Extending UML for Multidimensional Modeling

Sergio Luján-Mora, Juan Trujillo
Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante (Spain)
{slujan,jtrujillo}@dlsi.ua.es

Abstract

Multidimensional (MD) modeling is the foundation of data warehouses, MD databases, and OLAP applications. In the last years, there have been some proposals to represent MD properties at the conceptual level. Nevertheless, none of them considers all multidimensional properties at both the structural and dynamic levels. In this paper, we present an object-oriented (OO) approach to accomplish the MD modeling at the conceptual level. This approach defines an extension by means of stereotypes to the the Unified Modeling Language (UML) for MD modeling. The extension uses the Object Constraint Language (OCL) for expressing well-formedness rules of the new defined elements. The advantages of our proposal are twofold: on the one hand, this extension allows us to represent MD models with the UML, allowing us to specify the whole system in a uniform way; on the other hand, an OO approach can elegantly consider main MD properties at the conceptual level. 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. MD modeling structures information into facts and dimensions. A fact is an analyzed item of interest for a company (sales, deliveries, etc.), whereas a dimension represents the context for analyzing a fact (product, customer, time, etc.). Various approaches for the conceptual design of MD systems have been proposed in the last few years [6][13][15][14] to represent main MD structural and dynamic properties. However, none of them has been widely accepted as a standard conceptual model for MD modeling. Due to space constraints, we refer the reader to [1] for a detailed comparison and discussion about most of these models.

The Unified Modeling Language (UML) [3, 10] has become the *de facto* standard for modeling systems (and not just software) using object-oriented concepts. UML is an extensible language, in the sense it provides mechanisms (stereotypes, tagged values, and constraints) that allow introducing new elements for specific domains if it is needed, such as web applications, database applications, business modeling, software development processes, etc. The definition of a collection of enhancements (stereotypes, tagged values, and

constraints) that extend an existing diagram type to support a new purpose is called a *profile*.

In the last years, some proposals to extend the UML for database design have been presented. In [2], "...a profile that extends the existing class diagram definition to support persistence modeling" is presented. This profile is intended to make objects persistent (save objects between sessions) in different storages: files, relational databases, object-relational databases, etc. In [11], 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 [9], as it is said in the back cover of the book, "...brings you exactly the information you need to begin working with the UML and take full advantage of the technology for high-quality database modeling and design". Finally, in [8] 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 one. Nevertheless, to the best of our knowledge, there is no proposal to extend the UML for MD modeling.

In this paper, we present an extension to the UML for MD modeling, as it 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.

Moreover, we show how to apply this extension in a well-known model-driven development tool, i.e., Rational Rose. This tool provides the Rose Extensibility Interface that allows us to customize and extend the menus, integrate new stereotypes and tagged values in the models, and run scripts that validate the correctness of the MD models.

The remainder of this paper is structured as follows: Section 2 summarizes the UML Extensibility Mechanism. Section 3 introduces the main MD concepts such as fact, dimension, and hierarchy level that our approach comprises. Section 4 proposes the new UML extension (stereotypes, tagged values, and constraints) for MD modeling. Section 5 shows how to apply our MD extension in Rational Rose. Section 6 presents the main conclusions. Finally, Section 7 introduces our future work.

2 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[1]. For example, the UML 1.4 [10] includes a standard profile for modeling software development processes and another one for business modeling.

A *stereotype*[2] is a model element that defines additional values (based on tagged val-

[1] "A profile is a stereotyped package that contains model elements that have been customized for a specific domain or purpose by extending the metamodel using stereotypes, tagged definitions, and constraints" [10].

[2] "A stereotype is a model element that defines additional values (based on tag definitions), additional constraints, and optionally a new graphical representation. All model elements that are branded by one or

ues), 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 represented as a string between a pair of guillemots ($\ll$ $\gg$), but it can also be rendered by a new icon.

A *tagged value*[3] 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*[4] 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. There are three common kinds of constraints: preconditions, postconditions, and invariants. Preconditions and postconditions are applied to operations: a precondition must be true at the moment that the operation is going to be executed, whereas a postcondition must be true at the moment that the operation has just ended its execution. An invariant is a constraint that states a condition that must always be met by all instances of the class, type, or instance. An invariant on a stereotype is interpreted as an invariant on all types on which the stereotype is applied.

A constraint can be defined by means of an informal explanation or by means of Object Constraint Language (OCL) [16, 10] expressions. The OCL is a declarative language that allows software developers to write constraints over object models.

3 Multidimensional conceptual modeling

In this section, we will summarize how the conceptual MD modeling approach followed in this paper [14] represents both the structural and dynamic parts of MD modeling. In this approach, main MD modeling structural properties are specified by means of a UML class diagram in which the information is clearly separated into facts and dimensions.

Dimensions and facts are considered by *dimension classes* and *fact classes* respectively. Then, fact classes are specified as composite classes in shared aggregation relationships of n dimension classes. Thanks to the flexibility of shared aggregation relationships that the UML provides, *many-to-many* relationships between facts and particular dimensions can be considered by indicating the 1..* cardinality on the dimension class role. For example,

more particular stereotypes receive these values and constraints in addition to the attributes, associations, and superclasses that the element has in the standard UML. Stereotypes augment the classification mechanism based on the built in UML metamodel class hierarchy; therefore, names of new stereotypes must not clash with the names of predefined UML metamodel elements or standard elements" [10].

[3]"Tag definitions specify new kinds of properties that may be attached to model elements. The actual properties of individual model elements are specified using Tagged Values. These may either be simple datatype values or references to other model elements. Tag definitions can be compared to metaattribute definitions while tagged values correspond to values attached to model elements. They may be used to represent properties such as management information (author, due date, status), code generation information (optimizationLevel, containerClass)" [10].

[4]"A constraint is a Boolean expression over one or several elements that must always be true. A constraint can be specified in several different ways (e.g., using natural language or a constraint language)" [10].

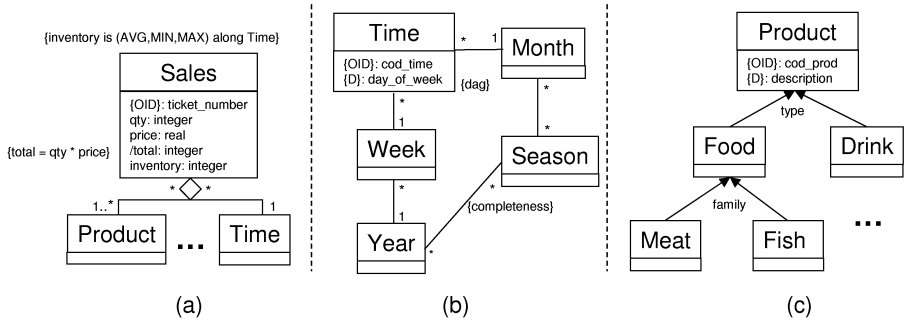

Figure 1: Multidimensional modeling using UML

on the left hand side of Figure 1, 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 nonadditive measures, additive rules are defined as constraints and are also placed in somewhere around the fact class. Furthermore, derived measures[5] can also be explicitly considered (constraint /) and their derivation rules are placed between braces in somewhere around the fact class, as can be seen in Figure 1.

This OO approach also allows us to define identifying attributes in the fact class, if convenient, by placing the constraint *{OID}* next to a measure name. In this way we can represent degenerate dimensions [5, 7], thereby providing other fact features in addition to the measures for analysis. For example, we could store the ticket and line numbers as other ticket features in a fact representing sales tickets, as reflected in Figure 1.

With respect to dimensions, every *classification hierarchy* level of a dimension 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). 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 (constraint *{D}*). These attributes are necessary for an automatic generation process into commercial OLAP tools, as these tools store this information in their metadata. The multiplicity *1* and *1..** defined in the target associated class role addresses the concepts of *strictness* and *non-strictness*. In addition, defining the *{completeness}* constraint in the target associated class role addresses the completeness of a classification hierarchy (see an example on the center of Figure 1). This approach considers all classification hierarchies non-complete by default.

[5] Although a derived measure can be completely derived from other measures and is therefore logically redundant, we obtain two advantages when using this kind of measure. On the one hand, the measure may be included to define a useful name or concept. On the other hand, the usual intent is that the measure should exist in the implementation to avoid the need for recomputation.

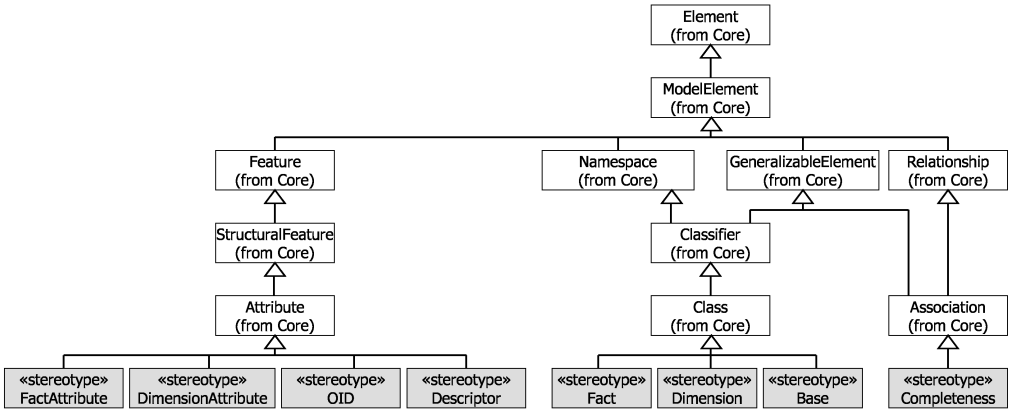

Figure 2: Extension of UML with stereotypes

The *categorization of dimensions*, used to model additional features for an entity's subtypes, is considered by means of generalization-specialization relationships. However, only the dimension class can belong to both a classification and a specialization hierarchy at the same time. An example of categorization for the Product dimension can be observed on the right hand side of Figure 1.

4 UML Extension for Multidimensional Modeling

According with [4], "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. This rules are used to determine whether a model is semantically consistent with itself". In this section, we summarize our UML extension for MD modeling following this structure:

- Description: A little description in natural language of the extension.
- Prerequisite Extensions: It indicates whether the current extension needs the existence of previous extensions.
- Stereotypes: The definition of the stereotypes.
- Well-Formedness Rules: The static semantics of the metaclasses are defined as a set of invariants defined by OCL expressions.
- Comments: Any additional comment, decision or example all written in natural language.

For the definition of stereotypes and tagged values, we follow the examples included in the UML specification [10]:

- 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) for the stereotype.
- Constraints: A list of constraints defined by OCL expressions associated with the stereotype, with an informal explanation of the expressions[6].
- Tagged values: A list of all tagged values that may be associated with the stereotype.

Regarding tagged value, we define them as follows:

- 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.

We have defined eight stereotypes: four stereotypes specialize the Attribute model element, three specialize the Class model element, and one specializes the Association model element. We have represented part of the UML metamodel[7] in Figure 2 to show where our stereotypes fit.

4.1 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 OID and FactAttribute, whereas a Dimension class consists of OID, Descriptor, and DimensionAttribute. Moreover, the hierarchy levels of a Dimension are represented by means of Base classes. Finally, a Completeness association is defined.

4.2 Prerequisite Extensions

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

[6] In OCL expressions, self, which can be omitted as a reference to the stereotype defining the context of the constraint, has been kept for clarity.

[7] All the metaclasses come from the Core Package, a subpackage of the Foundation Package.

4.3 Stereotypes

4.3.1 Fact

In this stereotype we do not define the following elements (but we use them in our MD approach) because they are defined in the UML metamodel and therefore, they are inherited by this stereotype:

- `name`: An attribute of `ModelElement`. It is an identifier for the `ModelElement`.
- `documentation`: A tagged value of `Element`. It is a comment, description or explanation of the element to which it is attached.

- Name: Fact
- Base class: Class
- Description: Classes of this stereotype represent facts in a MD model
- Icon: Figure 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 to Dimension classes:[8]
    self.allOppositeAssociationEnds->forAll(participant.oclIsTypeOf(Dimension))
- Tagged values: None

4.3.2 OID

In this stereotype we do not define the following elements (but we use them in our MD approach) because they are defined in the UML metamodel and therefore, they are inherited by this stereotype:

- `name`: An attribute of `ModelElement`. It is an identifier for the `ModelElement`.
- `documentation`: A tagged value of `Element`. It is a comment, description or explanation of the element to which it is attached.
- `type`: An association of `StructuralFeature`. Designates the classifier whose instances are values of the feature.
- `initialValue`: An attribute of `Attribute`. An expression specifying the value of the attribute upon initialization.

[8] allOppositeAssociationEnds is an additional operation defined in the UML specification [10]: "The operation allOppositeAssociationEnds results in a set of all AssociationEnds, including the inherited ones, that are opposite to the Classifier".

- Name: OID
- Base class: Attribute
- Description: Attributes of this stereotype represent OID attributes of Fact, Dimension or Base classes in a MD model[9]
- Icon: Figure 3.d
- Constraints: None
- Tagged values: None

4.3.3 FactAttribute

In this stereotype we do not define the following elements because they are defined in the UML metamodel: `name`, `documentation`, `type`, and `initialValue` (see OID). In addition, we do not define the tagged value `derived` because it is defined in the UML metamodel and therefore, they are inherited by this stereotype:

- `derived`: A tagged value of `ModelElement`. A true value indicates that the model element can be completely derived from other model elements and is therefore logically redundant.

- Name: FactAttribute
- Base class: Attribute
- Description: Attibutes of this stereotype represent attributes of Fact classes in a MD model
- Icon: Figure 3.e
- Constraints:
  - A FactAttribute can only belong to a Fact:
    self.owner.oclIsTypeOf(Fact)
  - If a FactAttribute is derived, then it need a derivation rule (an OCL expression):
    self.derived implies self.derivationRule.oclIsTypeOf(OclExpression)
- Tagged values:
  - derivationRule:
    - Type: UML::Datatypes::String
    - Multiplicity: 1
    - Description: If the attribute is derived, this tagged value represents the derivation rule

[9] In a Fact class, an OID attribute is needed to explicitly consider what is called *degenerated dimension*. In a Dimension or Base class, an OID attribute is needed to automatically generate the implementation of a MD model into commercial OLAP tools.

4.3.4 Dimension

In this stereotype we do not define the following elements because they are defined in the UML metamodel: `name`, and `documentation` (see Fact).

- Name: Dimension
- Base class: Class
- Description: Classes of this stereotype represent dimensions in a MD model
- Icon: Figure 3.b
- Constraints:
  - All attributes of a Dimension must be OID, Descriptor, or DimensionAttribute: self.feature->select(oclIsKindOf(Attribute))->forAll(oclIsTypeOf(OID) or oclIsTypeOf(Descriptor) or oclIsTypeOf(DimensionAttribute))
  - All associations of a Dimension with a Fact must be aggregations at the opposite end: self.association.association->forAll(associationEnd.participant.oclIsTypeOf(Fact) implies associationEnd.aggregation = #aggregate)
  - All associations of a Dimension with a Fact must not be aggregations at its end: self.association.association->forAll(associationEnd.participant.oclIsTypeOf(Fact) implies aggregation <> #aggregate)
  - A Dimension cannot be associated to another Dimension: self.allOppositeAssociationEnds->forAll(not participant.oclIsTypeOf(Dimension))
- Tagged values:
  - isTime:
    - Type: UML::Datatypes::Boolean
    - Multiplicity: 1
    - Description: Indicates whether the dimension represents a time dimension or not[10]

4.3.5 Descriptor

In this stereotype we do not define the following elements because they are defined in the UML metamodel: `name`, `documentation`, `type`, and `initialValue` (see OID). In addition, we do not define the tagged value `derived` because it is defined in the UML metamodel (see FactAttribute).

- Name: Descriptor
- Base class: Attribute
- Description: Attributes of this stereotype represent descriptor attributes of Dimension or Base classes in a MD model[11]
- Icon: Figure 3.f
- Constraints:

[10] The "Time dimension" is treated differently from the others in the OLAP tools.

[11] A descriptor attribute is needed to automatically generate the implementation of a MD model into commercial OLAP tools, since this attribute will be used as the default label in the data analysis.

  - A Descriptor attribute can only belong to a Dimension or Base: self.owner.oclIsTypeOf(Dimension) or self.owner.oclIsTypeOf(Base)

- Tagged values:
  - derivationRule:
    * Type: UML::Datatypes::String
    * Multiplicity: 1
    * Description: If the attribute is derived, this value represents the derivation rule

---

4.3.6 DimensionAttribute

In this stereotype we do not define the following elements because they are defined in the UML metamodel: `name`, `documentation`, `type`, and `initialValue` (see OID). In addition, we do not define the tagged value `derived` because it is defined in the UML metamodel (see FactAttribute).

- Name: DimensionAttribute
- Base class: Attribute
- Description: Attributes of this stereotype represent attributes of Dimension or Base classes in a MD model
- Icon: Figure 3.g
- Constraints:
  - A DimensionAttribute can only belong to a Dimension or Base: self.owner.oclIsTypeOf(Dimension) or self.owner.oclIsTypeOf(Base)
- Tagged values:
  - derivationRule:
    * Type: UML::Datatypes::String
    * Multiplicity: 1
    * Description: If the attribute is derived, this value represents the derivation rule

---

4.3.7 Base

In this stereotype we do not define the following elements because they are defined in the UML metamodel: `name`, and `documentation` (see Fact).

- Name: Base
- Base class: Class
- Description: Classes of this stereotype represent hierarchy levels in a MD model
- Icon: Figure 3.c
- Constraints:

- All attributes of a Base must be OID, Descriptor, or DimensionAttribute:
  self.feature->select(oclIsKindOf(Attribute))->forAll(oclIsTypeOf(OID) or oclIsTypeOf(Descriptor) or oclIsTypeOf(DimensionAttribute))
- A Base must have an OID attribute and a Descriptor attribute:
  self.feature->select(oclIsKindOf(Attribute))->exist(oclIsTypeOf(OID)) and self.feature->select(oclIsKindOf(Attribute))->exist(oclIsTypeOf(Descriptor))
- A Base can only be associated to another Base or another Dimension:
  self.allOppositeAssociationEnds->forAll(participant.oclIsTypeOf(Base) or participant.oclIsTypeOf(Dimension))
- A Base can only be child in one generalization:
  self.generalization->size <= 1
- A Base cannot simultaneously belong to a generalization/specialization hierarchy and an association hierarchy:
  (self.generalization->size > 0 or self.specialization->size > 0) implies (self.association->size = 0)

- Tagged values: None

4.3.8 Completeness

In this stereotype we do not define the following elements because they are defined in the UML metamodel: `name`, and `documentation` (see Fact).

- Name: Completeness
- Base class: Association
- Description: Associations of this stereotype represent complete associations[12]
- Icon: None
- Constraints:
  - The ends of a Completeness association can only be Dimension or Base classes:
    self.associationEnd.participant->forAll(oclIsTypeOf(Dimension) or oclIsTypeOf(Base))
- Tagged values: None

4.4 Well-Formedness Rules

4.4.1 Namespace

- All the classes in a MD model must be Fact, Dimension, or Base:[13]
  self.allContents->forAll(oclIsKindOf(Class) implies (oclIsTypeOf(Fact) or oclIsTypeOf(Dimension) or oclIsTypeOf(Base)))

---

[12] A complete association means that all members belong to one higher-class object and that object consists of those members only.

[13] allContents is an additional operation defined in UML specification [10]: "The operation allContents results in a Set containing all ModelElements contained by the Namespace".

Fact	Dimension	Base	OID	Fact Attribute	Descriptor	Dimension Attribute
3.a	3.b	3.c	3.d	3.e	3.f	3.g

Figure 3: Stereotype icons

	Fact	**Dimension**	**Base**
Fact	-	Aggregation	-
Dimension	-	-	Association Generalization
Base	-	-	Association Generalization

Table 1: Relationships between new elements

4.5 Comments

Code generators for specific OLAP tools are expected to specify additional tagged values for class and association stereotypes.

Table 1 shows the relationships we can find between the new elements we have defined. The table is not symmetric because ... *(Te lo cuento cuando nos veamos.)*

For the use of derived attributes, we use the tagged value `derived` in FactAttribute, Descriptor, and DimensionAttribute. The UML Notation Guide [10] indicates that the definition of the derived attribute can be expressed as a constraint string placed near the derived element. However, we have chosen to add a tagged value called derivationRule that stores the definition of the derived attribute as an OCL expression. We think that out approach is "more natural" and eases the exportation of MD conceptual models into commercial OLAP tools.

(Falta comentar algo sobre la aditividad.)

5 Using Multidimensional Modeling in Rational Rose

Rational Rose is the world's leading visual modeling tool. It allows OO modeling because it supports the UML. Rational Rose is extensible by means of add-ins, that allow to package customizations and automation of several Rose features through the Rose Extensibility Interface (REI) [12] into one package. 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.

In this section, we present an add-in we have developed, that allows us to use in Rational

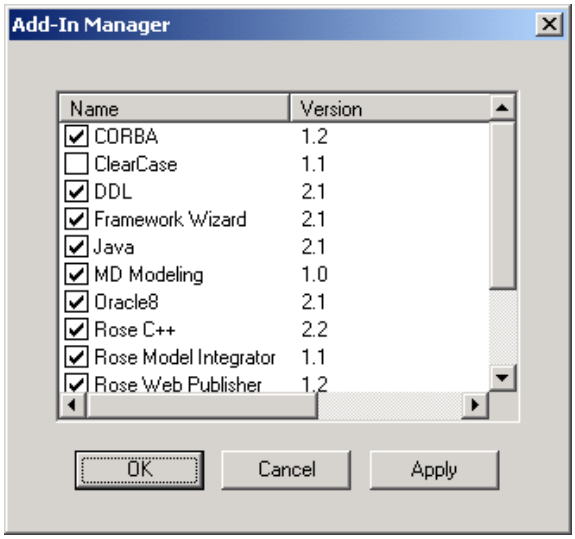

Figure 4: Rational Rose Add-In Manager

Rose the stereotypes and tagged values we have presented. Therefore, we can use this tool to easily model MD conceptual models.

Our add-in customizes the following elements:

- Menu item: We have added the new menu item MD Validate in the menu Tools. This menu item runs a Rose script that validates a MD model: our script checks all the constraints we have presented in Section 4.
- Stereotypes: We have defined the stereotypes we have presented in Section 4.
- Properties: We have defined the tagged values we have presented in Section 4.

Figure 4 shows the Add-In Manager that allows us to activate or deactivate add-ins. In Figure 4 our MD Modeling add-in is shown activated.

The best way to understand our extension is to show a tangible example. Figure 5 shows a MD conceptual model of the well-known example "Grocery" as described[14] in Chapter 2 of [7]. 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 clearness, the attributes of the classes are hidden. However, we can notice the FactAttribute list of Sales in the list of the browser (left hand panel in Figure 5): quantity_sold, dollar_revenue, dollar_cost, and customer_count.

[14] This example represents the data warehouse of a company that has 500 large grocery stores spread over three-state areas. Each store has roughly 60 000 individual products with a full complement of departments (frozen foods, dairy, meat, bakery, liquor, etc.). Furthermore, products are usually sold under different promotion conditions (price reduction, newspaper advertisement, etc.).

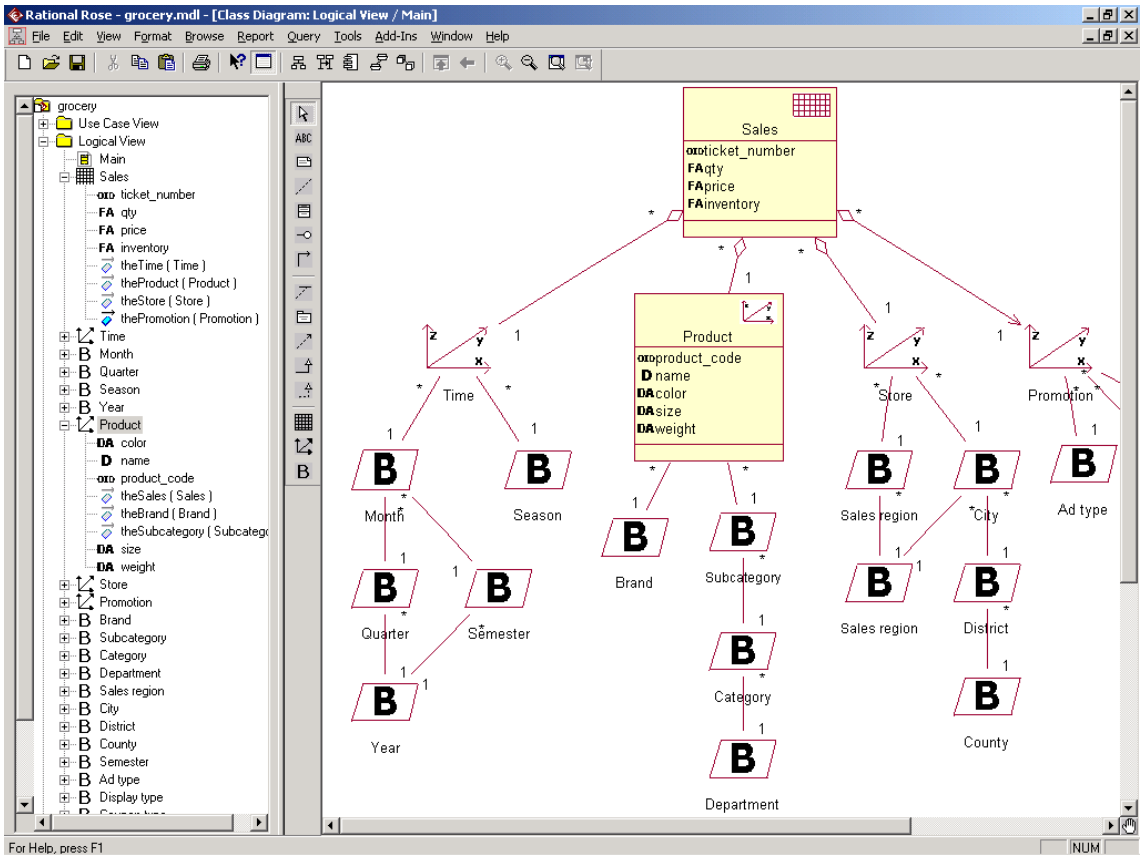

Figure 5: Multidimensional modeling using UML

Figure 6: Add-in configuration files for Rational Rose

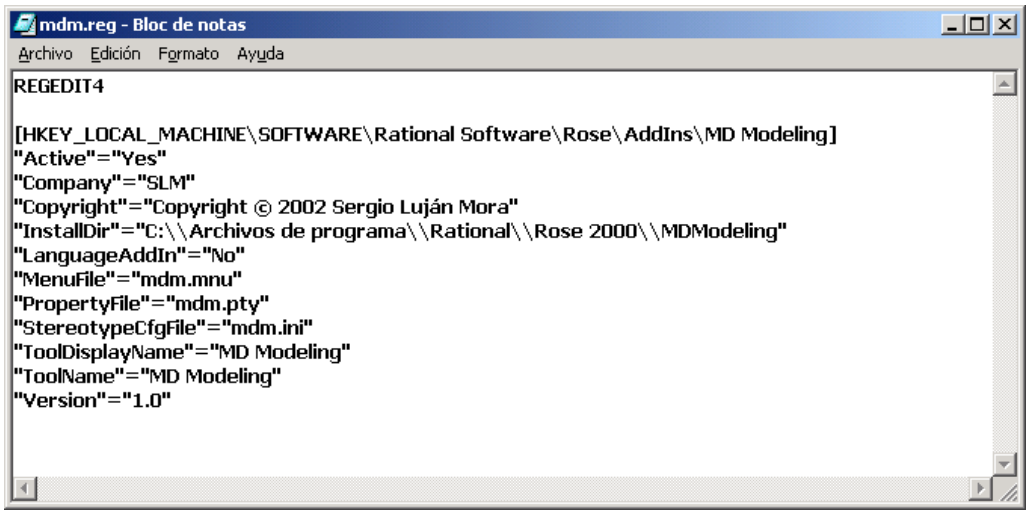

Figure 7: Installation of the add-in in Rational Rose

In Figure 6 we show the files that constitute our add-in for Rational Rose. The file mdm.reg displayed in Figure 7 contains the instructions to install our add-in in Rational Rose[15].

The file mdm.ini is a stereotype configuration file that contains our MD stereotypes (Figure 8). To graphically distinguish model elements of different stereotypes, each stereotype can have a graphical representation. Thus, for each stereotype, there may be four different icons (all the icons are stored in the folder called stereotypes):

- A diagram icon (Metafile field). It is displayed in the model diagram.
- A small diagram toolbar icon (SmallPalleteImages field). It is displayed in the toolbar (see the vertical toolbar in the middle of the Figure 5).
- A large diagram toolbar icon (MediumPalleteImages field). It is also displayed in the toolbar. There are two icons for the toolbar because it is possible to select the size of the toolbar buttons in Rational Rose.

[15]It actually registers our add-in Microsoft Windows' register.

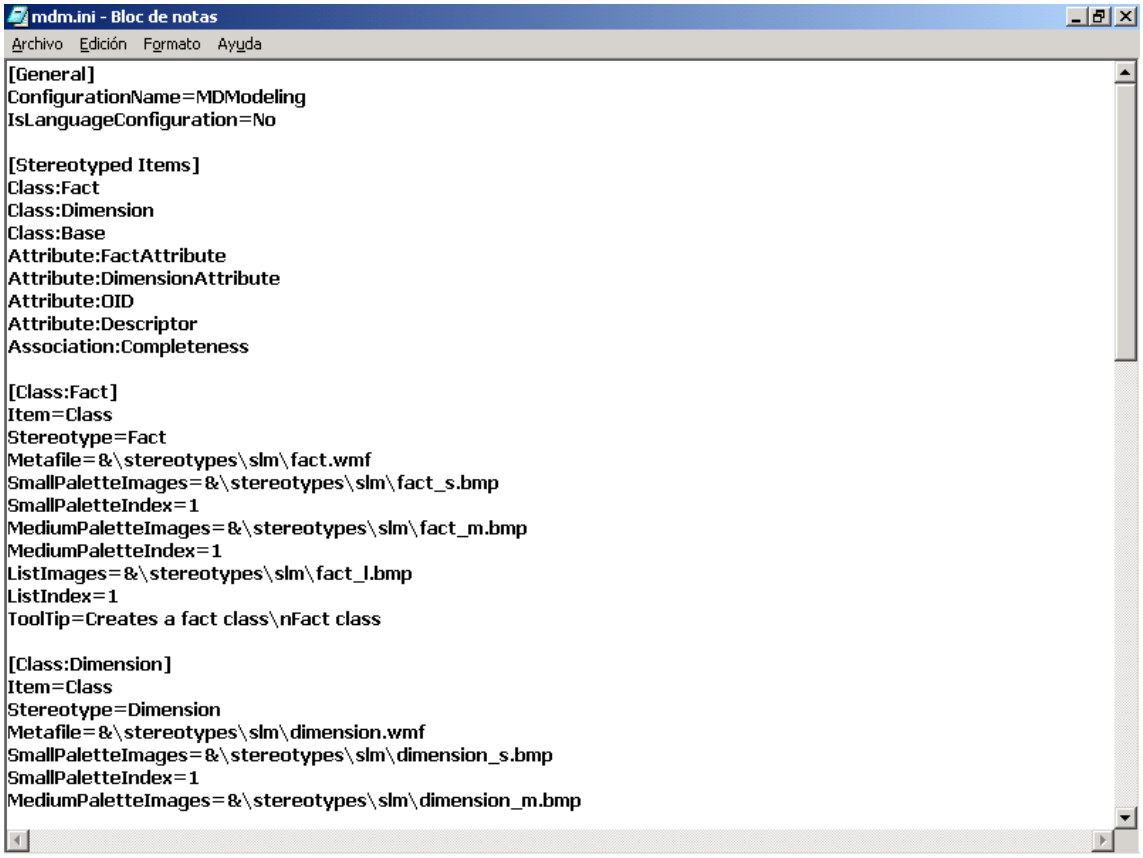

Figure 8: File defining stereotypes in Rational Rose

- A list view icon (ListImage field). It is displayed in the list of the browser (see left hand panel in Figure 5).

Furthermore, a stereotype can be displayed in different manners in Rational Rose. The four possible representations of a Fact are shown in Figure 9:

Use the None, Label, Decoration, and Icon options to control the display of stereotypes in diagrams: icon (the stereotype icon is displayed), decoration (the stereotype decoration is displayed inside the element), label (the stereotype name is displayed and appears inside guillemots), and none (the stereotype is not indicated).

Each add-in can optionally supply its own property file that defines a name space for its properties (tagged values) and a tab in the specification editor to hold the custom tool, sets, and properties. The file mdm.pty contains our defined model properties. For example, Class Specification for Time dimension is shown in Figure 5. We can notice the tab MD Modeling that contains the tagged value for a Dimension class: isTime.

Finally, an add-in can also extend or customize Rational Rose menus. The file mdm.mnu contains our customization of Rational Rose menus: we only add a new menu item called MD Validate in the Tools menu.

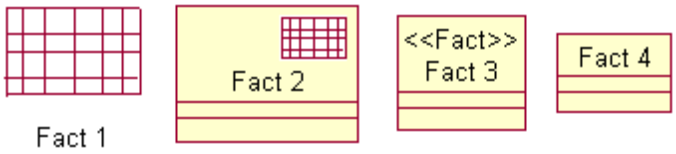

Figure 9: Possible representations of stereotypes in Rational Rose

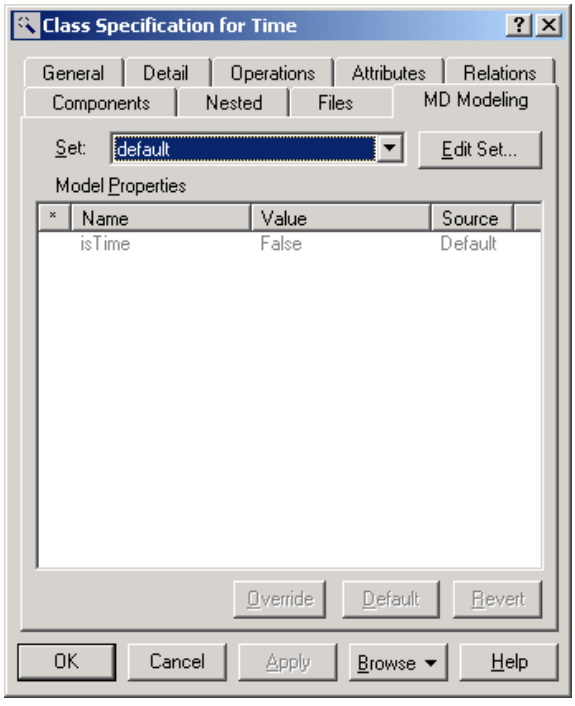

Figure 10: Properties (tagged values) in Rational Rose

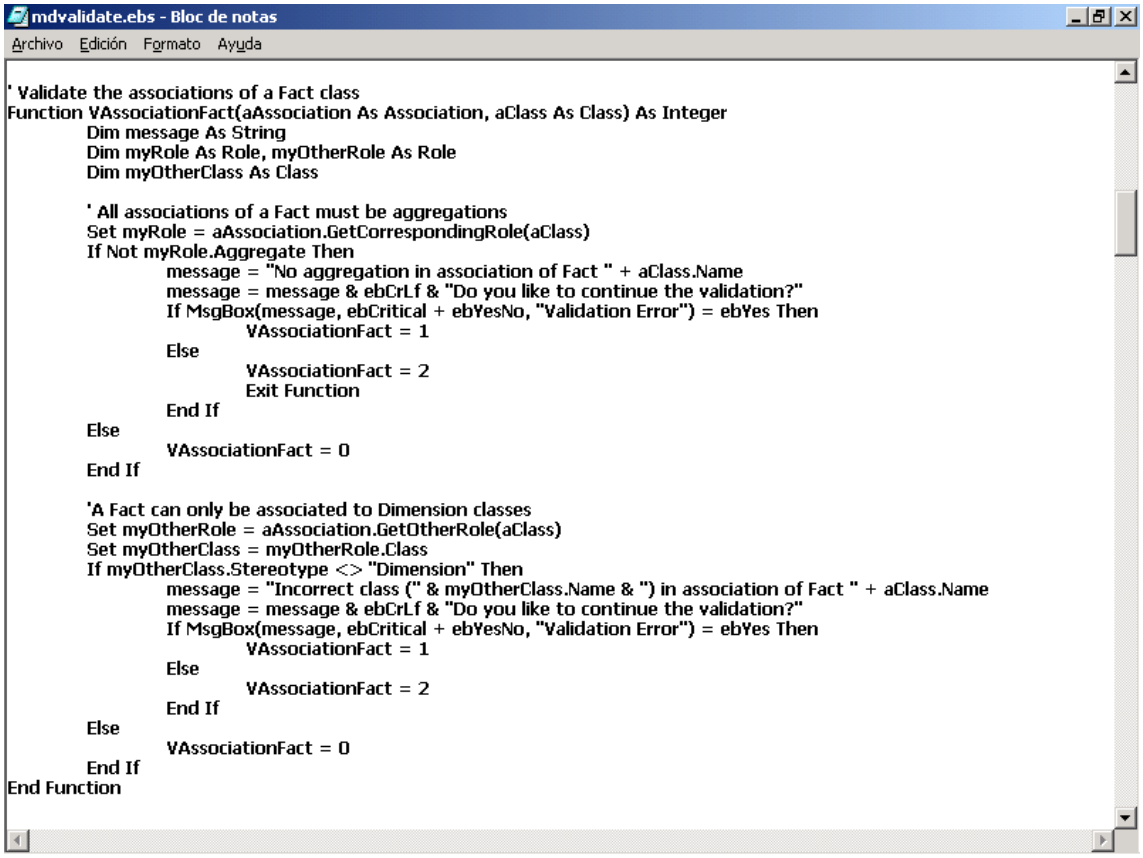

Figure 11: Validation script

This new menu item executes a Rose script, that is stored in mdmvalidate.ebs[16]. This script validates the correctness of a MD model: it checks all the constraints we have presented in Section 4. For example, a fragment of this script can be seen in Figure 11. In this fragment we can notice the function VAssociationFact, that validates the associations of a Fact. It checks the OCL constraints we have previously presented:

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

6 Conclusions

(Contar lo maravilloso que es UML y lo fantástico que es nuestro modelo para representar las propiedades multidimensionales).

[16]It is also possible to compile a script source (.ebs extension) and create a compiled script (.ebx extension).

With the MD modeling profile for UML, the UML fully supports MD modeling needs. It allows the support of software development and MD modeling with one unified language. Using this profile with Rational Rose unifies software development teams with a single, shared tool. It allows us to face up to the particular design challenges that MD modeling poses.

7 Future work

- Extend UML to represent the dynamic part (cube classes).
- Define a methodology for MD modeling.
- Implement the automatic exportation of MD models into ROLAP tools, such as Informix Metacube, from Rational Rose.
- Propose a method to translate a MD model defined in UML into object-relational and object-oriented databases.

References

[1] A. Abelló, J. Samos, and F. Saltor. Benefits of an Object-Oriented Multidimensional Data Model. In K. Dittrich, G. Guerrini, I. Merlo, M. Oliva, and E. Rodriguez, editors, *Proc. of the Symposium on Objects and Databases in 14th European Conference on Object-Oriented Programming (ECOOP'00)*, volume 1944 of *LNCS*, pages 141–152. Springer-Verlag, 2000.

[2] S. Ambler. Persistence Modeling in the UML. Software Development Online. Internet: http://www.sdmagazine.com/documents/s=755/sdm9908q/, August 1999.

[3] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language: User Guide.* Object Technology Series. Addison-Wesley, 1999.

[4] J. Conallen. *Building Web Applications with UML.* Object Technology Series. Addison-Wesley, 2000.

[5] W. Giovinazzo. *Object-Oriented Data Warehouse Design. Building a star schema.* Prentice-Hall, New Jersey, USA, 2000.

[6] M. Golfarelli and S. Rizzi. A methodological Framework for Data Warehouse Design. In *Proc. of the ACM 1st Intl. Workshop on Data warehousing and OLAP (DOLAP'98)*, pages 3–9, Washington D.C., USA, 1998.

[7] R. Kimball. *The data warehousing toolkit.* John Wiley, 2 edition, 1996.

[8] E. Marcos, B. Vela, and J. M. Cavero. Extending UML for Object-Relational Database Design. In Martin Gogolla and Cris Kobryn, editors, *Proc. of the 4th Intl. Conference UML 2001*, volume 2185 of *LNCS*, pages 225–239. Springer-Verlag, 2001.

[9] E.J. Naiburg and R.A. Maksimchuk. *UML for Database Design*. Object Technology Series. Addison-Wesley, 2001.

[10] Object Management Group (OMG). Unified Modeling Language Specification 1.4. Internet: http://www.omg.org/cgi-bin/doc?formal/01-09-67, September 2001.

[11] Rational Software Corporation. The UML and Data Modeling. Internet: http://www.rational.com/media/whitepapers/Tp180.PDF, 2000.

[12] Rational Software Corporation. *Using the Rose Extensibility Interface*. Rational Software Corporation, 2001.

[13] C. Sapia, M. Blaschka, G. Höfling, and B. Dinter. 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*, pages 105–116. Springer-Verlag, 1998.

[14] J. Trujillo, M. Palomar, J. Gómez, and Il-Yeol Song. Designing Data Warehouses with OO Conceptual Models. *IEEE Computer, special issue on Data Warehouses*, 34(12):66–75, 2001.

[15] N. Tryfona, F. Busborg, and J.G. Christiansen. starER: A Conceptual Model for Data Warehouse Design. In *Proc. of the ACM 2nd Intl. Workshop on Data warehousing and OLAP (DOLAP'99)*, Kansas City, Missouri, USA, 1999.

[16] J. Warmer and A. Kleppe. *The Object Constraint Language. Precise Modeling with UML*. Object Technology Series. Addison-Wesley, 1998.