Formalizing the Model Transformation Using Metamodeling Techniques

Carlos G. Neil [1], Claudia Pons [2]

[1]Universidad Abierta Interamericana, Facultad de Tecnología Informática
Buenos Aires, Argentina
carlos.neil@vaneduc.edu.ar

[2] Universidad Nacional de La Plata, LIFIA
Buenos Aires, Argentina
cpons@info.unlp.edu.ar

Abstract. This paper establishes a formal connection among data models. It applies Meta Object Facility (MOF), based on metamodeling techniques to represent the translation, by means of an algorithm, from the temporal Entity-Relationship model into the temporal multidimensional model. MOF class diagrams and their corresponding OCL rules were used to establish constraints to the metamodel, which implemented in a CASE tool will make it possible to keep the model consistency.

Keywords: Metamodel, Datawarehouse, Entity-Relationship Model, MOF, UML.

1. Introduction

Metamodeling is a technique frequently used in software designing, which permits to describe the basic abstractions to define models and their relationships. The Meta Object Facility [10] provides a framework to give support to different types of metadata and it can be used to define different information models. This characteristic allows designers to define models that differ from the philosophy or details of the initial model; in this context, the MOF is considered a meta-metamodel since it used to define metamodels, such as that of the UML language [18]. The data model architecture of the MOF matches up with a meta-metamodel consisting of a four-stage architecture. The MOF is used to define the structure and semantics of metamodels for both specific and general domains. The MOF being an object-oriented model, it is also suitable for defining object-orient or more general metamodels; for example the main aspects of the Entity-Relationship schema may be represented by means of MOF class diagrams [6]. The MOF is also used to define specific metamodels for databases, datawarehouse and model transformation.

At present, the Entity-Relationship model [4] is commonly used and it has had a huge impact on database modelling; many of the information systems implemented by companies over the last decades are relational and their documentation is based on Entity-Relationship schemes. Datawarehouse is a subject-oriented, time varying, non-volatile collection of data that is used in organizational decision-making [17]. In a datawarehouse, the dimensions determine the granularity adopted for representing facts and the hierarchy in the dimensions determines how the instances can be aggregated and selected for the decision-making processes [2]. In [5] a semiautomatic algorithm for building a conceptual datawarehouse model from an Entity-Relationship model was presented, which was broaden in [8] using temporal entity-relationships models.

Our research consists in linking the temporal Entity-Relationship model with the temporal multidimensional model, by means of an approach to MOF metamodeling; we will present a MOF Model for both schemes, which is similar to the modelling of UML by means of MOF [6], where class diagrams are specified and complemented by invariant expressed in OCL [13]. Besides, we will consider constraints in connection with the transformation, using a recursive algorithm, from temporal Entity-Relationship scheme into the temporal multidimensional model by means of MOF class diagrams and their respective OCL rules.

There is a big amount of CASE tools that make it easy to create and manipulate UML diagrams. Many of these tools also provide automatic generators of codes and reverse engineering of existing software systems. However, the support provided by these tools is not enough to validate the models on the designing stage [16]. A well-developed semantics is an essential prerequisite to build CASE tools with advanced validation characteristics. Particularly, if a CASE tool adopts the algorithmic transformation process, the constraints imposed over the metamodel will enable the resulting model to remain consistent.

This paper is structured as follows: In chapter 2, we present a multidimensional model. In chapter 3, we explain model transformation. In chapter 4, we explain the metamodel by means of class diagrams. In chapter 5, we express constraints by means of OCL rules. In chapter 6, we briefly explain the related research, and finally, in chapter 7, we present our conclusion and future research.

2. Temporal Multidimensional Model

A multidimensional scheme is made up of facts, measures, dimensions and hierarchies. A fact represents point of interest for the decision-making process, model the events that occur in the company. The measures are attributes that describe the fact from different points of view. The dimensions determine the granularity adopted for representing facts, and the hierarchy in the dimensions determines how the instances can be aggregated and selected for the decision-making process. The temporal multidimensional includes, apart from the main fact for analysis, temporal schemes that although they will not belong to the hierarchy in the dimensions, they will register the variation of certain attributes o relationships that will vary in time. The resulting conceptual scheme unify in only one model, the multidimensional and

temporal scheme to register and analyze the temporal variations and queries about the multidimensional structure.

Despite the fact that commercial object-oriented databases are available, the relational database technology for data storage tends to be used more frequently because of its maturity. A datawarehouse applied to a standard relational database administrator system is called ROLAP (OLAP Relational). These servers store data in a relational database, apart from supporting SQL extensions, special accesses and methods adopted to make the multidimensional model and its is functions more efficient [3]. In a relational architecture data are organized in star or snowflake schemes; the first one consists in a main fact table and several denormalized dimension table, interest measures are stored in the fact table. The normalized version of the star scheme is the snowflake scheme in which every aggregation level has its own dimension table [19]. In this work, we will limit ourselves to describe conceptual models that are independent from implantation.

3. Model Transformation

The transformation methodology involves a series of steps that we will explain in detail later. But, in short, it consists in applying an algorithm that has temporal Entity-Relationship model as its input and a temporal multidimensional model as its output.

To apply the recursive algorithm we transformed the Entity-Relationship diagram (Fig. 1) into a temporal Entity-Relationship model (Fig. 2). The multi-valued attribute will turn into a weak entity with a temporal relationship (named T) and the temporal relationship will turn into an entity with binary relationships (named T) related to the participating entities [8]. In cases where it is advisable to preserve a future hierarchy we suggest keeping both relationships (the instantaneous relationship and the temporal one).

In the example of Fig. 2, we keep the relationship between Supplier and the Place.

3.1. Building the Fact Scheme from the Temporal Entity-Relationship Model

A conceptual scheme of a datawarehouse will derive from a temporal Entity-Relationship model and it will include temporal aspects in its design. The methodology for building a schema of multidimensional facts consists of the following steps:

- Define facts
- For each fact:
  a) Building attributes graph
  b) Pruning and grafting the attribute graph
  c) Defining dimensions
  d) Defining fact attributes
  e) Defining hierarchies.

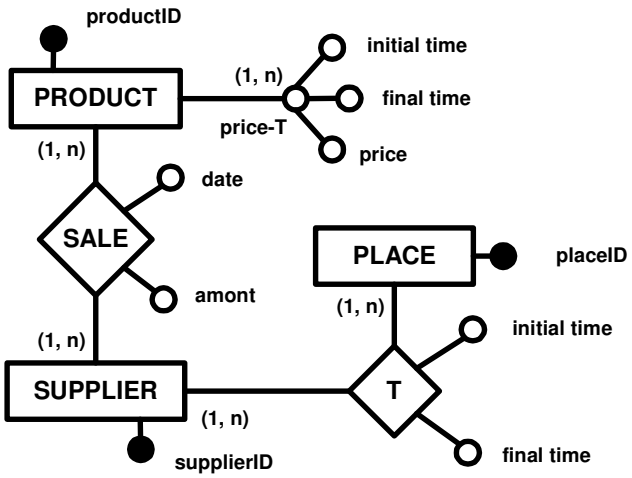

Fig. 1. Temporal Entity-Relationship Diagram

3.2. Building the Temporal Graph

The facts, as concepts of primary interest for decision-making process, correspond to events that occur dynamically in real life. These can be represented in the temporal Entity-Relationship model by means of an entity F or by means of an n-aria relationship R between entities E_1... E_n .

Given an area of interest of a temporal Entity-Relationship model and an entity F that belongs to it, we will call attributes graph to the graph in which:

- Each vertex corresponds to an attribute - simple or compound - of the Entity-Relationship model.
- The root corresponds to the identifier of F.
- For each vertex v, the corresponding attribute functionally determines all the attributes corresponding to the descendants of v.

Definition 1: Given a vertex v marked with a T, it is called terminal vertex if it does not have descendants that are identifiers of an entity.

Definition 2: Given a vertex v marked with a T, it is called non-terminal vertex if it has descendants that are identifiers of an entity.

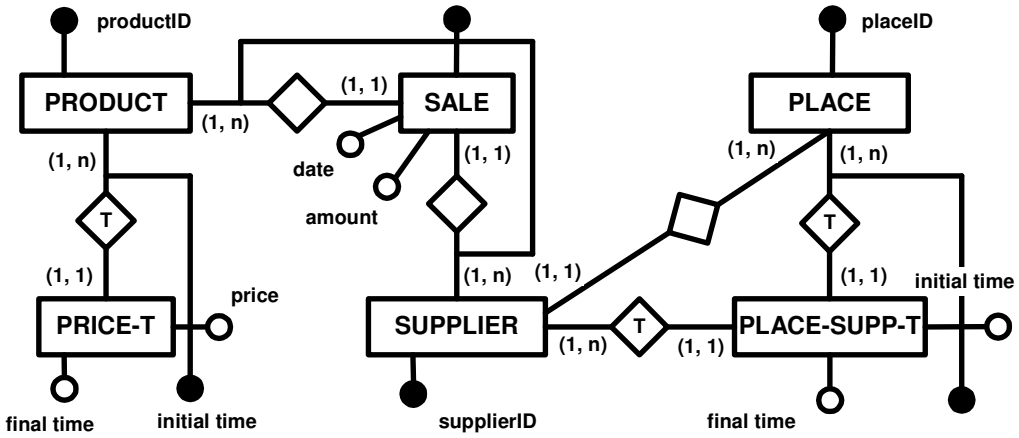

Fig. 2. Modified temporal Entity-Relationship diagram

In addition

- Each terminal vertex v will correspond to a temporal attribute.
- Each non-terminal vertex v will correspond to a temporal relationship.

The temporal vertexes represent schemas that have, as a focus of interest, the variation of attributes and relationships in accordance with time, at are related to the fact schema being of interest for the decision-making process. The attributes graph will be used in for building a facts schema corresponding to F. Given an identifier (F) that indicates a group of attributes that identifies the entity F, the attributes graph (Fig. 3) can be built semi-automatically by means of the application of the following recursive procedure:

```
root = newVertex(identifier(F));

//newVertex (<attributeSet>) returns a new vertex

//labeled with  the concatenation of the names of the

//attributes in the set

translate(F, root);

where

translate(E, v):

//E is the current entity, v is the current vertex
```

```
{ for each attribute a ∈ E | a ∉ identifier(E)  do
addChild (v, newVertex({a}));
//adds  child a to vertex  v
for each  entity G connected to E by relationship R
| card-max(E, R)= 1  or R is temporal  do
//Temporal relationships and attributes are considered
          { for each  attribute  b ∈ R do
          addChild (v, newVertex({b}));
          next = newVertex(identifier(G));
          addChild (v, next);
          translate(G, next);
}
}
```

When we amplified the Entity-Relationship model with temporal aspects, the varying attributes and relationships turn into entities linked to relationships marked with a T of the type x-to-many (card-max(E, R) > 1), so they cannot be included in the hierarchy to make aggregations. The line of dots in the attributes graph shows this peculiarity.

Probably all the attributes represented in the graph are not of interest for the data warehouse. For this reason, this can be pruned and grafted to eliminate the unnecessary details. The dimensions determine how fact instances can be aggregated in a significant way for the decision-making process. These must be chosen in the attributes graph among the son vertexes of the root. Measures are defined by means of the application, to the attributes of the graph, of aggregation functions that operate on all of the instances of F belonging to each primary fact instance. The last step in the construction of fact schemes is the definition of hierarchies on dimensions. Along these, the attributes must be ordered in the graph so as one a-to-one relationship is placed between each vertex and its descendants. The inclusion of attributes and temporal relationships (these are linked with each other by means of lines of dots) needs a special consideration in the transformation of the fact schema: these will not be part of the hierarchy for the operations of roll-up and drill-down, they will only enable to evaluate how certain attributes and relationships have varied as time passes by. They form what it is called non-strict hierarchies [15]. In the Fig. 4 the resulting scheme is shown.

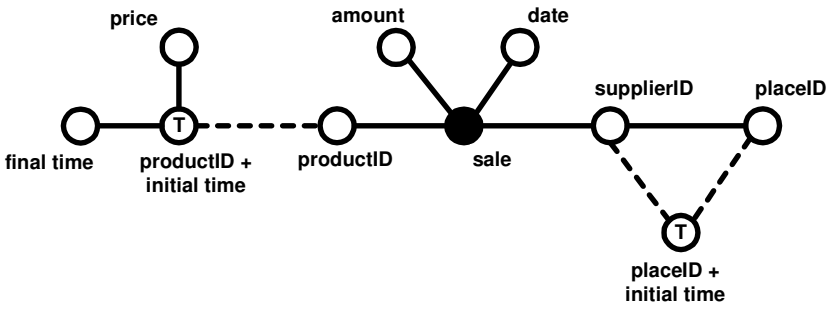

Fig. 3. Attribute Graph

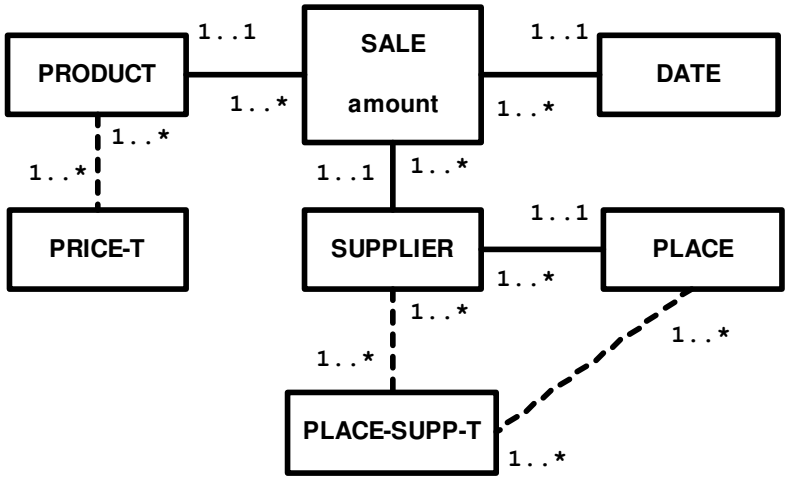

Fig. 4. Temporal Multidimensional Scheme.

4. MOF Class Diagrams

In Fig. 5 we describe, in a same figure, the metamodel for the temporal Entity-Relationship model and for the temporal multidimensional model, which we will explain below.

An `ERTSchema` consists, at least, in one `Entity` object and cero or more `Relationship` objects. These Entity objects have one or more than one `Attribute` object, which in turn, have an associated `Datatype`. The `Relationship` objects may have cero or more `Attributes` objects, also with an associated `Datatype`. The `Entity` objects may be temporal, that is, they may have temporal attributes. In this case, every `Entity` object is linked with an

`Interval` object that determines the validity interval of the attribute's value. The temporal relationships have no attributes and are linked with an Interval object. A `Relationship` object is related to with two `RelationshipEnd` objects, each of them being linked with an `Entity` object.

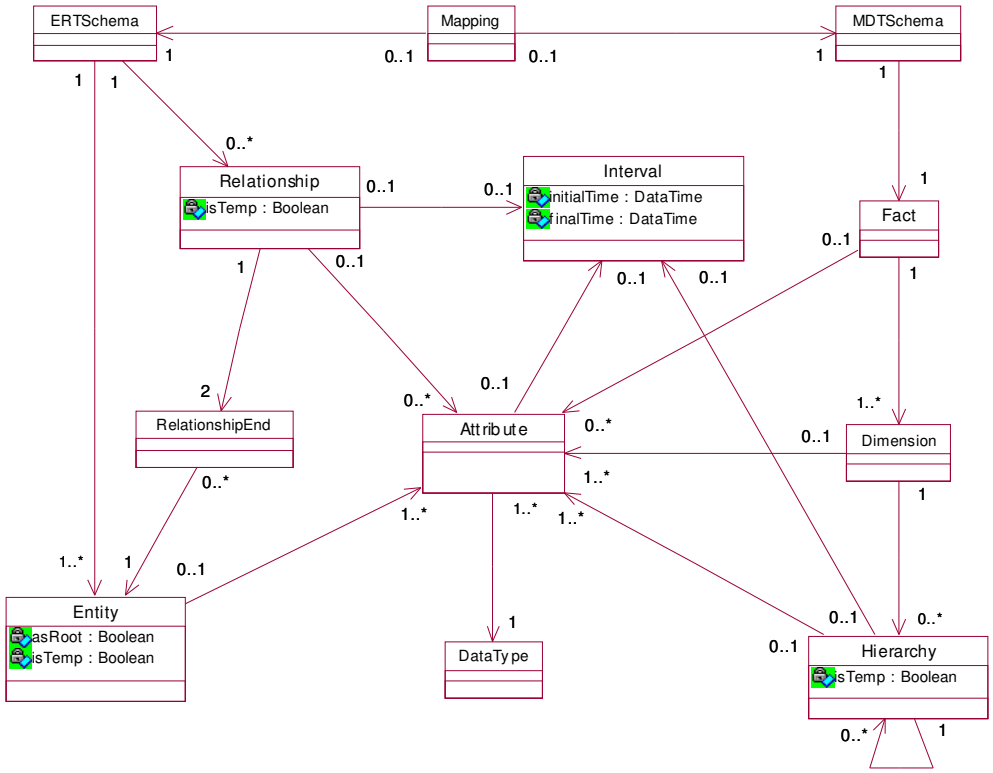

Fig. 5. Metamodel of Transformation.

A MDTSchema object consists in `Fact` object, which, in turn, it is related to one or more than one `Dimension` object, which, in turn, it is related to cero or more `Hierarchy` objects. A `Hierarchy` object may be linked with cero or more objects of the same type. A `Fact` object may have cero or more `Attribute` objects. Besides, both the `Dimension` and the `Hierarchy` objects have one or more than one `Attribute` object. The `Hierarchy` may be temporal. If it is derived from a temporal `Relationship`, no `Attribute` objects are linked with it. But it will be linked with an `Interval` object that determines the validity interval; if the

`Hierarchy` is derived from a temporal `Entity`, then it will linked with an `Attribute` object, which in turn, will be linked with an `Interval` object.

The transformation is expressed in the metamodel by means of the `Mapping`, together with two links specifying that a `Mapping` object is perfectly linked with an `ERTSchema` object and a `MTDSchema` object.

All the classes inherit the `name` attribute from a `Named` superclass, which is not displayed in the Figure.

5. Constraints to the Metamodel

In the object-oriented model, a graphic like the class diagram is not enough to achieve an accurate and unambiguous specification [11] [12]. There is a need to describe the additional constraints to the objects of the model. Many times, those constraints are described by means of a natural language. However, in practice, they frequently become ambiguous. To avoid these ambiguities, formal languages have been developed. Although they are suitable for people having an important maths background, the disadvantage is that they are difficult for the average system modeller. The OCL has been created to close that gap. It is a formal and easy- to-read-and-write language and provides extra information about the models used in the object-oriented development. It is a declarative language and without side effects. The state of an object does not change after having evaluated by an OCL expression. Every expression is written in the context of a class which has been defined in UML model and it defines a group of associated operation. In the data models (temporal database and datawarehouse) there is a series of constraints related to the application domain, which are nor frequently recorded. When one works with temporal ranks, determining constraints that will prevent the data established in the ranks according to the valid periods of time from being overlapped becomes a mechanism to keep the stored data integrity.

The constraints may be imposed both over the model and the metamodel. Next, we will show, as examples, a series of constraints applied in the metamodel (Fig. 5) using OCL sentences.

5.1. Constraints Over the Value of the Name Attribute

We can create constraints establishing that facts, dimensions and hierarchies have no attributes with the same name:

Two entities (or relationships) belonging to the same Entity-Relationship scheme cannot have the same name.

```
Context ERSTchema inv UniqueNameAttribute:
entity -> forAll (e1,e2 | e1.name = e2.name
```

```
implies e1 = e2)
```

The facts (dimensions and hierarchies) belonging to the same multidimensional scheme cannot have the same name.

```
Context MDTSchema inv NameFactUnique:
fact -> forAll (e1, e2 | e1.name = e2.name
implies e1 = e2)
```

The names of the attributes entities (and relationships) are unique

```
Context entity inv nameAttributeEntity Unique:
attribute -> forAll (e1,e2 | e1.name = e2.name
implies e1 = e2)
```

5.2. Constraints Over the Validity Interval

The attributes and temporal relationships have always an interval in which their value is valid:

If the relationship is temporal it is linked with no attribute, but only with a validity interval

```
Context Relationship inv TemporalRelationship:
self.isTemp  = ´true´ implies
self.attribute -> IsEmpty() and
self.interval -> notEmpty()
```

The temporal entities have at least a temporal attribute that is linked with a validity interval.

```
Context Entity inv Temporalentity:
self.isTemp  = ´true´ implies
self.attribute -> exists (a | a.interval -> notEmpty())
```

5.3. Constraints On the Model Transformation

The syntax and semantic constraints between the Entity-Relationship model and its corresponding multidimensional model generated by means of the transformation algorithm are expressed by means of the OCL expressions en the context of the metamodel of transformation.

There is a temporal hierarchy for each temporal attribute of the entity; this hierarchy has the same name as the attribute, and both of them are linked with the same validity interval

```
Context Mapping inv Temporalhierarchyattribute:
self.ERTSchema.entity.attribute
-> forAll (a | a.isTemp implies
self.MDTSchema.hierarchies -> exists(h | h.isTemp =
´true´ and a.name = h.name and a.interval = h.interval))
to be defined:   a.isTemp == a.interval -> notEmpty() =
´true´ s.hierarchies ==  s.fact.dimension.hierarchy
```

Every temporal relationship is linked with a non-strict temporal hierarchy sharing the same name with the relationship which is linked with a validity interval.

```
Context Mapping inv RelationshipTemporalHierarchy:
self.ERTSchema.relationship ->
forAll (r | r.isTemp =´true´ implies
self.MDTSchema.hierarchies -> exists ( h | r.name =
h.name and  r.interval = h.interval))
```

The set of fact attributes is included in the set of attributes of the root entity of the ERTSchema.

```
Context Mapping inv Root:
self.ERTSchema.entity -> exists (e | e.asRoot = 'true'
and   self.MDTSchema.fact.attribute   ->   forAll   (a   |
e.attribute -> includes(a)))
```

6. Related Research

This paper is related to other research where metamodeling techniques were used. In [7] a formal connection was made between the Entity-Relationship model and the relational model, using based-MOF metamodeling techniques to represent both models and their transformation. In [6] the semantics and syntax of the Entity-Relationship model, the relational model and their transformation were studied. In

both research, constraints were imposed over the metamodels and their transformations using OCL. In [14] a framework to represent metadata about source data, target data, transformations, and the processes and operations that create and administer a datawarehouse were presented. In [1] the problem arising in the scheme translation between different data models was studied and a theoretical-graphic formalism was introduced making it possible to represent uniformly schemes and models to make a comparison among different data models and to describe the translation performance. In [9] a transformation from the multidimensional model into UML was presented, and the constraints imposed both over the model and the metamodel, by means of OCL sentences were described

7. Conclusions and Future Research

In this paper, we propose the formalization of the transformation of the temporal Entity-Relationship model into the temporal multidimensional one; we use MOF class-diagrams to represent both models and we impose constraints over them using OCL sentences. A pending research is related to the transformation of the temporal multidimensional model into the relational model using metamodeling techniques. On the other hand, the temporal model has temporal attributes represented as multi-valued complex attributes, which, in turn are transformed into weak entities and temporal relationships. The integration of both metamodels through a Metamodel of Transformation allowed as to formally express a number of consistency constraints between the input and the output models of the transformation algorithm; however additional formalizations will be required to specify the transformations unambiguously

References

1. Atzeni P, Torlone R., Schema translation between heterogeneous data models in a lattice framework. 6th IFIP TC-2 Working Conference on Database Semantics (DS-6), Atlanta, Georgia, May 30-June 2, 1995.
2. Agrawal R, Gupta A, Sarawagi S., Modeling Multidimensional Databases, Research Report, IBM Almaden Research Center, San Jose, California, 1995. ICDE '97
3. Chaudhuri S. and Dayal U., An Overview of Data Warehousing and OLAP Technology, ACM SIGMOD Record 26(1), March 1997.
4. Chen P. The Entity-Relationship Model – Toward a Unified View of Data. ACM Transactions on Database System. 1976
5. Golfarelli M., Maio D., Rizzi S., The Dimensional Fact Model: a Conceptual Model for Data Warehouses. International Journal of Cooperative Information Systems, vol 7, n.2&3, 1998
6. Gogolla Martin, Lindow Arne, Richters Mark, Ziemann Paul: Metamodel Transformation of Data Models, Workshop in Software Model Engineering 2002

7. Gogolla Martin, Lindow Arne: Transforming Data Models with *UML*, IOS Press, 2003
8. Neil Carlos, Ale Juan. A Conceptual Design for Temporal Data Warehouse. 31º JAIIO. Santa Fe. Symposium of Argentine Software Engineering. 2002
9. Neil Carlos, Pons Claudia. Aplicando Restricciones a un Datawarehouse Temporal Utilizando UML/OCL Congreso Argentino de Ciencias de la Computación e Informática 2003.
10. MOF. Mata Object Facility 1.3. OMG (1999) www.omg.org
11. Pons, Claudia, Baum, Gabriel and Felder Miguel. Foundations of Object Oriented Modeling Notations in a Dynamic Logic Framework. Fundamentals of Information Systems. Kluwer Academic Publisher. Chapter 1. 1999.
12. Pons, Claudia, Baum, Gabriel and Felder Miguel. Formal Foundations of Object Oriented Modeling Notations. 3th International Conference on Formal Engineering Methods, IEEE ICFEM 2000. UK.
13. OCL. Object Constraint Language - version 1.5. 2002
14. OMG, ed: The Common Warehouse Metamodel Specifications. OMG (2000). www.omg.org
15. Pedersen T. B. and Jensen C. S, Multidimensional Data Modeling for Complex Data. 1998. ICDE 1999:336-345.
16. Richters, Mark and Gogolla, Martin. Validating UML Models and OCL Constraints. In Andy Evans and Stuart Kent, editors, Proc. 3rd Int. Conf. Unified Modeling Language (UML'2000), pages 265-277. Springer, Berlin, LNCS 1939, 2000
17. Surajit Chaudhuri and Umesh Dayal, An Overview of Data Warehousing and OLAP Technology, ACM SIGMOD Record 26(1), March 1997.
18. UML. The Unified Modeling Language Specification – version 1.3. 2001
19. Vassilliadis P, Sellis T. A Survey on Logical Models for OLAP Databases. SIGMOD Record 28(4), pp. 64-69, December 1999