D. Djurić, D. Gašević, V. Devedžić, Ontology Modeling and MDA, Journal on Object Technology 4 (1) (2005) 109–128.

[18] J. Sztipanovits, G. Karsai, Model-integrated computing, IEEE Computer April (1997) 110–112.

[19] Object Management Group (OMG), MDA Guide version 1.0.1, Document Number: omg/2003-06-01 (2003).

[20] S. Kelly, J.-P. Tolvanen, Domain-Specific Modeling: Enabling Full Code Generation, Wiley-IEEE Computer Society Press, 2008.

[21] G. Nordstrom, J. Sztipanovits, G. Karsai, A. Ledeczi, Metamodeling – Rapid Design and Evolution of Domain-Specific Modeling Environments, in: IEEE Int. Conf. and Workshop on the Engineering of Computer Based Systems (ECBS '99), 1999, pp. 68–74.

[22] A. Miles, J. R. Pérez-Agüera, SKOS: Simple knowledge organisation for the web, Cataloging and Classification Quarterly 43 (3-4) (2007) 69–83.

[23] D. Steinberg, F. Budinsky, M. Paternostro, E. Merks, EMF: Eclipse Modeling Framework, 2nd Edition, Addison-Wesley, 2008.

[24] C. Driver, S. Reilly, . Linehan, V. Cahill, S. Clarke, Managing embedded systems complexity with aspect-oriented model-driven engineering, ACM Trans. Embed. Comput. Syst. 10 (2) (2011) 21:1–21:26.

[25] F. Jouault, I. Kurtev, Transforming models with ATL, in: Proc. of the Model Transformations in Practice Workshop at MoDELS 2005, Montego Bay, Jamaica, 2005.

[26] J. Oldevik, MOFScript User guide, version 0.9 (MOFScript v 1.4.0), `http://www.eclipse.org/gmt/mofscript/doc/MOFScript-User-Guide-0.9.pdf` (2011).

[27] D. Nebert, A. Whiteside, P. Vretanos, OpenGIS – Catalogue Services Specification (version: 2.0.2), OGC 07-006r1, Open Geospatial Consortium Inc. (2007).

[28] International Organization for Standardization, Geographic information – Metadata – XML schema implementation, ISO/TS 19139:2007 (2007).

[29] A. Powell, M. Nilsson, A. Naeve, P. Johnston, T. Baker, DCMI Abstract Model, Dublin Core Metadata Initiative (DCMI), `http://dublincore.org/documents/2007/06/04/abstract-model/` (2007).

[30] F. Jouault, J. Bézivin, I. Kurtev, TCS: a DSL for the Specification of Textual Concrete Syntaxes in Model Engineering, in: GPCE'06: Proc. of the 5th int. conf. on Generative programming and Component Engineering, 2006.

[31] J. Lacasta, J. Nogueras-Iso, R. Béjar, P. R. Muro-Medrano, F. J. Zarazaga-Soria, A Web Ontology Service to facilitate interoperability within a Spatial Data Infrastructure: applicability to discovery, Data & Knowledge Engineering 63 (3) (2007) 945–969.

[32] E. Cuellar, G. Lozano, R. Morris, SwingML, Swing Markup Language Specification, http://swingml.sourceforge.net/files2/Specification.pdf (2004).

[33] J. Lacasta, J. Nogueras-Iso, J. López-Pellicer, P. M. Medrano, F. J. Zarazaga-Soria, ThManager: An Open Source Tool for creating and visualizing SKOS, Information Technology and Libraries (ITAL) 26 (3) (2007) 39–51.

[34] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord, J. Stafford, Documenting Software Architectures: Views and Beyond, 2nd Edition, SEI Series in Software Engineering, Addison-Wesley, 2011.

[35] J. Nogueras-Iso, J. Barrera, F. Gracia-Crespo, S. Laiglesia, P. R. Muro-Medrano, Integrating catalog and GIS tools: access to resources from CatMDEdit thanks to gvSIG, in: 4th Int. gvSIG conf.: moving forward together, Valencia, 2008.

[36] J. Nogueras-Iso, P. R. Muro-Medrano, F. J. Zarazaga-Soria, SDIGER – WFD Metadata Profile, v0.3, SDIGER project, http://sdiger.unizar.es/public_docs/wfd_profile_v0.3.pdf (2005).

[37] F. J. Zarazaga-Soria, J. Nogueras-Iso, M. A. Latre, A. Rodríguez, E. López, P. Vivas, P. Muro-Medrano, Providing SDI Services in a Cross-Border Scenario: the SDIGER Project Use Case, in: Research and Theory in Advancing Spatial Data Infrastructure Concepts, ESRI Press, 2007, pp. 113–126.

[38] European Commission, Directive 2007/2/EC of the European Parliament and of the Council of 14 March 2007 establishing an Infrastructure for Spatial Information in the European Community (INSPIRE), http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2007:108:0001:0014:EN:PDF (2007).

[39] European Commission, Joint Research Centre, INSPIRE Metadata Implementing Rules: Technical Guidelines based on EN ISO 19115 and EN ISO 19119, v. 1.0., http://inspire.jrc.ec.europa.eu/reports/ImplementingRules/metadata/MD_IR_and_ISO_20081219.pdf (2008).

[40] D. Ballari, A. S. Maganto, J. Nogueras-Iso, A. Rodríguez, M. A. Bernabé, Experiences in the use of an ISO19115 profile within the framework of the Spanish SDI, in: GSDI-9 Conference Proceedings, no. 10, Santiago, Chile, 2006.

[41] E. Domenech Tofiño, S. Molina Blázquez, C. García González, G. Villa Alcázar, A. Arozarena Villar, Producción de mosaicos por hojas del MTN50 en el Plan Nacional de Ortofotografía Aérea para la visualización y distribución vía Web, in: V Jornadas Técnicas de la IDE de España JIDEE2008, 2008.

[42] Instituto Geográfico Nacional. Equipo Técnico Nacional SIOSE, Sistema de Información de Ocupación del Suelo en España (SIOSE). Manual de fotointerpretación. Anexo II: Manual de metadatos SIOSE. Versión 1.3, http://www.ign.es/siose/Documentacion/Guia_Tecnica_SIOSE/AnexoII_ManualdeMetadatos/080609_ANEXOII_Manual_Metadatos_SIOSE_v1.3.pdf (2008).

[43] J. Nogueras-Iso, P. R. Muro-Medrano, F. J. Zarazaga-Soria, R. Rioja, SDIGER – Dublin Core Metadata Application Profile for geographical data mining, v0.5, SDIGER project, http://sdiger.unizar.es/public_docs/spatial_dc_profile_v0.5.pdf (2005).

[44] European Standardization Committee (CEN), Dublin Core Spatial Application Profile, CWA 14858, CEN/ISSS Workshop – Metadata for Multimedia Information – Dublin Core, http://www.cen.eu/cenorm/sectors/sectors/isss/cen+workshop+agreements/meta-data+dc.asp (2003).

[45] International Organization for Standardization, Geographic information – Methodology for feature cataloguing, ISO 19110:2005 (2005).

[46] J. L. Cánovas Izquierdo, O. Sánchez Ramón, J. Sánchez Cuadrado, J. García Molina, Utilidad de las transformaciones modelo-modelo en la generación de código, in: XII Jornadas de Ingeniería del Software y Bases de Datos (JISBD'07), 2007.

[47] Federal Geographic Data Committee, ISO Metadata Editor Review, http://www.fgdc.gov/metadata/iso-metadata-editor-review (April 2010).

[48] B. Vermeij, Implementing European Metadata Using ArcCatalog, ArcUser July-September, http://www.esri.com/news/arcuser/0701/metadata.html.

[49] Association for Geographic Information, Gigateway MetaGenie User Guide v2.0, http://www.gigateway.org.uk/metadata/pdf/MetaGenie_Guide.pdf (2006).

[50] D. M. Schneider, User's Manual: National Biological Information Infastructure MetaMaker Version 2.30 – Getting Started and Navigation Tips, U.S. Geological Survey, Midcontinent Ecological Science Center, Fort Collins, Colorado. Published by the Upper Midwest Environmental Sciences Center, La Crosse, Wisconsin, 1999.

[51] P. N. Schweitzer, Tkme: Another editor for formal metadata, http://geology.usgs.gov/tools/metadata/tools/doc/tkme.html (2009).

[52] GeoNetwork opensource Community, GeoNetwork Developer Manual, release 2.6.4, http://geonetwork-opensource.org/manuals/2.6.4/developer/GeoNetworkDeveloperManual.pdf (2011).

[53] A. Amaro Cormenzana, A. Domínguez Barroso, E. Prado Ortega, Definición de Perfiles y creación de ficheros XML de metadatos para imágenes de Teledetección, in: Actas de Jornadas Técnicas de la Infraestructura de Datos Espaciales de España (JIDEE'05), Madrid, 2005.

[54] S. Kéna-Cohen, Y. Bédard, M3Cat, Limiting Aggravation in Geospatial Metadata Cataloguing, in: Conférence GIS2001, Vancouver, Canada, 2001.

[55] J.-C. Desconnets, T. Libourel, S. Clerc, B. Granouillac, Cataloguing for distribution of environmental resources, in: 10th AGILE Int. Conf. on Geographic Information Science, Aalborg, Denmark, 2007.

[56] B. O'Rourke, Enabling Z39.50 Services for Relational Databases, White paper, Compusult Limited, Mount Pearl, NF, Canada, http://www.metadatamanager.com/metadatamanager/enable_z39.50_services.PDF (1999).

[57] L. Bermudez, M. Piasecki, Metadata Community Profiles for the Semantic Web, Geoinformatica 10 (2006) 159–176.

[58] Federal Geographic Data Committee, Preparing for International Metadata - North American Profile of ISO 19115: Geographic Information – Metadata, http://www.fgdc.gov/metadata/documents/csdgm-to-nap-transition-guidance.pdf (2010).

[59] ESRI, Metadata and GIS, An ESRI White Paper, http://www.esri.com/library/whitepapers/pdfs/metadata-and-gis.pdf (October 2002).

[60] ESRI, Metadata standards and the ArcGIS metadata format, ArcGIS Desktop Help 9.3., http://webhelp.esri.com/arcgisdesktop/9.3/index.cfm?TopicName=Metadata_standards_and_the_ArcGIS_metadata_format (28 March 2008).

[61] J. Nogueras-Iso, J. Barrera, A. F. Rodríguez, R. Recio, C. Laborda, F. Zarazaga-Soria, Development and deployment of a services catalog in compliance with the INSPIRE metadata implementing rules, in: SDI Convergence: Research, Emerging Trends, and Critical Assessment, The Netherlands Geodetic Commission, Delft, the Netherlands, 2009, pp. 21–34.

[62] R. F. Paige, P. J. Brooke, J. S. Ostroff, Metamodel-based model conformance and multiview consistency checking, ACM Trans. Softw. Eng. Methodol. 16 (3) (2007) 1–49.

JAVIER NOGUERAS-ISO holds MS and PhD degrees in Computer Science from the University of Zaragoza (Spain). After working for the Economic and Social Committee of the European Communities (Brussels) in 1998, he started his research at the Advanced Information Systems Laboratory of the Computer Science and Systems Engineering Department of the University of Zaragoza. Currently, he is an Associate Professor of Computer Science at that University. Additionally, he completed a postdoctoral stay at the Institute of Environment and Sustainability of the Joint Research Centre (Ispra, Italy) in 2005. His research interests are focused on Spatial Data Infrastructures and Geographic Information Retrieval. Within this context, he has co-authored numerous publications in books, journals or conference proceedings; and has collaborated in several R+D projects.

MIGUEL ÁNGEL LATRE holds a MS degree in Computing from the University of Zaragoza (Spain) since October 1999. He is currently a tenured Assistant Professor at the Department of Computer Science and Systems Engineering at the same university. He is working with the Advanced Information Systems Laboratory where he has been involved in several R&D projects related to the software engineering aspects of Geographic Information Systems and its application to the Environmental field. He is now working to complete his PhD degree in Computer Science.

RUBÉN BÉJAR holds MS and PhD degrees in Computing from the University of Zaragoza (Spain). He is currently a tenured Assistant Professor at the Department of Computer Science and Systems Engineering at the same university. He is working with the Advanced Information Systems Laboratory where he has been involved in several R\&D projects related to the software engineering aspects of Geographic Information Systems and has several publications in that area.

PEDRO R. MURO-MEDRANO holds MS and PhD degrees in Industrial Engineering from the University of Zaragoza (Spain). He has worked in the private industry for 2 years and has hold different visiting research positions at the Carnegie Mellon University's Robotics Institute (Pittsburgh, PA, USA), the University of Maryland (College Park, MD, USA) and the US National Institutes of Health (Bethesda, MD, USA). He has 25 years of experience with R\&D activities in software development and engineering and he is Professor of Computer Science at the University of Zaragoza. He is co-author of numerous national and international publications in books, journals and conference proceedings. Currently, he is the head of the Advanced Information Systems Laboratory at the Computer Science and Systems Engineering Department and the Aragón Institute for Engineering Research from the University of Zaragoza. His research interests include Spatial Data Infrastructures, Geographic Information Systems, Environmental Information systems and Remote Sensing.

F. JAVIER ZARAZAGA-SORIA holds MS degree in computer science from the Technical University of Valencia (Spain) and PhD degree in Computer Science from the University of Zaragoza (Spain). He did his MS Thesis at the Middlesex University's Road Safety Engineering Laboratory, London (UK). In September 1994 he began to work at the Advanced Information Systems Laboratory. He has been Assistant Professor at the Computer Science and Systems Engineering Department of the University of Zaragoza since November 1996 to June 2003. Since June 2003 he is Associate Professor at that University. He is

co-author of numerous national and international publications in books, journals and conference proceedings. He has participated in several R\&D projects with companies and/or Public Administrations. His research interests include Information Retrieval and Ontologies in the context of Spatial Data Infrastructures.