

# Extending the UML for Multidimensional Modeling<sup>\*</sup>

Sergio Luján-Mora<sup>1</sup>, Juan Trujillo<sup>1</sup>, and Il-Yeol Song<sup>2</sup>

<sup>1</sup> Dept. de Lenguajes y Sistemas Informáticos  
Universidad de Alicante (Spain)  
{slujan, jtrujillo}@dlsi.ua.es

<sup>2</sup> College of Information Science and Technology  
Drexel University (USA)  
songiy@drexel.edu

**Abstract.** Multidimensional (MD) modeling is the foundation of data warehouses, MD databases, and On-Line Analytical Processing (OLAP) applications. In the past few years, there have been some proposals for representing the main MD properties at the conceptual level providing their own notations. In this paper, we present an extension of the Unified Modeling Language (UML), by means of stereotypes, to elegantly represent main structural and dynamic MD properties at the conceptual level. We make use of the Object Constraint Language (OCL) to specify the constraints attached to the defined stereotypes, thereby avoiding an arbitrary use of these stereotypes. The main advantage of our proposal is that it is based on a well-known standard modeling language, thereby designers can avoid learning a new specific notation or language for MD systems. Finally, we show how to use these stereotypes in Rational Rose 2000 for MD modeling.

**Keywords:** UML, UML extensions, multidimensional modeling, OCL, Rational Rose

## 1 Introduction

Multidimensional (MD) modeling is the foundation of data warehouses (DW), MD databases, and On-Line Analytical Processing (OLAP) applications. These systems provide companies with many years of historical information for the decision making process. Various approaches for the conceptual design of MD systems have been proposed in the last few years such as [1][2][3][4][5] to represent main MD structural and dynamic properties. These approaches provide their own graphical notations, which forces designers to learn a new specific model together with its corresponding MD modeling notation. Furthermore, none of these approaches has been widely accepted as a standard conceptual model for

---

<sup>\*</sup> This paper has been partially supported by the Spanish Ministry of Science and Technology, project number TIC2001-3530-C02-02.

MD modeling. Due to space constraints, we refer the reader to [6] for a detailed comparison and discussion about most of these models.

On the other hand, the Unified Modeling Language (UML) [7][8] has been widely accepted as the standard object-oriented (OO) modeling language for modeling various aspects of software systems. Therefore, any approach using the UML will minimize the effort of developers to learn new notations or methodologies for every subsystem to be modeled. Following this consideration, we have previously proposed in [5] an OO conceptual MD modeling approach, based on the UML for a powerful conceptual MD modeling. This proposal considers major relevant structural and dynamic MD properties at the conceptual level in an elegant and easy way.

The UML is an extensible language, in the sense that it provides mechanisms (stereotypes, tagged values, and constraints) to introduce new elements for specific domains if necessary, such as web applications, database applications, business modeling, software development processes, etc. [9,10]. A collection of enhancements that extend an existing diagram type to support a new purpose is called a *profile*.

In this paper, we present a UML profile for MD modeling based on our previously proposed approach [5], which easily and elegantly considers main MD properties at the conceptual level, such as the many-to-many relationships between facts and dimensions, degenerate dimensions, multiple and alternative path classification hierarchies, and non-strict and complete hierarchies. Our extension uses the Object Constraint Language (OCL) [8] for expressing well-formedness rules of the new defined elements, thereby avoiding an arbitrary use of this extension. Moreover, we program this extension in a well-known model-driven development tool such as Rational Rose to show its applicability.

The remainder of this paper is structured as follows: Section 2 briefly presents other works related to extending the UML for database design. Section 3 introduces the main MD concepts such as fact, dimension, and hierarchy level that our approach comprises. Section 4 summarizes the UML Extensibility Mechanism. Section 5 proposes the new UML extension for MD modeling. Section 6 shows how to use our MD extension in Rational Rose. Finally, Section 7 presents the main conclusions and introduces our future work.

## 2 Related Work

In the past few years, some proposals to extend the UML for database design have been presented, since the UML does not explicitly include a data model. In [11], “...a profile that extends the existing class diagram definition to support persistence modeling” is presented. This profile is intended to make objects persistent in different storages: files, relational databases, object-relational databases, etc. In [12], the Data Modeling profile for the UML is described, “...including descriptions and examples for each concept including database, schema, table, key, index, relationship, column, constraint and trigger”. In [10], the process of UML-based database modeling and design is explained: it presents the UML Profile

for Database Design created by Rational Software Corporation. Finally, in [13] an Object-Relational Database Design Methodology is presented. The methodology defines new UML stereotypes for Object-Relational Database Design and proposes some guidelines to translate a UML schema into an object-relational schema. However, these proposals do not reflect the peculiarities of MD modeling. Therefore, in this paper we present a proposal to extend the UML for MD modeling.

### 3 Object-Oriented Multidimensional Modeling

In this section, we summarize<sup>3</sup> how an OO MD model, based on the UML, can represent main structural and dynamic MD properties at the conceptual level. Most of the MD features considered by this approach such as the many-to-many relationships between facts and dimensions, degenerate dimensions, multiple and alternative path classification hierarchies, and non-strict and complete hierarchies are misinterpreted by most of the conceptual MD models. In the approach proposed in this paper, the main structural properties of MD modeling are specified by means of a UML class diagram in which the information is clearly separated into facts and dimensions<sup>4</sup>.

Facts and dimensions are represented by *fact classes* and *dimension classes*, respectively. Then, fact classes are specified as composite classes in shared aggregation relationships of  $n$  dimension classes. The flexibility of shared aggregation in the UML allows us to represent *many-to-many* relationships between facts and particular dimensions by indicating the 1..\* cardinality on the dimension class role. For example, in Fig. 1 (a), we can see how the fact class Sales has a many-to-many relationship with the dimension class Product and a one-to-many relationship with the dimension class Time.

By default, all measures in the fact class are considered additive. For non-additive measures, additivity rules are defined as constraints and are included in the fact class. Furthermore, derived measures can also be explicitly considered (indicated by /) and their derivation rules are placed between braces near the fact class, as shown in Fig. 1 (a).

This OO approach also allows us to define identifying attributes in the fact class, by placing the constraint  $\{OID\}$  next to an attribute name. In this way we can represent *degenerate dimensions* [14][15], thereby providing other fact features in addition to the measures for analysis. For example, we could store the ticket number (ticket\_number) as a degenerate dimension, as reflected in Fig. 1 (a).

With respect to dimensions, every *classification hierarchy* (Fig. 1 (b)) level is specified by a class (called a *base class*). An association of classes specifies the relationships between two levels of a classification hierarchy. The only prerequisite is that these classes must define a Directed Acyclic Graph (DAG) rooted in the dimension class (constraint  $\{dag\}$  placed next to every dimension class).

<sup>3</sup> We refer the reader to [5] for a complete description of this approach.

<sup>4</sup> It is not the scope of this paper to consider dynamic MD properties.

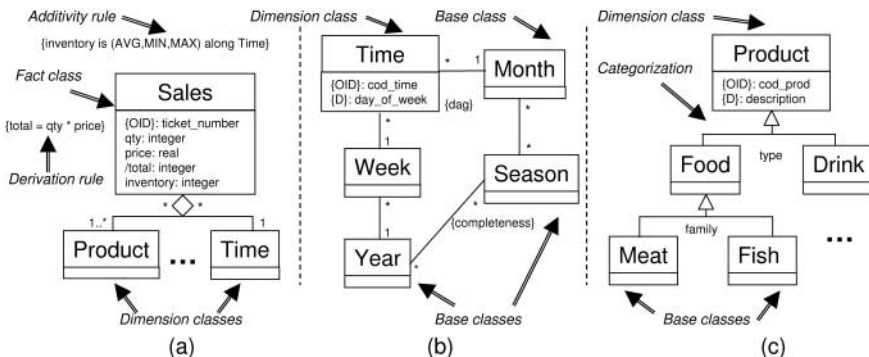


Fig. 1. Multidimensional modeling using the UML

The DAG structure can represent both alternative path and multiple classification hierarchies. Every classification hierarchy level must have an *identifying* attribute (constraint  $\{OID\}$ ) and a *descriptor* attribute<sup>5</sup> (constraint  $\{D\}$ ). These attributes are necessary for an automatic generation process into commercial relational OLAP (ROLAP) tools, as these tools need to store these attributes in their metadata. The multiplicity  $1$  and  $1..*$  defined in the target associated class role addresses the concepts of *strictness* and *non-strictness* respectively. Strictness means that an object at a hierarchy’s lower level belongs to only one higher-level object (e.g., as one month can be related to more than one season, the relationship between both of them is non-strict). Moreover, defining the  $\{completeness\}$  constraint in the target associated class role addresses the completeness of a classification hierarchy (see an example in Fig. 1 (b)). By completeness we mean that all members belong to one higher-class object and that object consists of those members only. For example, all the recorded seasons form a year, and all the seasons that form the year have been recorded. Our approach assumes all classification hierarchies are non-complete by default.

The *categorization of dimensions*, used to model additional features for class’s subtypes, is considered by means of generalization-specialization relationships. However, only the dimension class can belong to both a classification and specialization hierarchy at the same time. An example of categorization for the **Product** dimension can be observed in Fig. 1 (c).

## 4 UML Extensibility Mechanism

The UML Extensibility Mechanism package is the subpackage from the UML metamodel that specifies how specific UML model elements are customized and extended with new semantics by using stereotypes, tagged values, and constraints. A coherent set of such extensions, defined for specific purposes, constitutes a UML *profile*. For example, the UML 1.4 [8] includes a standard profile

<sup>5</sup> A descriptor attribute will be used as the default label in the data analysis.

for modeling software development processes and another one for business modeling.

A *stereotype* is a model element that defines additional values (based on tagged values), additional constraints, and optionally a new graphical representation (an icon): a stereotype allows us to attach a new semantic meaning to a model element. A stereotype is either represented as a string between a pair of guillemots ( $\ll \gg$ ) or rendered as a new icon.

A *tagged value* specifies a new kind of property that may be attached to a model element. A tagged value is rendered as a string enclosed by brackets ( $( )$ ) and placed below the name of another element.

A *constraint* can be attached to any model element to refine its semantics. As it is stated in [16], “A constraint is a restriction on one or more values of (part of) an object-oriented model or system”. In the UML, a constraint is rendered as a string between a pair of braces ( $\{ \}$ ) and placed near the associated model element. A constraint on a stereotype is interpreted as a constraint on all types on which the stereotype is applied. A constraint can be defined by means of an informal explanation or by means of OCL [16][8] expressions. The OCL is a declarative language that allows software developers to write constraints over object models.

## 5 UML Extension for Multidimensional Modeling

According to [9], “An extension to the UML begins with a brief description and then lists and describes all of the stereotypes, tagged values, and constraints of the extension. In addition to these elements, an extension contains a set of well-formedness rules. These rules are used to determine whether a model is semantically consistent with itself”. According to this quote, we define our UML extension for MD modeling following the schema shown in Table 1.

<ul style="list-style-type: none"> <li>- <b>Description:</b> A little description of the extension in natural language.</li> <li>- <b>Prerequisite Extensions:</b> It indicates whether the current extension needs the existence of previous extensions.</li> <li>- <b>Stereotypes:</b> The definition of the stereotypes.</li> <li>- <b>Well-Formedness Rules:</b> The static semantics of the metaclasses are defined as a set of invariants defined by means of OCL expressions.</li> <li>- <b>Comments:</b> Any additional comment, decision or example, usually written in natural language.</li> </ul>
---

**Table 1.** Extension definition schema

For the definition of the stereotypes and the tagged values, we follow the structure of the examples included in the UML Specification [8]. In Table 2 and Table 3 we show the schemas followed in our definition of the stereotypes and the tagged values, respectively.

We have defined eight stereotypes: three specialize in the Class model element, four specialize in the Attribute model element, and one specializes in the

- **Name:** The name of the stereotype.
- **Base class** (also called Model class): The UML metamodel element that serves as the base for the stereotype.
- **Description:** An informal description with possible explanatory comments.
- **Icon:** It is possible to define a distinctive visual cue (an icon).
- **Constraints:** A list of constraints defined by means of OCL expressions associated with the stereotype, with an informal explanation of the expressions.
- **Tagged values:** A list of all tagged values that are associated with the stereotype.

**Table 2.** Stereotype definition schema

- **Name:** The name of the tagged value.
- **Type:** The name of the type of the values that can be associated with the tagged value.
- **Multiplicity:** The maximum number of values that may be associated with the tagged value.
- **Description:** An informal description with possible explanatory comments.

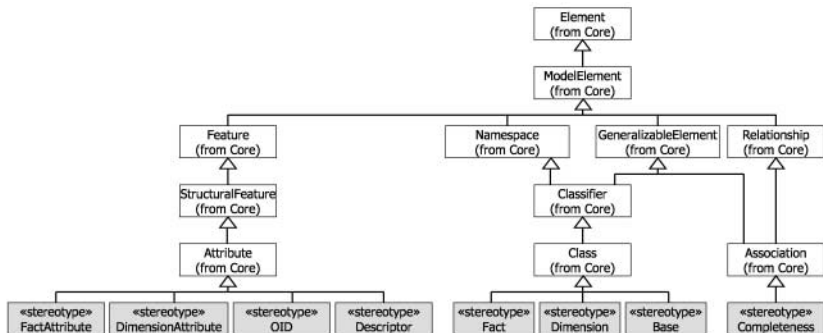
**Table 3.** Tagged value definition schema

Association model element. In Fig. 2, we have represented a portion of the UML metamodel<sup>6</sup> to show where our stereotypes fit.

Some issues of our MD approach, such as the derivation rule or the initial value of an attribute, are not defined in our stereotypes because these concepts have already been defined in the UML metamodel. We provide a list of these concepts in Table 4.

In the following, we present our extension following the extension definition schema shown in Table 1.

<sup>6</sup> All the metaclasses come from the Core Package, a subpackage of the Foundation Package.



**Fig. 2.** Extension of the UML with stereotypes

Concept	Comes from	Description	Used by
name	ModelElement	It is an identifier for the ModelElement	Base, Completeness, Descriptor, Dimension, DimensionAttribute, Fact, FactAttribute, OID
documentation	Element	It is a comment, description or explanation of the Element to which it is attached	Base, Completeness, Descriptor, Dimension, DimensionAttribute, Fact, FactAttribute, OID
type	StructuralFeature	Designates the classifier whose instances are values of the feature	Descriptor, DimensionAttribute, FactAttribute, OID
initialValue	Attribute	An expression specifying the value of the Attribute upon initialization	Descriptor, DimensionAttribute, FactAttribute, OID
derived	ModelElement	A true value indicates that the ModelElement can be completely derived from other model elements and is therefore logically redundant	Descriptor, DimensionAttribute, FactAttribute

Table 4. Concepts inherited from the UML metamodel

### 5.1 Description

This extension to the UML defines a set of stereotypes, tagged values, and constraints that enable us to model MD models. The stereotypes are applied to certain components that are particular to MD modeling, allowing us to represent them in the same model and on the same diagrams that describe the rest of the system. The principal elements to MD modeling are the Fact class and the Dimension class. A Fact class consists of OIDs and FactAttributes, whereas a Dimension class consists of an OID, Descriptor, and DimensionAttributes. Moreover, the hierarchy levels of a Dimension are represented by means of Base classes. Finally, a Completeness association is defined.

### 5.2 Prerequisite Extensions

No other extension to the language is required for the definition of this extension.

### 5.3 Stereotypes

#### Stereotypes of Class

- Name: **Fact**
- Base class: Class
- Description: Classes of this stereotype represent facts in a MD model
- Icon: Fig. 3 (a)
- Constraints:
  - All attributes of a Fact must be OID or FactAttribute:  
`self.feature->select(oclIsKindOf(Attribute))->forAll(oclIsTypeOf(OID) or oclIsTypeOf(FactAttribute))`
  - All associations of a Fact must be aggregations:  
`self.association->forAll(aggregation = #aggregate)`

- A Fact can only be associated with Dimension classes:  
self.allOppositeAssociationEnds->forAll(participant.ocllsTypeOf(Dimension))
  - Tagged values: None
- 
- Name: **Dimension**
  - Base class: Class
  - Description: Classes of this stereotype represent dimensions in a MD model
  - Icon: Fig. 3 (b)
  - Constraints:
    - All attributes of a Dimension must be OID, Descriptor, or DimensionAttribute:  
self.feature->select(ocllsKindOf(Attribute))->forAll(ocllsTypeOf(OID) or  
ocllsTypeOf(Descriptor) or ocllsTypeOf(DimensionAttribute))
    - All associations of a Dimension with a Fact must be aggregations at the end  
of the Fact (the opposite end):  
self.association.association->forAll(associationEnd.participant.ocllsTypeOf(Fact) im-  
plies associationEnd.aggregation = #aggregate)
    - All associations of a Dimension with a Fact must not be aggregations at the  
end of the Dimension (the current end):  
self.association.association->forAll(associationEnd.participant.ocllsTypeOf(Fact) im-  
plies aggregation <> #aggregate)
    - A Dimension cannot be associated with another Dimension:  
self.allOppositeAssociationEnds->forAll(not participant.ocllsTypeOf(Dimension))
  - Tagged values:
    - isTime:
      - \* Type: UML::Datatypes::Boolean
      - \* Multiplicity: 1
      - \* Description: Indicates whether the dimension represents a time dimension  
or not

- 
- Name: **Base**
  - Base class: Class
  - Description: Classes of this stereotype represent dimension hierarchy levels in a  
MD model
  - Icon: Fig. 3 (c)
  - Constraints:
    - All attributes of a Base must be OID, Descriptor, or DimensionAttribute:  
self.feature->select(ocllsKindOf(Attribute))->forAll(ocllsTypeOf(OID) or  
ocllsTypeOf(Descriptor) or ocllsTypeOf(DimensionAttribute))
    - A Base must have an OID attribute and a Descriptor attribute:  
self.feature->select(ocllsKindOf(Attribute))->exist(ocllsTypeOf(OID)) and  
self.feature->select(ocllsKindOf(Attribute))->exist(ocllsTypeOf(Descriptor))
    - A Base can only be associated with another Base or another Dimension:  
self.allOppositeAssociationEnds->forAll(participant.ocllsTypeOf(Base) or partici-  
pant.ocllsTypeOf(Dimension))
    - A Base can only be child in one generalization:  
self.generalization->size <= 1
    - A Base cannot simultaneously belong to a generalization/specialization hier-  
archy and an association hierarchy:  
(self.generalization->size > 0 or self.specialization->size > 0) implies  
(self.association->size = 0 )
  - Tagged values: None

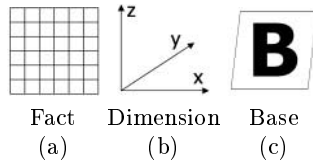


Fig. 3. Stereotype icons

### Stereotypes of Attribute

- Name: **OID**
- Base class: Attribute
- Description: Attributes of this stereotype represent OID attributes of Fact, Dimension or Base classes in a MD model<sup>7</sup>
- Icon: Fig. 4 (a)
- Constraints:
  - An OID cannot be derived:  
not self.derived
- Tagged values: None

- Name: **FactAttribute**
- Base class: Attribute
- Description: Attributes of this stereotype represent attributes of Fact classes in a MD model
- Icon: Fig. 4 (b)
- Constraints:
  - A FactAttribute can only belong to a Fact:  
self.owner.ocllsTypeOf(Fact)
  - If a FactAttribute is derived, then it needs a derivation rule (an OCL expression):  
self.derived implies self.derivationRule.ocllsTypeOf(OclExpression)
- Tagged values:
  - derivationRule:
    - \* Type: UML::Datatypes::String
    - \* Multiplicity: 1
    - \* Description: If the attribute is derived, this tagged value represents the derivation rule

- Name: **Descriptor**
- Base class: Attribute
- Description: Attributes of this stereotype represent descriptor attributes of Dimension or Base classes in a MD model
- Icon: Fig. 4 (c)
- Constraints:
  - A Descriptor can only belong to a Dimension or Base:  
self.owner.ocllsTypeOf(Dimension) or self.owner.ocllsTypeOf(Base)

<sup>7</sup> See Section 3 or [5] for further information on OID and Descriptor attributes.

- If a Descriptor is derived, then it needs a derivation rule (an OCL expression):  
self.derived implies self.derivationRule.ocllsTypeOf(OclExpression)
- Tagged values:
  - derivationRule:
    - \* Type: UML::Datatypes::String
    - \* Multiplicity: 1
    - \* Description: If the attribute is derived, this value represents the derivation rule

- Name: **DimensionAttribute**
- Base class: Attribute
- Description: Attributes of this stereotype represent attributes of Dimension or Base classes in a MD model
- Icon: Fig. 4 (d)
- Constraints:
  - A DimensionAttribute can only belong to a Dimension or Base:  
self.owner.ocllsTypeOf(Dimension) or self.owner.ocllsTypeOf(Base)
  - If a DimensionAttribute is derived, then it needs a derivation rule (an OCL expression):  
self.derived implies self.derivationRule.ocllsTypeOf(OclExpression)
- Tagged values:
  - derivationRule:
    - \* Type: UML::Datatypes::String
    - \* Multiplicity: 1
    - \* Description: If the attribute is derived, this value represents the derivation rule

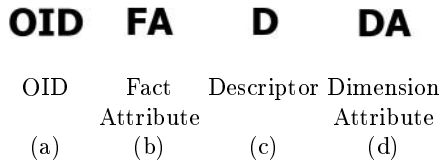


Fig. 4. Stereotype icons

### Stereotype of Association

- Name: **Completeness**
- Base class: Association
- Description: Associations of this stereotype represent complete associations<sup>8</sup>
- Icon: None
- Constraints:
  - The ends of a Completeness association can only be Dimension or Base classes:  
self.associationEnd.participant->forAll(ocllsTypeOf(Dimension) or ocllsTypeOf(Base))
- Tagged values: None

<sup>8</sup> A complete association means that all members belong to one higher-class object and that object consists of those members only.

## 5.4 Well-Formedness Rules

### Namespace

- All the classes in a MD model must be Fact, Dimension, or Base:<sup>9</sup>  
self.allContents->forAll(ocllsKindOf(Class) implies (ocllsTypeOf(Fact) or  
ocllsTypeOf(Dimension) or ocllsTypeOf(Base)))

## 5.5 Comments

Next, we summarize the UML elements we have just used or defined to consider the main relevant MD properties:

- Facts and dimensions: they are represented by means of Fact and Dimension stereotypes.
- Many-to-many relationships: thanks to the flexibility of the shared-aggregation relationships, we can consider many-to-many relationships between facts and particular dimensions by means of the 1..\* cardinality on the dimension class role.
- Derived measures: they are represented by means of `derived` attributes from the UML metamodel and the tagged value `derivationRule` we have defined in the Descriptor, DimensionAttribute, and FactAttribute stereotypes.
- Additivity: all FactAttributes are considered additive by default. For nonadditive FactAttributes, their additivity rules are defined as constraints placed near the corresponding Fact class.
- Classification hierarchies: they are considered by means of the association between Dimension and Base stereotypes.
- Strictness: the multiplicity 1 and 1..\* defined in the target associated class role of a classification hierarchy address the concepts of strictness and non-strictness.
- Completeness: the stereotype Completeness addresses the completeness of a classification hierarchy.
- Categorizing dimensions: we use generalization-specialization relationships to categorize a Dimension.

## 6 Using Multidimensional Modeling in Rational Rose

Rational Rose (RR) is one of the most well-known visual modeling tools. RR is extensible by means of add-ins, which allows to package customizations and automation of several RR features through the Rose Extensibility Interface (REI) [17] into one component. An add-in is a collection of some combination of the following: main menu items, shortcut menu items, custom specifications, properties (UML tagged values), data types, UML stereotypes, online help, context-sensitive help, and event handling.

<sup>9</sup> allContents is an additional operation defined in the UML specification [8]: “The operation allContents results in a Set containing all ModelElements contained by the Namespace”.

In this section, we present an add-in we have developed, which allows us to use the extension we have defined in RR. Therefore, we can use this tool to easily model MD conceptual models.

Our add-in customizes the following elements:

- Stereotypes: We have defined the stereotypes by means of a stereotype configuration file.
- Properties: We have defined the tagged values by means of a property configuration file.
- Menu item: We have added the new menu item MD Validate in the menu Tools by means of a menu configuration file. This menu item runs a Rose script that validates a MD model: our script checks all the constraints we have presented in Section 5.

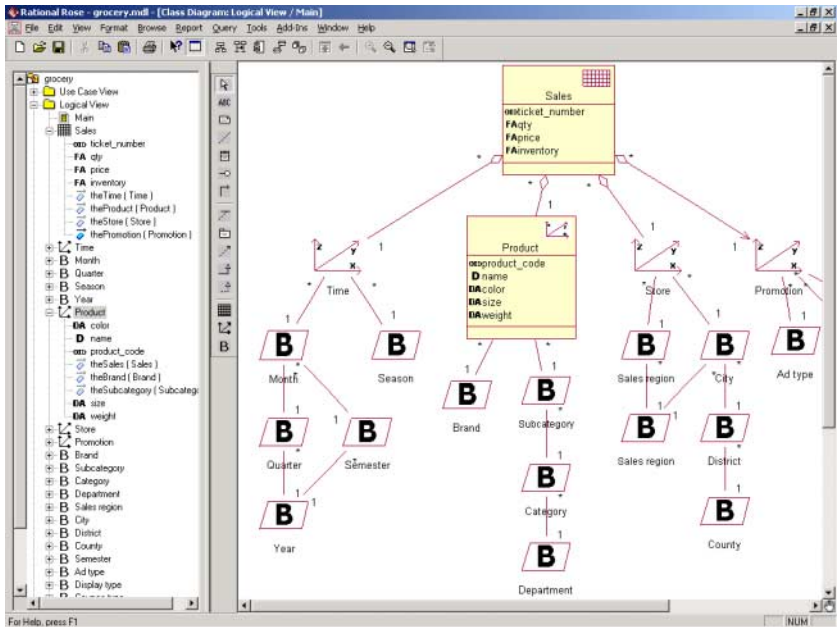


Fig. 5. Multidimensional modeling using Rational Rose

The best way to understand our extension is to show a tangible example. Fig. 5 shows a MD conceptual model of the well-known example “Grocery” as described in Chapter 2 of [15]. This example contains one Fact class, Sales, and four Dimension classes: Time, Product, Store, and Promotion. Every classification hierarchy level of a Dimension class is represented by a Base class. For example, the classification hierarchy of Time comprises the following Base classes: Month, Quarter, Semester, Year, and Season. For the sake of clarity, we do not show all the attributes: only the attributes of Sales and Product are displayed. We can also notice that a stereotype can be displayed in different manners in RR: Sales

and `Product` are displayed using the stereotype icon inside the class, whereas the rest of elements are displayed using the stereotype icon.

Fig. 6 displays the stereotype configuration file that contains our MD stereotypes. To graphically distinguish the model elements of different stereotypes, each stereotype can have a graphical representation. Thus, we have provided all the needed icons concerning these graphical stereotype representations to facilitate their use.

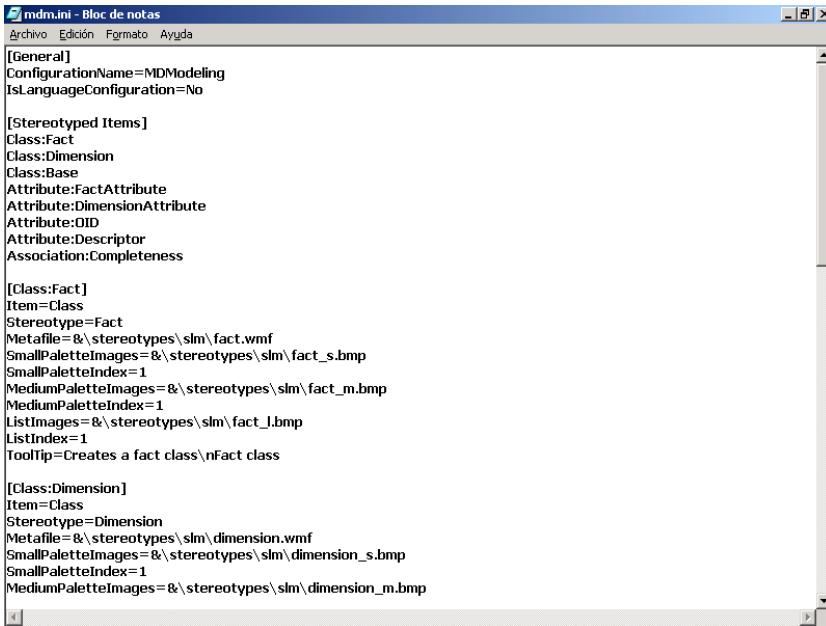


Fig. 6. Definition of the stereotypes in Rational Rose

Finally, an add-in can also extend or customize RR menus by means of a menu configuration file. We only add a new menu item called `MD Validate` in the `Tools` menu. This new menu item executes a `Rose` script that validates the correctness of a MD model: it checks all the OCL constraints we have presented in Section 5. For example, a fragment of this script is shown in Fig. 7. In this fragment we can notice the function `VAssociationFact`, which validates the associations of a `Fact`. It checks the OCL constraints we have previously presented in the `Fact` stereotype:

- All associations of a `Fact` must be aggregations.
- A `Fact` can only be associated with `Dimension` classes.

```

' Validate the associations of a Fact class
Function VAssociationFact(aAssociation As Association, aClass As Class) As Integer
  Dim message As String
  Dim myRole As Role, myOtherRole As Role
  Dim myOtherClass As Class

  ' All associations of a Fact must be aggregations
  Set myRole = aAssociation.GetCorrespondingRole(aClass)
  If Not myRole.Aggregate Then
    message = "No aggregation in association of Fact " + aClass.Name
    message = message & vbCrLf & "Do you like to continue the validation?"
    If MsgBox(message, vbCritical + vbYesNo, "Validation Error") = vbYes Then
      VAssociationFact = 1
    Else
      VAssociationFact = 2
      Exit Function
    End If
  Else
    VAssociationFact = 0
  End If

  ' A Fact can only be associated to Dimension classes
  Set myOtherRole = aAssociation.GetOtherRole(aClass)
  Set myOtherClass = myOtherRole.Class
  If myOtherClass.Stereotype <> "Dimension" Then
    message = "Incorrect class (" & myOtherClass.Name & ") in association of Fact " + aClass.Name
    message = message & vbCrLf & "Do you like to continue the validation?"
    If MsgBox(message, vbCritical + vbYesNo, "Validation Error") = vbYes Then
      VAssociationFact = 1
    Else
      VAssociationFact = 2
    End If
  Else
    VAssociationFact = 0
  End If
End Function

```

Fig. 7. Validation script for Rational Rose

## 7 Conclusions and Future Work

In this paper, we have presented an extension of the UML that allows us to represent the major relevant structural MD properties at the conceptual level. This extension contains the needed stereotypes, tagged values and constraints for a complete and powerful MD modeling. We have used the OCL to specify the constraints attached to these new defined elements, thereby avoiding an arbitrary use of them. We have also programmed this extension in a well-known visual modeling tool, Rational Rose. The main relevant advantage of this approach is that it uses the UML, a widely-accepted object-oriented modeling language, which saves developers from learning a new model and its corresponding notations for specific MD modeling. Furthermore, the UML allows us to represent some MD properties that are hardly considered by other conceptual MD proposals.

We are currently working on defining a methodology for MD modeling based on the extension presented in this paper. This methodology will explicitly consider all underlying design guidelines that are hidden under every defined new MD element. Furthermore, in this UML extension we are also considering new stereotypes regarding object-oriented and object-relational databases for an automatic generation of the database schema into these kinds of databases.

## References

1. Golfarelli, M., Rizzi, S.: A methodological Framework for Data Warehouse Design. In: Proc. of the ACM 1st Intl. Workshop on Data warehousing and OLAP (DOLAP'98). (1998) 3–9
2. Sapia, C., Blaschka, M., Höfling, G., Dinter, B.: Extending the E/R Model for the Multidimensional Paradigm. In: Proc. of the 1st Intl. Workshop on Data Warehouse and Data Mining (DWDM'98). Volume 1552 of LNCS., Springer-Verlag (1998) 105–116
3. Tryfona, N., Busborg, F., Christiansen, J.: starER: A Conceptual Model for Data Warehouse Design. In: Proc. of the ACM 2nd Intl. Workshop on Data warehousing and OLAP (DOLAP'99). (1999)
4. Husemann, B., Lechtenborger, J., Vossen, G.: Conceptual Data Warehouse Design. In: Proc. of the 2nd Intl. Workshop on Design and Management of Data Warehouses (DMDW'00), Stockholm, Sweden (2000)
5. Trujillo, J., Palomar, M., Gómez, J., Song, I.Y.: Designing Data Warehouses with OO Conceptual Models. *IEEE Computer*, special issue on Data Warehouses **34** (2001) 66–75
6. Abelló, A., Samos, J., Saltor, F.: A Framework for the Classification and Description of Multidimensional Data Models. In: Proc. of the 12th Intl. Conference on Database and Expert Systems Applications (DEXA'01), Munich, Germany (2001) 668–677
7. Booch, G., Rumbaugh, J., Jacobson, I.: *The Unified Modeling Language: User Guide*. Object Technology Series. Addison-Wesley (1999)
8. Object Management Group (OMG): *Unified Modeling Language Specification 1.4*. Internet: <http://www.omg.org/cgi-bin/doc?formal/01-09-67> (2001)
9. Conallen, J.: *Building Web Applications with UML*. Object Technology Series. Addison-Wesley (2000)
10. Naiburg, E., Maksimchuk, R.: *UML for Database Design*. Object Technology Series. Addison-Wesley (2001)
11. Ambler, S.: *Persistence Modeling in the UML*. *Software Development Online*. Internet: <http://www.sdmagazine.com/documents/s=755/sdm9908q/> (1999)
12. Rational Software Corporation: *The UML and Data Modeling*. Internet: <http://www.rational.com/media/whitepapers/Tp180.PDF> (2000)
13. Marcos, E., Vela, B., Cavero, J.M.: Extending UML for Object-Relational Database Design. In: Proc. of the 4th Intl. Conference UML 2001. Volume 2185 of LNCS., Springer-Verlag (2001) 225–239
14. Giovinazzo, W.: *Object-Oriented Data Warehouse Design. Building a star schema*. Prentice-Hall, New Jersey, USA (2000)
15. Kimball, R.: *The data warehousing toolkit*. 2 edn. John Wiley (1996)
16. Warmer, J., Kleppe, A.: *The Object Constraint Language. Precise Modeling with UML*. Object Technology Series. Addison-Wesley (1998)
17. Rational Software Corporation: *Using the Rose Extensibility Interface*. Rational Software Corporation (2001)