1st International Workshop on Algebraic Foundations for OCL and Applications (WAFOCA'06)

Valencia, March 22nd, 2006
Technical University of Valencia
Information Systems and Computation Department

Workshop Description

WAFOCA'06 is the 1st International Workshop on Algebraic Foundations for OCL and Applications. WAFOCA'06 is funded by the DYNAMICA project. Its aim is to bring theoreticians, developers and practitioners together to discuss different formal approaches to apply OCL as both constraint and query language in the Model-Driven Engineering field.

Aims and Scope

Model-Driven Development has evolved to the Model-Driven Engineering field, where not only design and code generation tasks are involved, but also traceability, model management, metamodeling issues, model interchange and persistence, etc. To fulfil these tasks, model transformations and model queries are relevant tasks that must be solved. In the MDA context, they are dealt from an open-standard point of view. The standard Query/Views/Transformations (QVT) provides support for both transformations and queries, where OCL seems to be the best choice for defining model queries. OCL is a textual language that is defined as a standard "add-on" to the UML standard. It is used to define constraints and queries on UML models, allowing the definition of more precise and more useful models. It can also be used to provide support for metamodeling. Despite its many advantages, while there is wide acceptance for UML design in CASE tools, OCL lacks a well-suited technological support.

In this workshop, we focus on the development of tools that provide formal support for OCL and on its applications as well. Special interest is given to algebraic formalisms or graph grammar-based approaches, and applications. This workshop solicits research contributions and experience reports having an impact on the study of formal approaches for OCL and its applications. Paper submission will be done by **invitation**, whereas the attendance to the workshop is free. Topics of interest include (but are not restricted to):

- executable specifications;
- algebraic or graph grammar-based approaches for OCL;
- experience reports on development of tools or proposals;
- experience reports on usage of tools or proposals:
  - definition of metrics,
  - support for precise modelling/metamodelling,
  - support model transformations,
  - extensions for behaviour specifications,
  - support for model testing and simulation,
  - ontology development and validation for the Semantic Web

Important Dates

Deadline for abstract submissions: **Marh 3, 2006**
Deadline for full paper submissions: **Marh 10, 2006**
Notification of authors: March 15, 2006
Deadline for final version: March 20, 2006
Workshop date: March 22, 2006

Invited Speaker

Dr. Reiko Heckel (http://www.cs.le.ac.uk/people/rh122/)

Paper Submission and Workshop Proceedings

Paper submission will be done by **invitation**.

Submitted papers should be 8-15 pages in length. Papers must be submitted by email as postscript or pdf documents to José Á. Carsí (pcarsi@dsic.upv.es). The organizing committee will review the submissions and select papers according to their relevance and interest for the discussions that will take place at the workshop. Authors are encouraged to use the Springer LNCS format (http://www.springer.de/comp/lncs/authors.html) for their papers.

The official language of the workshop is **English**. All accepted papers will be published by the publishing services of the Technical University of Valencia as Technical Report (ISBN is being negotiated). Online proceedings will be available at the WAFOCA'06 web site after the workshop. It is mandatory, that at least one author of each accepted paper attends the workshop and presents the paper there.

Location

WAFOCA 2006 will be held in Valencia, Spain in March 22nd, 2006. More information for venue, registration and accomodation will be available after the notification deadline.
For information about Valencia, see among others:

- the city web page: http://www.ayto-valencia.es/
- tourist information about the city: http://www.turisvalencia.es/index.aspx
- information about the Arts and Science City in Valencia: http://www.cac.es/
- official web site of the FALLES festival: http://www.fallas.com/

Workshop Organizers

- Isidro Ramos (iramos@dsic.upv.es), Technical University of Valencia
- José Á. Carsí (pcarsi@dsic.upv.es), Technical University of Valencia
- Artur Boronat aboronat@dsic.upv.es) , Technical University of Valencia

ISSI Ingeniería del Software y Sistemas de Información

Proyecto DYNAMICA

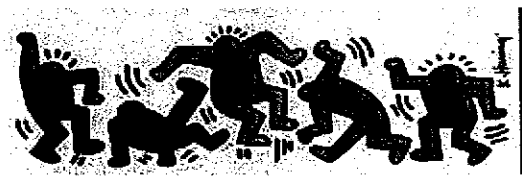
Isidro Ramos, José Á. Carsí and Artur Boronat (eds.)

Algebraic Foundations for OCL and Applications

First International Workshop (WAFOCA'06)
Valencia, Spain, March 22nd, 2006

DSIC

Information Systems and Computation Department
Technical University Of Valencia



Proyecto DYNAMICA

Preface

March 2006

This Technical Report comprises the final versions of all technical papers presented at the ***1st International Workshop Algebraic Foundations for OCL and Applications (WAFOCA'06)*** held in Valencia (Spain), March 22nd, 2006. WAFOCA 2006 is funded by the DYNAMICA project. Its aim was to bring theoreticians, developers and practitioners together to discuss different formal approaches to apply OCL as both constraint and query language in the Model-Driven Engineering field.

Model-Driven Development has evolved to the Model-Driven Engineering field, where not only design and code generation tasks are involved, but also traceability, model management, metamodeling issues, model interchange, persistence, etc. To fulfill these tasks, model transformations and model queries are relevant tasks that must be solved. In the MDA context, they are dealt from an open-standard point of view. The standard Query/Views/Transformations (QVT) provides support for both transformations and queries, where OCL seems to be the best choice for defining model queries. OCL is a textual language that is defined as a standard "add-on" to the UML standard. It is used to define constraints and queries on UML models, allowing the definition of more precise and useful models. It can also be used to provide support for metamodeling. Despite its many advantages, while there is wide acceptance for UML design in CASE tools, OCL lacks a well-suited formal and technological support. In this workshop, we focus on the development of tools that provide formal support for OCL and on its applications as well. Special interest is given to algebraic formalisms or graph grammar-based approaches, and applications.

The workshop was organized in two main sessions: a first session on ***OCL Applications***, which was constituted by three technical papers; and a second session on ***OCL Tools and Demos***, which was constituted by three technical papers that were complemented with tool demos.

Finally, we would like to express our sincere gratitude to Antonio Vallecillo for the his help in some organizing tasks of the workshop and to Reiko Heckel for coming to the workshop as invited speaker. Last but not least, the authors of all submitted papers are gratefully thanked for having made this workshop possible.

Isidro Ramos, José Á. Carsí and Artur Boronat

Index

OCL^2: Using OCL in the Formal Definition of OCL Expression Measures

Luis Reynoso[1], Marcela Genero[2] and Mario Piattini[2]

[1] Department of Computer Science, University of Comahue,
Buenos Aires 1400, 8300, Neuquén, Argentina
lreynoso@uncoma.edu.ar
[2] Alarcos Research Group, Department of Computer Science,
University of Castilla-La Mancha, Ciudad Real, Spain
{Marcela.Genero, Mario.Piattini}@uclm.es

Abstract. Within the Object Oriented software measurement a lot of measures have proliferated during the last decades. However, most of the existent measures differ in the degree of formality used in their definition. If the measure definition is not precise enough, for instance when natural language is used, misinterpretations and misunderstanding of their intent can be introduced. Therefore, this situation may flaw the interpretation of experimental findings or even can difficult in building adequate measures extraction tools. This paper carefully describes how a set of measures that capture the structural properties of expressions specified with the Object Constraint Language (OCL) were precisely defined upon the OCL metamodel. So, we used OCL twice (OCL^2): as a language for defining measures and as a target to capture its core concepts through measures.

1 Introduction

A plethora of Object Oriented (OO) measures have been proposed from the nineties till nowadays. Intrinsically to any measure is its definition along with their theoretical or empirical validation. However, before addressing if the measures are theoretically or empirically valid it is more important to know how the measures are defined, i.e. what we intent to measure. As Baroni et al. [2] commented many difficulties arise when unclear or imprecise measure definition is applied:

- experimental findings can be misunderstood due to the fact may be not clear what the measure really capture,
- measures extraction tools can arrive to different results,
- and experiments replication is hampered.

Most of the existent measures differ in the degree of formality used in their definition. Two extreme approaches were used, informal and rigorous definitions. However none of these approaches had been widely accepted due to the fact of a weakness in some aspect. On one hand, measures using a informal definition, such as measures defined in natural languages, may be ambiguously defined, and everybody knows that the use of this practice introduce misinterpretations and misunderstanding. On the other ex-

treme, in a rigorous approach, authors have used a combination of set theory and simple algebra to express their measures [6], [10] but this approach was not popular due the majority of members of OO community may not have the required background to understand the underpinning of the complex mathematical formalism used.

An example of how the use of natural language in a measure definition introduces ambiguity is considered in [2] which use the "Number of Times a Class is Reused" measure of Lorenz et al. [13]. This measure is defined as the number of references to a class. We agree with Baroni that is not clear "what references are and how the metric should be computed, and many questions arise as: Should internal and external references be counted? Should references be considered in different modules, packages or subsystem? Does the inheritance relationship count as a reference?".

An important contribution to solve the problem of the formality degree in the measure definition is to use the Object Constraint Language [15] upon a design metamodel.

As part of our research work during the last two years, we have proposed a set of measures for OCL expressions, trying to find indicators for the understandability and modifiability of OCL expressions [17]. When we decided to formally define them we considered that the usage of OCL for that purpose could have two advantages:

- The first is that OCL itself is precisely defined through metamodeling facilities, as an instance of the meta-metamodel of the OMG Meta Object Facility (MOF) [19], and the measure definition can be suitably placed at the same level (the M2 level) as the OCL definition.
- The second is that a same language, OCL, is used as a formal language to define the UML and OCL semantics (at M2 Level) and is used by modelers for defining constraints on their models (at M1 Level). In fact the OCL language was claimed as a language easy to use and easy to learn, and to be easily grasped by anybody familiar with OO modeling [7], [12], [21], [22]. So, the familiarity of this language can make the definition of our measures more modeler-friendly.

The approach of defining measures for OCL expression using the OCL metamodel and the OCL language as the formal language allow unambiguous definition. OCL was previously used by the QUASAR (QUantitative Approaches on Software Engineering And Reengineering) Research Group [3],[4],[5] for defining measures. However, in this case the metamodel upon which OCL was used was the UML metamodel. In our case, OCL is used as a language for defining measures for OCL expressions upon the OCL metamodel. This is why we called OCL^2 to the work presented in this paper.

In our approach when we compute the value of a specific measure we represent an OCL expression as an instantiation of OCL metaclasses. The instantiation has the shape of a tree, an abstract syntax tree (*ast*). We traverse the dynamic hierarchical structure (the *ast*) and meanwhile visit every elements in the tree, we evaluate if each element of the tree is meaningful for the measure we want to compute. If it does, the measure is incremented otherwise it remains as it is. Due the methodology for defining the measures is almost the same for the whole set of defined measures in [17] we will only explain in this paper the definition of two measures, the Number of Attributes referred through self (NAS) and the Number of Comparison operation (NCO).

This paper is structured as follow: section 2 briefly explains the OCL metamodel and some of its metaclasses used to explain an instantiation, section 3 describes an *ast*

sample for an OCL expression and section 4 explains the implemented strategy using a visitor pattern for traversing the *ast* and we show the formal definition of the measures NAS and NCO. Finally, section 5 concludes the paper and outlines the future work.

2 OCL metamodel

As we previously mentioned the concepts of OCL 2.0 and their relationships have been defined in the form of a MOF-compliant metamodel [15]. The benefit of a metamodel for OCL is that it precisely defines the structures and syntax of all OCL concepts like types, expressions, and values in an abstract way and by means of UML features. Thus, all legal OCL expressions can be systematically derived and instantiated from the metamodel.

- The Expression package contains the main metaclasses of the OCL metamodel which are essential for the formal definition of our proposed measures. Figure 1 shows the core part of the Expression package. The basic structure in the package consists of the classes OclExpression, PropertyCallExp and VariableExp [15]. In OCL the concept of "property" conceptualize an attribute, method or rolename applied to an object –or expression who evaluates to an object-.

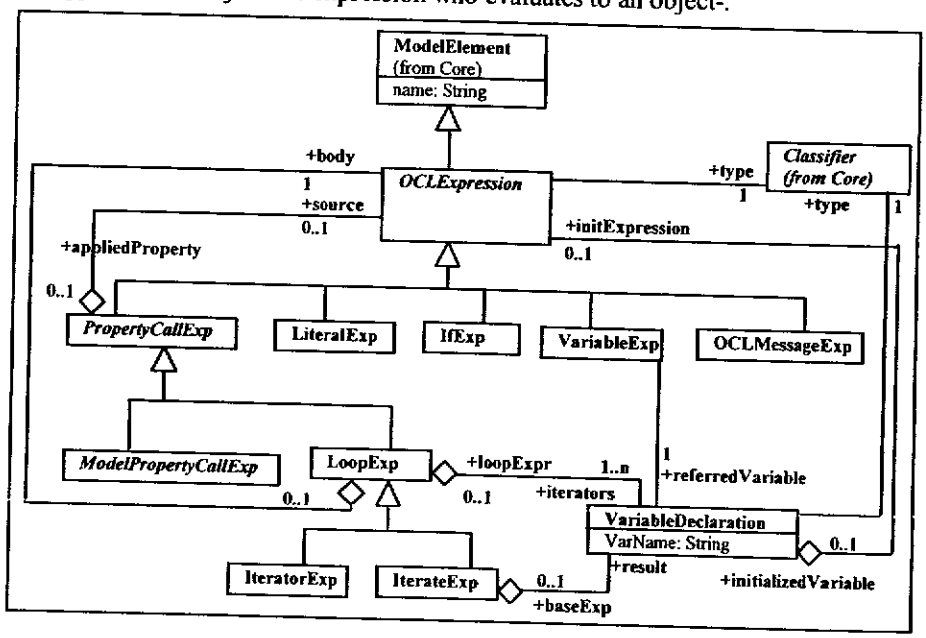


Figure 1: Abstract syntax kernel metamodel for Expressions

This definition is consistent with the fact that: (1) each PropertyCallExp has exactly one source, identified by an OclExpression; (2) A ModelPropertyCallExp (see Figure 2) –a specialization of PropertyCallExp- generalizes all property calls that refer to Features or AssociationEnds in the UML metamodel [16], for instance:

- An AttributeCallExp is a reference to an Attribute of a Classifier defined in the UML model.
- A NavigationCallExp is a reference to an AssociationEnd or AssociationClass defined in the UML model.
- An OperationCallExp refers to an Operation in a Classifier.

This is shown in Figure 2 by the three different subtypes, each of which is associated with its own type of ModelElement.

Due to the fact that the OCL metamodel is composed of a set of more than thirty metaclasses we are not able to explain each of them. Nevertheless in the following section we will describe an OCL expression as an example of instantiating some of the aforementioned OCL metaclasses.

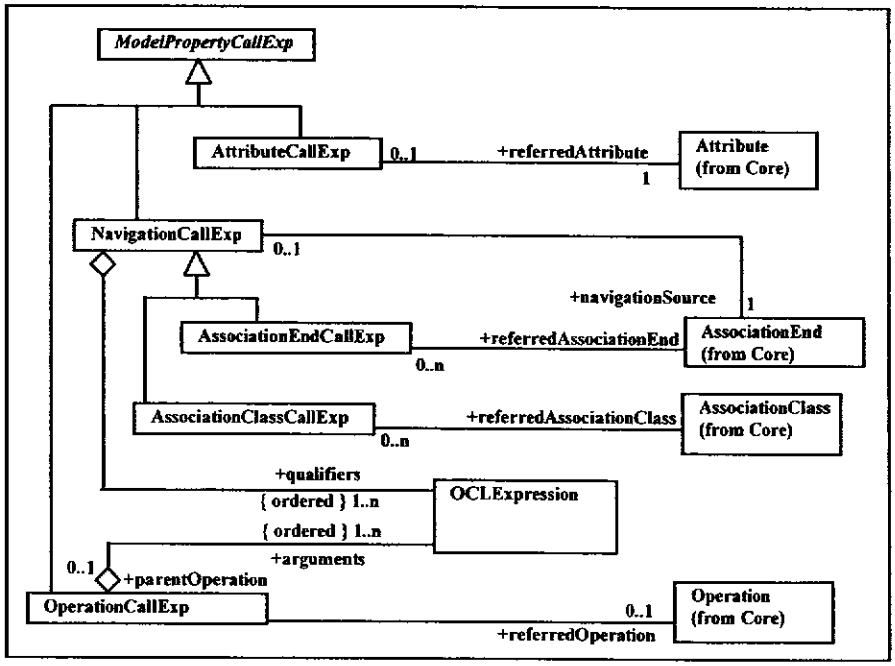


Figure 2: Abstract syntax metamodel for ModelPropertyCallExp

3 A sample of an "Abstract Syntax Tree" for an OCL expression

The purpose of this section is to show an example of the *ast* built from an OCL expression. We choose as an example the following invariant OCL expression:

context Company inv:
self.numberOfEmployees > 50

In this example we suppose a modeller defines a class named Company with an invariant expression attached in order to represent that all companies have more than fifty employees. The basic instantiation of this fragment of the model for our example is consistent with the standard place where an invariant OCL expression occurs in the

UML and OCL metamodel (shown in Figure 3). An OCL expression always constitutes the body of a Constraint object associated with one or more ModelElement objects. So, the instantiation (see Figure 4) includes two important objects:

- a class object where its name is Company (Classifier is a UML concept which represents a class, an interface, etc.)
- and a constraint object to represent an invariant constraint (see the two classes at the left top of Figure 4).

The body of the constraint will be represented by the object diagram for the *ast* of the invariant expression. The object diagram of Figure 4 basically shows an *ast* in the right part.

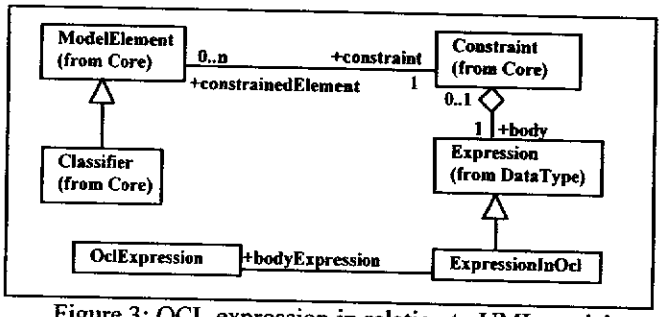


Figure 3: OCL expression in relation to UML models

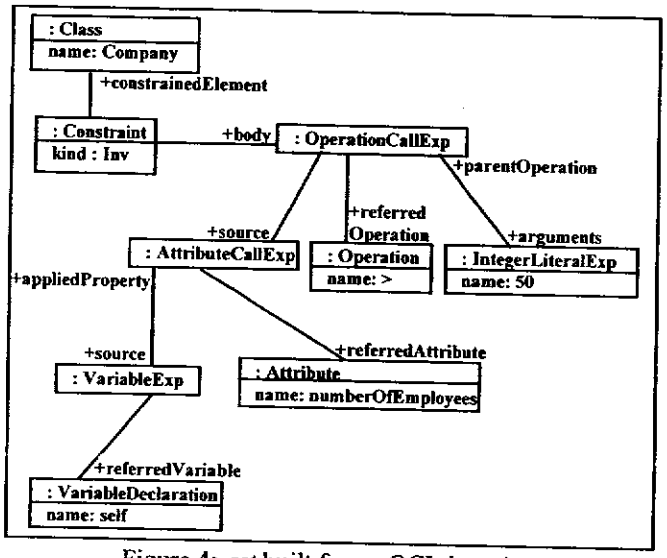


Figure 4: *ast* built for an OCL invariant

In order to build the tree, instances of the OperationCallExp, AttributeCallExp, Operation, IntegerLiteralExp, VariableExp and VariableDeclaration OCL's metaclasses had been used. In the tree there is also an instance of the Attribute UML metaclass which constitutes the attribute referred by the AttributeCallExp.

The root of the tree of Figure 4 is the OperationCallExp expression. It has three child branches. First, the source of the Operation Call expression is an AttributeCallExpression. The second branch models the referred operation, meanwhile the third branch represents the argument, an Integer Literal expression.
In order to compute the value of a specific measure we must visit each of the tree nodes (instances of OCL metaclass) and verify if each of them belongs to a particular metaclass we are interested to measure. The implemented strategy for visiting the elements is shown in the following section.

4 Implemented strategy

There are many operations we must define in order to compute the measures values, and these operations should be specified in many OCL metaclasses, but we do not want to clutter the OCL metaclasses with these operations. To solve this problem we used a Visitor Pattern [18]. The operations we must define are located into a separate object (a visitor). The visitor is sent to the tree root, eventually each element forwards the requests to its children and also its calls activate the visitor. The visitor performs operations on each element. The main participants of a Visitor are:

- **Visitor:** declares a Visitor operation for each class of ConcreteElement in the object structure.
- **ConcreteVisitor:** implements each operation declared by Visitor.
- **Element:** defines an Accept operation that takes a visitor as an argument.
- **ConcreteElement:** implements an Accept operation
- **ObjectStructure:** can enumerate its elements.

A complete explanation of this pattern can be found in [9], [18]. In our case, the element and concrete elements are represented by OCL metaclasses in the Expression package, and we will define Accept operations on them. Visitor and ConcreteVisitor are new classes introduced in our strategy to define the measures, and the object structure may be represented by either Constraint or ExpressionInOCL classes of Figure 3.
Figure 5 shows the basic UML design for implementing the strategy with a visitor. In this solution we also used an Enumeration UML class, MetricAcronym, which includes the acronym of the proposed measures, i.e. its values are NAS; NCO; DN; NAN; NES; etc.
This section is divided as follow: Subsection 4.1 shows how Accept operations were defined in the OCL metaclasses of the Expression Package. Subsection 4.2 shows the Visitor Class and its operations. Subsection 4.3 describes how to obtain the value of a measure. All the expressions used in this section were syntactically verified using ECLIPSE [8] and OCTUPUS component [11], an Eclipse plug-in.

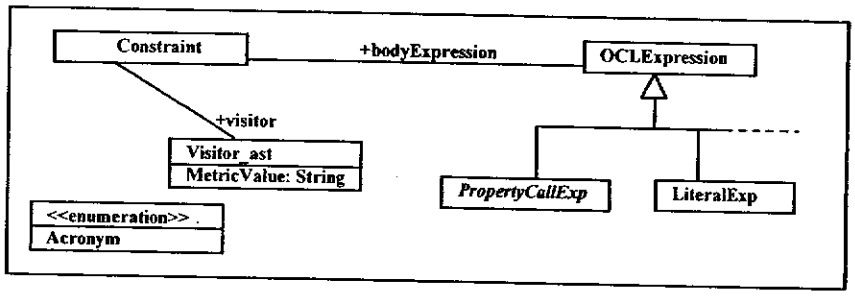


Figure 5: Design of the implemented strategy

4.1 Implementing the Accept operations in the Expression Package classes.
Accept operations are implemented in the OCL metaclasses of the Expression Package. Their definitions include many forward operations, so, it is important to understand the OCL metaclasses and their relationships. We will show as an example the Accept operations of AttributeCallExp and OperationCallExp metaclasses.

- An attributeCallExp object allows the visitor to visit itself, and forwards the visitor to its source (the expression which represent an object to which the attributes applies) whether the source is not empty.
  context AttributeCallExp::accept_new(v:Visitor, metricName: MetricAcronym)
  post: v^visitAttributeCallExp(self, metricName) and
  (self.source->notEmpty() implies
  self.source^accept_new(v, metricName))

- An OperationCallExp object allows the the visitor to visit itself, and forwards the visitor to its source (whenever the source is not empty) as well as to each of the elements of its arguments.
  context OperationCallExp::accept_new(v:Visitor, metricName: MetricAcronym)
  post: v^visitOperationCallExp(self, metricName) and
  (arguments->size() >= 1 implies
  arguments->forAll(a | a^accept_new(v, metricName)))
  and
  (self.source->notEmpty()
  implies self.source^accept_new(v, metricName))

4.2 A Visitor Class for obtaining the value of OCL measures
Visitor_*ast* class (see Figure 6) defines the visitor operations for each class of the OCL metaclasses. We exemplify how the visitor operations are defined for the OperationCallExp and AttributeCallExp classes. In these visitor operations we also show how the NAS and NCO measures are computed.

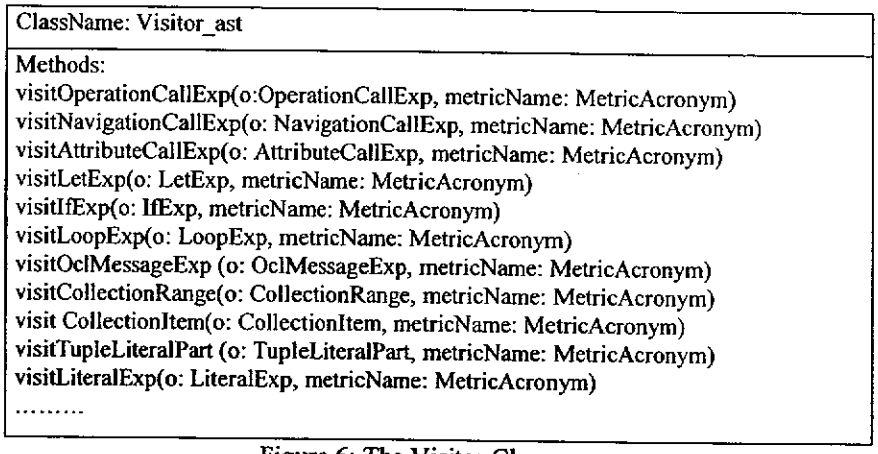


Figure 6: The Visitor Class.

When VisitOperationCallExp allows the visitor to visit itself, the visitor performs the following operation on it:

```
context Visitor::visitOperationCallExp(o: OperationCallExp, metricName: MetricAcronym)
pre: o.oclIsKindOf(OclAbstractSyntax::Expressions::OperationCallExp)
post: (metricName = MetricAcronym::NCO and
      Set{'=', '<=', '>=', '<', '>'}->includes(o.referredOperation.name))
          implies  valueMetric = valueMetric@pre + 1
```

The operation only increments the result if the measureName that is currently computed (a parameter of the operation) is NCO, and the name of the referredOperation of the OperationCallExp is an element of the set { '=', '<=', '>=', '<', '>'}. For example regarding the *ast* of Figure 4 the OperationCallExp has a referredOperation having as a name '<', and this is relevant for NCO which measure the number of comparison operation in an OCL expression.

When an AttributeCallExp allows the visitor to visit itself, the visitor performs the following operation on it:

```
context Visitor::visitAttributeCallExp(o: AttributeCallExp, metricName: MetricAcronym)
post: (metricName = MetricAcronym::NAS
          and o.source.oclIsTypeOf(VariableExp)
          and o.source.oclAsType(VariableExp).referredVariable.varName = 'self' )
          implies valueMetric = valueMetric@pre + 1
```

The operation only increments the result if the measureName that is currently computed is the NAS measure, and the name of the referred variable of visitAttributeCallExp is "self". For example, according to Figure 4, the subtree that represents self.numberofEmployee has as root an AttributeCallExp, its source variable is Vari-

ableExp whereas the referredVariable of the source is a VariableDeclaration having as a name 'self'. This condition is relevant for a NAS measure which counts the number of attributes referred by the contextual instance (self) in an OCL expression.

4.3 Explanation of how the value of a measure is obtained

Within the Constraint class many operations for obtaining the value of a measure can be defined, such as the following for NAS measure. This operation sends a visitor to the root of an *ast* in order to compute the value of the measure, then each node of the *ast* forwards in turn the request for allowing the visitor to act.

```
context Constraint::Value_of_NAS() :Integer
post: self.body^accept(self.visitor, MetricAcronym::NAS) and
        result = self.visitor.valueMetric
```

5 Conclusions

Within the OO software measurement community the formal definition of measures is an important aspect that was almost neglected. Although there is a huge amount of OO measures, the lack of formalization constitutes a serious matter. Only natural language or rigorous mathematical definitions were used, being none of these extremes suitable and widely adopted. Sometimes an increase in the focus on one aspect such a more rigorous definition, for instance when mathematical background is used, decrease focus on one other such as that not everybody is familiar with the mathematical formalism applied. On the other side, if less rigorous is applied for gaining a wide range of audience, ambiguity is introduced. Our belief is that the combination of the metamodeling facilities and the OCL as a language for defining OCL semantics, such as in defining the UML and OCL languages, allows also unambiguous measure definition achieving both understandability and formality in their specification. We claim that the formal definition of our measures using OCL language is easy to grasp by anybody familiar with metamodeling.

The relevance of the proposed approach, the specification of measures using OCL at M2 of MOF, will became more important by the proliferation of MOF-compliant architectures and the growing field of Model Driven Engineering [14], [1], [20] paradigm. The formal specification of measures allows the development of precise measures extraction tools and we believe that in the future, UML and OCL measures extraction can be translated from their formal definition to platform specific models (PSM) and from PSM to code.

Acknowledgements

This research is part of the ENIGMAS project (PBI-05-058) financed by "Consejería de Ciencia y Tecnología de la Junta de Comunidades de Castilla-La Mancha", the CALIPO project supported by "Dirección General de Investigación del Ministerio de Ciencia y Tecnología (Spain)" (TIC2003-07804-C05-03) and the COMPETISOFT-project (506PI0287) financed by "CYTED (Programa Iberoamericano de Ciencia y Tecnología para el Desarrollo)".

References

[1]. C. Atkinson and T. Kuhne. Model Driven Development: A Metamodeling Foundation. IEEE Transactions on Software Engineering, 20(5),36-41, 2003.

[2] A. L. Baroni. Formal Definition of Object-Oriented Design Metrics. Master of Science in Computer Science Thesis, Vrije Universiteit Brussel, Belgium, 2002.

[3] A. L. Baroni, S. Braz, and F. Brito e Abreu. Using OCL to Formalize Object-Oriented Design Metrics Definitions. In Proc. of QUAOOSE'2002, Malaga, Spain, 2002.

[4] A. L. Baroni and F. Brito e Abreu. Formalizing Object-Oriented Design Metrics upon the UML Meta-Model. In Proc. of the Brazilian Symposium on Software Engineering, Gramado - RS, Brazil, 2002.

[5] A. L. Baroni and F. Brito e Abreu. A Formal Library for Aiding Metrics Extraction. International Workshop on Object-Oriented Re-Engineering at ECOOP 2003. Darmstadt, Germany, 2003.

[6] S. Chidamber and C. Kemerer. A Metrics Suite for Object Oriented Design. IEEE Transactions on Software Engineering, 20(6):476-493, 1994.

[7] S. Cook, A. Kleepe, R. Mitchell, B. Rumpe, J.Warmer, and A. Wills. The Amsterdam Manifiesto on OCL. Clark T. and Warmer J., editors, Advances in Object Modelling with the OCL, 115-149, 2001.

[8] Eclipse Foundation, Inc. Ottawa, Ontario, Canada. www.eclipse.org.

[9] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley, 1995.

[10] B. Henderson-Sellers. Object-Oriented Metrics: Measures of Complexity. Prentice-Hall, 1996.

[11] Klasse Objecten. OCTUPUS: OCL Tool for Precise UML Specification. Available at http://www.klasse.nl/octopus/index.html.

[12] W. Liang. Uml Object Constraint Language in Meta-Modeling. School of Computer Science. McGill University, 2002.

[13] M. Lorenz and J. Kidd. Object-Oriented Software Metrics: A Practical Guide. Prentice Hall, Englewood Cliffs, NJ, EUA. 1994.

[14] Object Management Group. MDA - The OMG Model Driven Architecture. 2002.

[15] Object Management Group. UML 2.0 OCL Final Adopted Specification. OMG Document 2005-06-06, 2005.

[16] Object Management Group. UML Specification. OMG Document formal/03-03-01, 2003.

[17] L. Reynoso, M. Genero and M. Piattini: Measuring OCL Expressions: An approach based on Cognitive Techniques. M. Genero, M. Piattini, and M. Calero (Eds.). Metrics For Software Conceptual Models. Imperial College Press, UK. 2005.

[18] L. Reynoso and R. Moore. GoF Behavioural Patterns: A Formal Specification. Technical Report Nro. 200. UNU/IIST, P.O.Box 3058, Macau, 2000.

[19] M. Richters. A Precise Approach to Validating UML Models and OCL Constraints. Monographs of the Bremen Institute of Safe Systems, 2001.
[20] B. Selic. The Pragmatics of Model-Driven Development. IEEE Software., 20(5),19-25, 2003.
[21] J. Warmer and A. Kleppe. The Object Constraint Language. Precise Modeling with UML. Object Technology Series.Addison Wesley, 1999.
[22] J. Warmer and A. Kleppe. The Object Constraint Language. Second Edition. Getting Your Models Ready for MDA. Object Technology Series. Addison-Wesley. Massachusetts., 2003.