CdmCL, a Specific Textual Constraint Language for Common Data Model

Ahmed Ahmed, Paola Vallejo, Mickaël Kerboeuf, Jean-Philippe Babau

Lab-STICC / UBO / UEB, Brest, France
alahmed4ever@yahoo.com,{vallejoco,kerboeuf,babau}@univ-brest.fr

Abstract. Common Data Model is an abstract data model for scientific datasets that can be constrained by OCL. To hide complexity of OCL, *CdmCL* is proposed as a specific textual constraint language for CDM. *CdmCL* is based on the CDM structure and results in a set of constraint categories. *CdmCL* provides a user-friendly front end in order to define constraints which are subsequently translated to OCL. The conformity tool is based on an existing OCL checker integrated in EMF. *CdmCL* is experimented on the OceanSITES standard.

Keywords: OCL, common data model, conformity, constraint generation

1 Introduction

To improve interoperability, scientific dataset modeling follows standards like Unidata's Common Data Model (CDM) [1]. Since CDM is a general purpose model, scientists use specific standards like OceanSITES [2] for specific data modeling. A standard defines a set of additional constraints, classically expressed in a natural language. To check if CDM data conforms to a standard, like OceanSITES, a code-oriented checker is used classically. Thus, the constraints are not formalized and a modification in the standard results in a manual modification of the code.

To handle the problems of a code-centric approach, constraints can be implemented using Object Constraints Language (OCL) [3]. OCL is a formal language that significantly improves the clarity of models and makes them more precise [4]. But unfortunately, it is difficult to write correct OCL statement as many OCL constraints results in inaccurate and erroneous constraints [5], [6].

In this paper, we propose a textual domain specific constraint language *CdmCL* to reduce the complexity of handling OCL syntax. *CdmCL* is dedicated for scientific data standards. It is based on a set of constraints categories deduced from the CDM structure. Then, OCL constraints are generated and used by an OCL checker integrated in the Eclipse Modeling Framework (EMF) [7].

The paper is organized as follows. In the first section, CDM is introduced. Then OceanSITES is presented as a motivating example. *CdmCL* and the conformity tool generation are then presented before to be evaluated on OceanSITES. Before to conclude, related works are discussed.

2 Common Data Model

Unidata's Common Data Model (CDM) is an abstract data model for scientific datasets. It is based on three layers, data access layer, coordinate system layer and scientific feature type layer. Our work focuses on data access layer also called syntactic layer that handles data modeling part. The complete data model and detailed description is given in [1]. The main classes (see Figure 1) are:

- *DataSet*: a file, such as NetCDF file, characterized by a file name (*location*).
- *Group*: a container for *dimensions*, *attributes*, *variables* and nested *subGroups*.
- *Dimension*: the array shape of a *Variable*, characterized by a *name* and a *length*;
- *Variable*: a container for data, characterized by a *name*, a *dataType*, a set of *dimensions* that define its array shape, and optionally a set of *attributes*.
- *Attribute*: a metadata to characterize a *Variable* or a *Group*, characterized by a *name*, a *dataType* and a *value*.

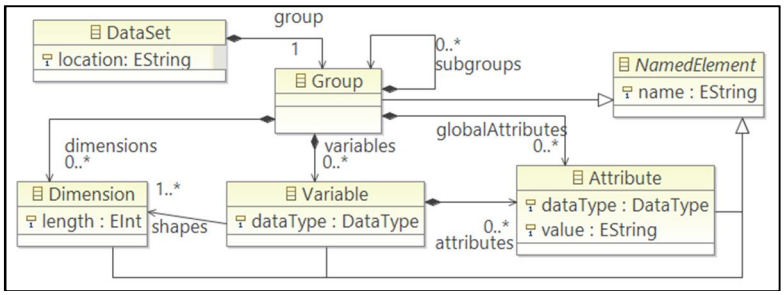

Fig. 1. An excerpt of the Common Data Model ecore (CDM.ecore)

3 OceanSITES

OceanSITES [2] is a worldwide system for gathering and measuring scientific data especially for time series sites, called ocean reference stations. It conforms to the CDM model but with some constraints. The OceanSITES User Manual holds around 30 pages of constraints expressed in natural language. These constraints are of different forms: naming conventions, possible attribute values, constraints on dimension length and many others. As example, a DataSet should hold instances of Dimension called *TIME*, *LATITUDE*, *LONGITUDE*, instances of Attributes called *data_type* and *format_version*, and an instance of Variable called *TIME*. The variable *TIME* should be of *double* datatype. Figure 2 illustrates a small excerpt of OceanSITES standard from the manual to the left and a small excerpt of CDM data respecting the OceanSITES standard to the right.

To check the data conformity to OceanSITES, a Java tool already exists [8]. Since constraints are not formalized and the tool is hand-coded, there is no guaranty on checking. Furthermore, for each different data format, a particular tool should be developed. To avoid constraint edition ambiguity and to reduce conformity tool development, we present now the *CdmCL* language.

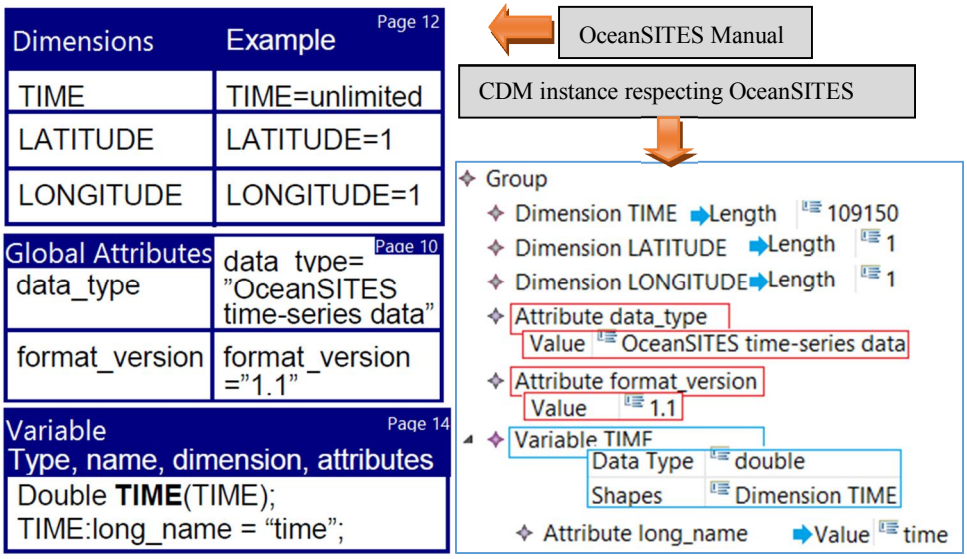

Fig. 2. Excerpts of OceanSITES manual and CDM instance

4 *CdmCL* Language

4.1 Concept References

This work focuses on the automatic generation of OCL constraints from *CdmCL*. On the one hand, the *CdmCL* front end needs to be human readable and close to the classical standard. Therefore, Xtext has been used to define the textual grammar. On the other hand, each CdmCL concept has a semantic expressed using OCL.

In standards, most of the constraints are related to a specific instance of a named CDM concept (Variable, Dimension and Attribute): "the attribute named *data_type* can hold either the value *OceanSITES metadata,* or *OceanSITES profile data***ð**; "the dimension of the variable named *TIME* is the dimension *TIME***ð**. Thus, *CdmCL* is structured by three abstract classes DimensionConstraint, VariableConstraint and AttributeConstraint (see figure 3). Then, to express a constraint related to a specific instance, *CdmCL* follows the three different scenarios, defining nine concrete concepts:

- Constraint is applied to a specific CDM concept referenced by its *name*: *aDimensionConstraint*, *aVariableConstraint* or *anAttributeConstraint.*

- Constraint is applied to items whose *name* matches a specific regular expression: *TemplateDimensionRegex*, *TemplateVariableRegex*, or *TemplateAttributeRegex*.

- Constraint is applied to a set of items, characterized by a set of names: *TemplateDimensionList*, *TemplateVariableList* or *TemplateAttributeList*.

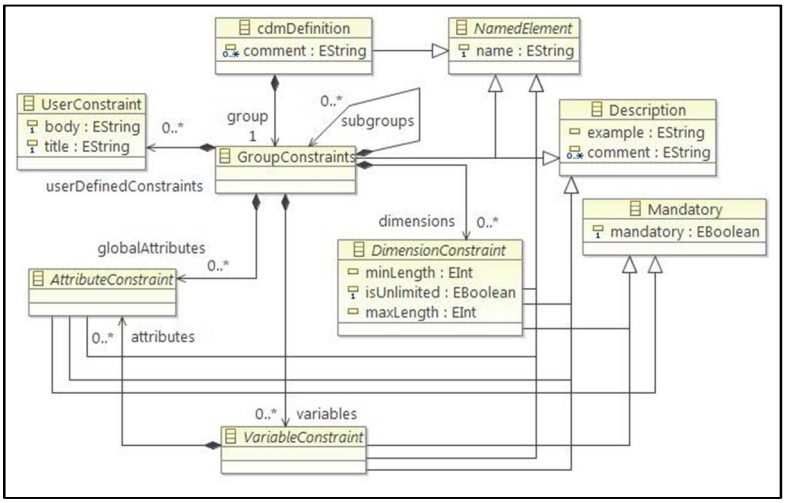

Fig. 3. An excerpt of the ecore *CdmCL* model (CdmCL.ecore)

4.2 Common Constraints

The following section introduces the constraints that are common between *DimensionConstraint*, *VariableConstraint* and *AttributeConstraint*.

Mandatory: This constraint verifies that the *name* of the related concept exists. For the template list concept, an extra *Boolean* attribute *or* permits to indicate whether one of the items of the list is mandatory or all the items are mandatory. As an example, figure 4 presents the *CdmCL* expressions and the corresponding OCL statements for *aDimension* called *DEPTH* (a single dimension) and a *TemplateDimensionList* (coordinate dimension list holding *LATITUDE* and *LONGITUDE*).
The values between parentheses permit the definition of dimension length and are explained in the next section. Figure 5 presents a CDM instance to the left respecting the OceanSITES standard, thus the instance is valid, whereas the other instance is not valid because the mandatory dimension *LONGITUDE* and the mandatory variable *TIME* are missing, thus the OCL constraints checkMandatory_dimension_LONGITUDE and checkMandatory_variables_TIME are violated.

Repetition: This constraint checks that a *name* of a concept is never repeated.

Format: This constraint verifies that names of a set of concept have a specific format, either uppercase, lowercase or matches a specific regular expression.

*User Defined Constraints***:** This concept increases the flexibility of the language, by allowing the user to enter manually an OCL statement. However, it is the user's responsibility to verify the correctness of the OCL statement regarding the CDM metamodel. These constraints are defined in the context of *Group*.

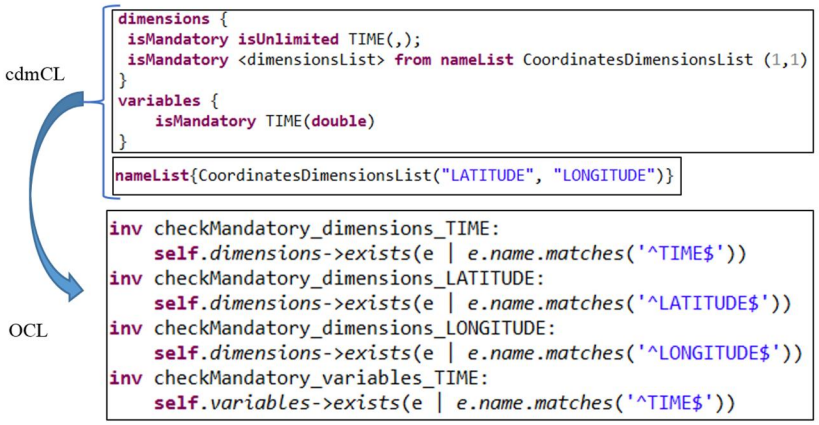

Fig. 4. Mandatory constraint example with *CdmCL* and OCL correspondence

Fig. 5. CDM data respecting and not respecting OceanSITES standards

4.3 Concept Related Constraints

In addition, other constraints are related to specific concepts.

- **Dimensions length constraints**: The *length* value for a dimension can be a limited or unlimited (any positive value). For the limited length, it can be a specific value, a value in a range, greater than or equal to a specific value, smaller than or equal. To achieve these constraint objectives, *dimensionConstraint* concept has two Integer attributes called *minLength* and *maxLength* and one Boolean attribute called *IsUnlimited* (see table 1).

case	*Min*	*Max*	*isUnlimited*	*Description*
1	-	-	true	Length is unlimited, i.e. any positive value
2	x	x	false	Length is equal to x, a specific value
3	x	y	false	Length is between x and y, y>x>0
4	x	-1	false	Length is greater than x
5	-1	y	false	Length is smaller than y
6	-1	-1	false	No constraint on length

Table 1. Dimension *length* constraints categories

Figure 6 illustrates the OCL statement generated for *aDimension TIME* with *unlimited* dimension length who's *CdmCL* is given in figure 4. The concept related

constraints are built all in the same way. The first part of the OCL constraint (exists, select) defines the context of the specific concept (here the TIME Dimension). Then, the second part expresses the constraint itself (forAll).

```
inv checkDimensionLength_TIME:
self.dimensions->exists(e | e.name.matches('^TIME$')) implies
self.dimensions->select(e | e.name.matches('^TIME$'))->forAll( length >= 1 )
```

Fig. 6. Dimension length OCL constraint

- **Variable shape constraint**: A variable is characterized by a set of shapes i.e. a set of dimensions (see figure 1). A variable can have a shape of the same name as the variable's name. For example a variable named *TIME* is associated with a dimension named *TIME*. Moreover, a variable can be associated with a dimension or a set of dimensions. For example, a variable named *TIME_QC* is associated with dimension named *TIME*. To accomplish this type of verification, we have two concepts *SimilarDimensionConstraint* and/or a set of *PredefinedShape* concept for a *VariableConstraint.* On one hand *SimilarDimensionConstraint* concept allow the generation of an OCL invariant that verifies that a variable is associated with a dimension of same variable's name. On the other hand *PredefinedShape* concept permits to verify that a variable is associated with any preexisting dimensions. Figure 7 illustrates the previous discussion and presents the *CdmCL* representation along with the OCL to be generated.
- **VariableConstraint and AttributeConstraint DataType**: This constraint verifies that the variables and the attributes can have any data type from a list of data types.
- **Attribute Value Constraint**: A constraint on the *value* of an *Attribute*, the possible categories are given in table 2.

Case	Constraint
Unique value	The value should have this and only this value
Regular Expr	The value should match the regular expression
Range	A range of values between min and max, similar to dimension length
List	The value can be any value from a list of values.
Standard	The value matches a regex given by a standard (e.g. ISO8601).

Table 2. Attribute's value constraints categories

5 Conformity tool and experiment

Based on *CdmCL*, we developed a conformity checker for netCDF[1] files which conform to CDM metamodel. The tool architecture is shown in figure 8. It is developed in Java and based on the Eclipse Modeling Framework (EMF). First, the tool transforms the *CdmCL* model into OCL statements in a separated file. This part

[1] NetCDF (Network Common Data Form) is a CDM compliant file serialization format

(*CdmCL2OCL*) uses Xtext API, and the library we developed for the transformation of *CdmCL* expressions to OCL constraints. Then, the tool transforms a NetCDF file into an instance of CDM model. This part (*nc2CDM*) is simply based on the Java NetCDF API and on the CDM Java API (provided by EMF). Finally, the tool checks the conformity of the CDM file to the CDM metamodel enriched with the generated OCL file, and indicates whether it is valid or not. This part is based on the integrated OCL checker of the EMF modeling tool.

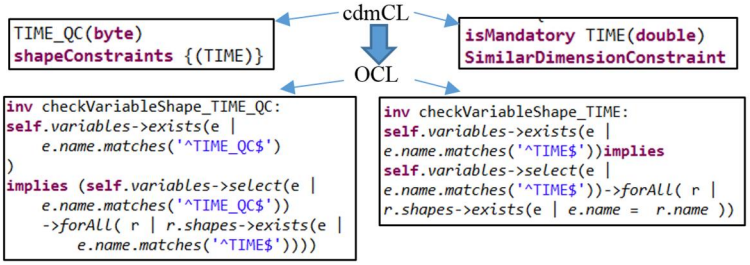

Fig. 7. CdmCL representation for two variables one with *PredefinedShape* concept and the other with *SimilarDimensionConstraint* concept and their corresponding OCL

The tool has been tested on OceanSITES. Due to the lack of space we express only the OceanSITES global attributes standard given in figure 2 in *CdmCL*. The *CdmCL* shown in figure 9 indicates that a global attribute named *data_type* is mandatory, of type *string* and has any of the values presented by the list *dataTypeGlobalList*. This list hold the values *OceanSITES metadata*, *OceanSITES profile data*, *OceanSITES time-series data* or *OceanSITES trajectory data*. Furthermore, it indicates that a global attribute named *format_version*, of type *string* and should hold the value "*1.1*". The *CdmCL* expressions and the generated OCL after using CdmCL2OCL tool are given in figure 9.

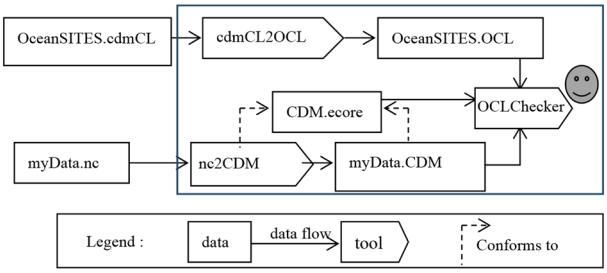

Fig. 8. *CdmCL* conformity tool architecture

The generated OCL file is then used with the CDM model to check the conformity of an input netCDF file. If the input file does not respect the constraints, a message indicates that the input file does not conform to CDM model with the set of OCL violated constraints. Figure 10 presents an excerpt of CDM data converted from a netcdf file. These data are validated with this generated OCL. It is seen that the left instance is valid whereas the right instance is invalid because the global attribute *format_version* value constraint is violated.

```
globalAttributes {
   isMandatory data_Type (string)
   value in valueList dataTypeGlobalList
   example "OCEANSITES time-series data";
   isMandatory format_version (string)
   value = "1.1"
   -- "format version" example "1.1";
          …       …       …
```

```
valueList{
dataTypeGlobalList (
    "OceanSITES metadata",
    "OceanSITES profile data",
    "OceanSITES time-series data",
    "OceanSITES trajectory data")}
```

```
inv checkMandatory_globalAttributes_data_Type:
    self.globalAttributes->exists(e | e.name.matches('^data_Type$'))
inv checkMandatory_globalAttributes_format_version:
    self.globalAttributes->exists(e | e.name.matches('^format_version$'))
inv checkDataType_globalAttributes_data_Type:
    self.globalAttributes->exists(e | e.name.matches('^data_Type$'))
    implies self.globalAttributes->select(e | e.name.matches('^data_Type$'))
              ->forAll( e | e.dataType = DataType::string  )
inv checkDataType_globalAttributes_format_version:
    self.globalAttributes->exists(e | e.name.matches('^format_version$'))
    implies self.globalAttributes->select(e | e.name.matches('^format_version$'))
           ->forAll( e | e.dataType = DataType::string )
inv checkGlobalAttributeValue_data_Type:
    self.globalAttributes->exists(e | e.name.matches('^data_Type$'))
    implies
    self.globalAttributes->select(e | e.name.matches('^data_Type$'))->forAll(g |
        g.value.matches('^OceanSITES metadata$')  or
        g.value.matches('^OceanSITES profile data$')  or
        g.value.matches('^OceanSITES time-series data$')  or
        g.value.matches('^OceanSITES trajectory data$'))
inv checkGlobalAttributeValue_format_version:
    self.globalAttributes->exists(e | e.name.matches('^format_version$'))
    implies self.globalAttributes->select(e | e.name.matches('^format_version$'))
              ->forAll(g | g.value.matches('^1.1$'))
```

Fig. 9. *CdmCL* and generated OCL for OceanSITES

Figure 10 illustrates that the right version is invalid because *format_version* constraints was violated, but in reality this global attribute can have the value of 1.2 and even 1.3 since OceanSITES standard has been evaluated to the new version 1.3 with backward compatibility. Therefore by just modifying the CdmCL *format_version* value to hold the values (1.1, 1.2 or 1.3), we can have the new standard OCL representation without any professional interference and the instance will be valid with respect to OceanSITES version 1.3.

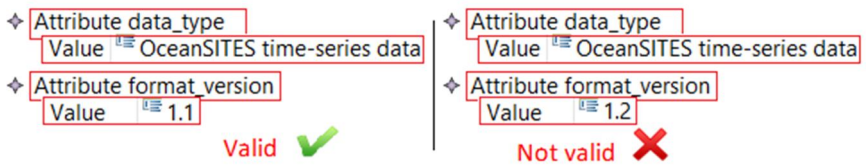

Fig. 10. Valid and Invalid cdm instance with respect to OceanSITES standard version 1.1

Experimenting *CdmCL* on OceanSITES, it is observed that more than 90% percent of the constraints are achieved in the *CdmCL* except the constraints that are related to multiple CDM concepts at the same time. For example, a constraint verifies the existence of either a variable named *TIME* with attribute named *QC_indicator* or a variable named *TIME_QC*.

```
inv userDefinedConstraint_oneTimeQC:
(not self.variables->exists(e | e.name.matches('^TIME_QC$')))
and self.variables->select(e | e.name.matches('^TIME$'))
.attributes->exists(e | e.name.matches('^QC_indicator$'))
xor self.variables->exists(e | e.name.matches('^TIME_QC$'))
and ( not self.variables->select(e | e.name.matches('^TIME$'))
    .attributes->exists(e | e.name.matches('^QC_indicator$')))
```

Fig. 11. User defined OCL constraints

For *CdmCL* syntax readability, the two global attributes of OceanSITES given in figure 2 are represented by approximately eight lines in *CdmCL* expressions and generate around 30 lines of OCL statements. One page of textual constraints from the standard is expressed by around 25 lines in *CdmCL* and generates around 300 lines of OCL.

The language CdmCL was introduced to OceanSITES users (standard reader) and they confirm the expressiveness and the readability of *CdmCL*.

6 Related Works

Several studies are proposed to support OCL integration on modeling processes. The Dresden OCL Toolkit [9] proposes an OCL library and the recent version provides an OCL-to-Java Code Generator. USE (UML-based Specification Environment) [10] is a tool to facilitate the validation of UML models with OCL constraints. USE supports consistency, independence of constraints, and relevance of constraints analysis. These OCL tools are complementary to the proposed approach and can be used to facilitate the management of the generated OCL constraints.

In the domain of OCL generation, [11] propose OCL automatic generation from UML class diagrams. The approach aims at simplifying the process of generation of OCL statements. The approach involves expressing constraints by a class diagram. In the addressed domain, most of the constraints are related to specific instances. Following this approach would result in too many classes (one per instance constrained), and the class diagram syntax is far from the scientific standard edition.

In [12], the authors propose to convert natural language expressions to the equivalent OCL statements. The expressions are constraints and pre/post conditions related to UML diagrams. OCL generation is based on the Semantic Business Vocabulary and Rules language (SBVR) to avoid inconsistencies. With *CdmCL*, we prefer to define a DSL because, in a small and well identified domain, it promotes the development of efficient and accurate solutions [13]. As a DSL, CdmCL disambiguates scientific standard edition. And in addition, it is close enough to the natural language

so that its *use* does not require technical skills on OCL. However, the *creation* of a DSL usually requires both domain knowledge and language development expertise [13]. But, the production cost of *CdmCL* is low since it relies directly on OCL semantics, while hiding unnecessary OCL features. Thus, translation to OCL may be directly done without considering SBVR intermediate level.

7 Conclusion

This paper proposes a domain specific constraint language for CDM. The language structure is based on CDM structure and results in a set of constraint categories. These categories permit to define constraints in a human readable language and serves in the automatic generation of OCL. The approach hides the complexity of writing OCL manually and increases the productivity by generating a large number of OCL statements for few lines written in *CdmCL*. Furthermore, any standard edition and conformity checking can be done easily without OCL and programming interference.

In perspective, we are working on developing domain specific operators to automate scientific data migration from a standard to another one.

8 References

1. *Unidata, Common Data Model (CDM).* Version 4. http://www.unidata.ucar.edu/software/thredds/current/netcdf-java/CDM/
2. *OceanSITES User's Manual*, Version 1.2. http://www.oceansites.org/
3. Object Management Group: *Object Constraint Language (OCL) Specification*, Version, 2.4, http://www.omg.org/spec/OCL/2.3.1/
4. Meyer B., *Object-Oriented Software Construction,* International Series in Computer Science, Second Edition, Prentice-Hall (1997)
5. Linehan M., *Ontologies and rules in Business Models*, in the 11h IEEE Enterprise Distributed Object Conference (EDOC), 149-156, 2008.
6. Wahler M., *Using Patterns to Develop Consistent Design Constraints*, PhD Thesis, ETH Zurich, Switzerland, 2008.
7. *Eclipse Modeling Framework* (EMF). http://www.eclipse.org/modeling/emf/
8. *OceanSITES file format checker*. http://www.coriolis.eu.org/Observing-the-ocean/Observing-system-networks/OceanSITES/Access-to-data.
9. Schütze L., Wilke C., Demut B., *Tool-Supported Step-By-Step Debugging for the Object Constraint Language,* In OCL@MODELS 2013, 2013.
10. Gogolla M., et al. *USE: A UML-Based Specification Environment for Validating UML and OCL*, Science of Computer Programming, Vol. 69, 27-34, 2007.
11. Li Tan, Zongyuan Yang and Jinkui Xie, *OCL Constraints Automatic Generation for UML Class Diagram*, IEEE International Conference on Software Engineering and Service Sciences (ICSESS) 392 – 395, 2010.
12. Bajwa I.S., Bordbar B. and Lee M.G., *OCL Constraints Generation from Natural Language Specification*, in the 14th IEEE Enterprise Distributed Object Conference (EDOC), 204-213, 2010.
13. Mernik M., Heering J.and Sloane A. *When and how to develop domain-specific languages* in ACM Computing Surveys, 37(4), 316-344, 2005.