Detecting patterns and OLAP operations in the GOLD model

Juan Trujillo
Research Group of Logic Prog. and Information Systems
Dept. of Financial Economics
University of Alicante
E-03071. Alicante. Spain
+34 965 90 36 57
juan.trujillo@ua.es

Manuel Palomar
Research Group of Logic Prog. and Information Systems
Dept. of Languages and Information Systems. University of Alicante
E-03071. Alicante. Spain
+34 965 90 36 53
mpalomar@dlsi.ua.es

Jaime Gómez
Research Group of Logic Prog. and Information Systems
Dept. of Languages and Information Systems. University of Alicante
E-03071. Alicante. Spain
+34 965 90 33 17
jgomez@dlsi.ua.es

ABSTRACT

The aim of our GOLD model ([7],[9]) is to provide an Object Oriented (OO) Multidimensional data model supported by an OO formal specification language that allows us to automatically generate prototypes from the specification at the conceptual level, and therefore, to animate and check system properties. Within the context of OO modeling and automatic prototyping, the basis of the mapping from modeling to programming is focused on the identification of (cardinality and behavioral) patterns in the design phase and their relationships with the data model, process model and interface design.

The aim of this paper, therefore, is the identification of these patterns based on the relationships between the dimension attributes included in cube classes. These patterns will associate data together with OLAP operations and will allow us to have a concise execution model that maps every pattern of modeling into its corresponding implementation making users able to accomplish OLAP operations on cube classes. Furthermore, we extend the set of classical OLAP operations with two more operations (*combine*, *divide*) to allow us to navigate along attributes that are not part of any classification hierarchy.

1. INTRODUCTION

In multidimensional modeling (MD), the analysis of data is addressed by the dimension attributes, and the relationships between them, included in cubes or statistical tables, which determines the set of OLAP operations that users can accomplish.

With reference to conceptual approaches, in [11], dimension attributes are represented by a directed acyclic and weakly connected graph and are joined by one-to-one or many-to-one edges. In [4], a special binary relationship set (rolls-up relationship) is defined together with a directed acyclic graph (rolls-up graph) to consider classification hierarchies. These two approaches, however, do not contemplate either data dynamic aspects or behavior, and therefore, their conceptual models cannot contemplate OLAP operations permitted on cubes.

On the other hand, many multidimensional data models have been proposed recently such as [1], [2], [6], [10], [12], [16] and [17] (a comprehensive review of these can be found in [3]) in which basic structures to consider dimensions with their attributes (and their relationships) are defined as well as cubes or statistical tables to which a set of OLAP operations can be applied. Most of them, however, focus on either an OLAP query language such as [2], [12] and [17] or on a presumption about a subsequent implementation, such as [12] or [17]. Furthermore, **they are all lacking** in certain key issues in multidimensional modeling, such as derived measures, derived dimension attributes and the additivity of measures.

The GOLD model ([7], [9]), however, allows us to consider key issues in multidimensional modeling that are hardly considered by other models, such as derived measures, derived dimension attributes, the additivity of measures and multiple classification hierarchies. Furthermore, this model also provides an Object Definition Language (GOLD Definition Language, GDL), which is a multidimensional extension to the OASIS formal specification language ([14],[15]) (developed by the Technical University of Valencia) to exploit the advantages provided by the use of formal specification languages such as the fact that these languages can generally be interpreted and executed, and thus prototype generation can potentially be made automatic.

One of the main advantages of the OASIS language is that it allows us to automatically obtain an implementation of the system from the specification at the conceptual level [13]. This process of automatically generating code is based on a **pattern finite catalog** [5]. A concise execution model maps every pattern of modeling into its corresponding representation on the solution space (implementation). Thus, software prototypes that allow us to animate and check system properties can be obtained.

In this paper, therefore, we identify patterns within our GOLD model. They describe concrete scenes (situations) which occur over an over again in the context of OLAP operations. We start by identifying cardinality patterns based on the relationships between dimension attributes. We then identify behavioral patterns by defining the OLAP operations that users can apply to cube classes based on the cardinality patterns previously defined. This approach allows us to propose a standard mechanism to

address the peculiarities of OLAP operations from the conceptual level. Furthermore, we extend the set of classical OLAP operations with two more operations (*combine*, *divide*) to allow us to navigate along attributes that are not part of any classification hierarchy.

This paper is organized as follows. In section 2 we summary the GOLD model with the last extensions above-commented. In section 3, we identify different cardinality and behavioral patterns. Section 4 defines the set of OLAP operations that can be accomplished for a subsequent data analysis based on the patterns previously defined. Finally, in section 5, we draw some conclusions and section 6 sketches further research issues.

2. THE GOLD MODEL

In this section we summarize the basic concepts of the GOLD model ([7], [8] and [9]) and provide a brief outline of the latest achievements of previous versions. The GOLD model is based on the OO paradigm and a MDB schema is defined by *dimension classes* (DC), *fact classes* (FC), *cube classes* (CC) and *views* (V). *Fact classes* are specified as *composite classes* in an aggregation relation where *dimension classes* are the components. *Cube classes*, to which OLAP operations are applied to allow us to accomplish a subsequent data analysis, are then defined from these DC and FC, based on user requirements.

Unlike other models presented so far, the GOLD model considers derived measures and derived dimension attributes by defining complex predicates using both arithmetic operations and relational grouping functions. With respect to the additivity of fact attributes (only considered in [11]). The GOLD model provides Aggregation Patterns (AP) to represented it. Thus, fact attributes can be additive if the SUM operator can be applied along all dimensions, semi-additive if it is not additive along all dimensions, and non-additive if it is not additive along any of the dimensions.

Finally, relationships between dimension attributes are considered by a directed acyclic and weakly connected graph in which each edge represents a -to-one relationship between attributes. We can therefore distinguish between Attribute Roll-up Relation Paths (ARRP), and Attribute Classification Paths (ACP), depending on whether there is a classification hierarchy defined on dimension attributes along the path or not.

For another thing, users can query the database basic schema by *cube classes*. These *cube classes* will encapsulate not only data but also operations allowed on objects in the given class. Thus, OLAP operations (*roll-up, drill-down, slice-dice* and *pivoting*; and the extended *combine* and *divide*) that allow us to accomplish a subsequent data analysis are considered as public methods of *cube classes*. By applying these OLAP operations, new *views* are defined on *cube classes* that share (inherit) the same basic properties, i.e. OLAP operations can also be applied to these *views*.

One of the great advantages of our GOLD model is that we have linked it with an appropriate Object Definition Language (GDL) to define multidimensional conceptual schemas . This is why we have selected a subset of the OASIS formal specification language ([14], [15]). One of the main features of this language is that it provides an ODL to specify conceptual schemas based on the OO paradigm. Although we need not use all the potential that a formal OO specification language provides, we will extend OASIS to deal with the peculiarities of multidimensional databases. In this way, GOLD classes will be considered as OASIS classes with multidimensional capabilities.

In figure 1, it can be observed the specification of a fact class (*sales of products*) built using the aggregation operator provided by the GDL. Derivation rules allow us to calculate derived measures from other measures. It can also be noted that AP that describe the additivity of measures are mapped into static constraints. In this particular case, *product_price*, *qty_sold* and *total_price* are additive as they can be aggregated along all dimensions while *n_of_client* is semi-additive as it cannot be aggregated along the *product dimension*.

```
COMPLEX CLASS sales_products AGGREGATION OF product, store, time
Identification
   By_cod_sales  (KA)
Constant_attributes
   KA : string;  product_price, qty_sold, n_of_clients: nat;
Derived_attributes
   total_price: nat;
private_events
   new_ product  new;  del_product  delete;
constraints static
   product_price ∑ ALL,   qty_sold ∑ ALL,   n_of_clients c product,
   n_of_clients ∑ {time, store},   total_price ∑ ALL
derivation
   total_price = (sales_products.product_price * sales_products.qty_sold)
end_class
```

Figure 1. The class template for sales_product fact class

On the other hand, the specification of a *product dimension class* can be observed in figure 2. Note that the reserved words (Roll-up, Drill-down, Combine and Divide) within path sections allow us to distinguish between ARRP and ACP paths (section 3.1).

```
Class product
Identification
        By_cod_product : (KA)
Constant_attributes
KA, Group, Colour, Department, Manager, Mark_group, Type, Supplier, Family, Brand: string;
Weight: float;   Quantity: nat;
Derived_attributes
transport_cost        : float;
Attribute_Roll-up_Relation_paths
   KA= Roll-up brand | Roll-up type
   Brand = Drill-down KA
   type= Roll-up Mark_group |
     | Roll-up family | Drill-down KA
   Mark_group = Drill-down type
   family= Roll-up group | Drill-down type
   group = Roll-up department |
     | Drill-down family
   department = Drill-down group
Attribute_Classification_Paths
   KA= Combine color | Combine weight|
     | Combine quantity | Combine trans_cost
   color= Divide KA
   weight= Divide KA
   quantity = Divide KA
   type= Combine supplier
   supplier = Divide type
   trans_cost= Divide KA
   Department = Combine manager
   Manager = Divide department
private_events
  new_ product      new;  del_product    delete;
derivation
  transport_cost =quantity * weight
end_class
```

Figure 2. The class template for product dimension class

3. DETECTING PATTERNS AS BASIC ELEMENTS FOR AUTOMATIC PROTOTYPING

As we have stated before in the paper, our GOLD model will be able to generate a prototype for a subsequent implementation of it in an OO programming environment. In this sense, the translation of OO analysis/design to an OO software development environment is divided into three parts:

(i) On the one hand, we consider the basic structures of our model in the data model. Due to the peculiarities of

multidimensional modeling, special emphasis has been stated on the relationship between dimension attributes (types of paths and cardinality patterns). This data model is going to be programmed in the database using its Object Definition Language (GDL).

(ii) On the other hand, additional client-side code will be programmed in a particular programming language. It will be considered as the process model, and in a similar way to the data model, the code will be organized as software components mapping behavioral patterns.

(iii) Additionally, the implementation has to be presented to users in a friendly and comfortable way, this point is achieved by means of the interface design (presentation model) considered in the patterns.

The main idea for implementing the GOLD model will be the identification of those patterns in the conceptual model (OO design) that are going to be translated into software components. The study of these patterns, therefore, will be a crucial task.

3.1 Detecting types of paths within graphs and cardinality patterns

According to the last version of the GOLD model in [9], the relationship between dimension attributes is represented by a directed acyclic and weakly connected graph rooted in the key attribute of the dimension class. Let us suppose from now on that we are analyzing the *sales of products* in a great store-chain in different *stores* and at different *time*. Our multidimensional model, therefore, consists of a fact class (*sales of products*) and three dimension classes (*store*, *product* and *time*). Thus, dimensions with their attributes[1] can be observed in figure 3. It may also be observed that we distinguish two kinds of paths: ARRP and ACP.

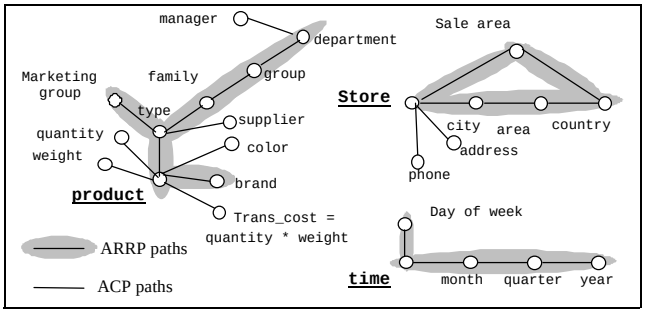

Figure 3. Directed acyclic graphs of dimensions

- ARRP represent classification hierarchies defined on dimension attributes along dimensions. These hierarchies determine how fact instances may be aggregated and selected significantly for the decision making process. Finally, the cardinality of the edges along these paths is many-to-one as each attribute is connected by a many-to-one relationship. For example, one group of products consists of many (several) families of products.

- ACP paths represent attributes that provide more characteristics about another attribute of the hierarchy within the dimension. Nevertheless, attributes within these ACP cannot be used for aggregation, as this would not make any sense. For example, it would not make sense to aggregate sales according to the size of products or stores according to their addresses. These attributes, however, can be used to provide more information about other/s attribute/s of the dimension. Finally, the cardinality of the edges of these paths can be either many-to-one or one-to-one.

3.2 Detecting behavioral patterns (BP) on cube classes

In this paper, we will focus on the OLAP operations that will allow us to navigate along cube classes (*roll-up*, *drill-down*, and the extended *combine* and *divide*). These operations are accomplished from one attribute to another, and for that reason, the relationship between these two attributes should be checked in advanced to perform us to execute the corresponding operation. We will identify, therefore, different behavioral patterns based on these relationships between the dimension attributes (types of paths and cardinality patterns) included in the corresponding cube class.

3.2.1 ARRP path between attributes (a_i, a_j)

As this means that there is a classification hierarchy defined on these attributes, users can apply *roll-up* and *drill-down* operations on them to aggregate and de-aggregate data respectively. For example, users would be able to aggregate data by applying the *roll-up* operation from the *store.city* attribute into the *store.area* one (and conversely, de-aggregate data by *drilling-down* from *store.area* into *store.city*).

3.2.2 ACP path between attributes (a_i, a_j)

This means that the attribute a_j provides additional information about another attribute within a classification hierarchy. Independently of the type of the cardinality that joins both attributes, the operation that users can execute is *combine* or *divide* to show more characteristics about a_i or to hide them respectively. For example, users would be able to know the address of each store by applying the *combine* operation from the *store.key_attribute* attribute into the *store.address* one and conversely, to hide this property by *dividing* from the *store.address* attribute into *store.key_attribute* one.

3.2.3 No path between attributes (ai, aj)

In this particular case in which there is not any path directly connecting both attributes, users will apply the operations c*ombine* and d*ivide* as there is not any classification hierarchy defined on them. Other authors such as R. Kimball in [18] argue that *rolling-up* and *drilling-down* is nothing more than adding or deleting row headers in the cube, and therefore, *rolling-up* (and *drilling-down*) should be allowed on attributes even if there is not a classification hierarchy defined on them. For example, users would have to be able to provide the color feature of products by applying the *combine* operation from the *product.family* attribute into the *product.color* one, and conversely, to hide this property by *dividing* from the *product.color* attribute into the *product.family* one.

[1] Note that the KA of any dimension class is always considered as the root of the graph, and therefore, the first element in any classification hierarchy

As it happens in the previous example, users may be interested in grouping data by an attribute within a classification hierarchy (*product.family*) and adding more information with an attribute out of that classification hierarchy, i.e. *product.color*. In this particular case, there is an implicit aggregation and de-aggregation process accomplished by the *combine* and *divide* operations respectively, which was not allowed in the previous BP (3.2.2). This is not a problem, however, as we can take advantage of some characteristics from OO modeling such as overwrite operators and polymorphism that will easily solve this issue of having different behavior for the same operation (method) depending on data static properties.

A summary of the patterns defined previously can be observed in Table 1

Path (a_i, a_j)	**Cardinality Patterns**	**Behavioral Patterns (Operations applied)**
ARRP	many-to-one	*roll-up $_{di}$ (a_i, a_j), drill-down $_{di}$ (a_j, a_i)*
ACP	one-to-one many-to-one	*combine $_{di}$ (a_i, a_j), divide $_{di}$ (a_j, a_i)*
No path	----	*combine $_{di}$ (a_i, a_j), divide $_{di}$ (a_i, a_j)*

Table 1. Cardinality and behavioral patterns

4. OLAP OPERATIONS ON CUBE CLASSES BASED ON PATTERNS

In this section we will define the set of OLAP operations used for navigational process[2] related to the cardinality and behavioral patterns defined in the previous section.

Definition 1 Let *cube class* (CC) be a 6-tuple (DC,FC,A_c,C,E,CE)[3], we define the *roll-up* operation as *aggregation_operator roll-up $_{d1}$ {a_1,a_2}, $_{d2}$ {a_1,a_2} ,..., $_{dn}$ {a_1,a_2}: CC → V* in which the corresponding aggregation operation has been applied to *cube class* objects along the dimension *di* and from the attribute a_1 to a_2 (both along an ARRP path) such as the resulting view *(V)* is a 6-tuple (DC,FC,A_c',C,E,CE) where all components are the same except that A_c'=A_c / $d_i.a_1$ and CE'=CE / *roll-up $_{di}${a_1,a_2}*, i.e. it inherits the rest of characteristics from CC.

Definition 2 Let ARRP be an Attribute Roll-up Relation Path within ag, we define the roll-up domain function as $domroll-up: v_i, a_j \rightarrow V_j$ to obtain the attribute values according to the classification hierarchy. Conversely, we define the drill-down domain function as $domdrill-down: V_j, a_i \rightarrow V_i$

Example for BP 3.2.1. Let us consider the cube class (*CC*) at the top of figure 4. Let us now apply the following *roll-up* operation applying the SUM operator as follows:
SUM roll-up $_{product}${type, family}, $_{store}${city,area} : CC → V. Then, this function will use *domroll-up* functions taking the corresponding values as follows:

domroll-up:Sevilla,area→south
domroll-up:Alicante,area→east
domroll-up:Valencia,area→east
domroll-up:Cordoba,area→south
domroll-up:33m,family→PC486
domroll-up:66m,family→PC486
domroll-up:200m,family→PentiumII
domroll-up:100m,family→PentiumII

Therefore, the implicit *SUM* aggregation operator will therefore be applied to those attributes that have the same attribute value for the *family* and *area* attributes. This roll-up operation is applied to the objects contained in the *cube class* at the top of figure 4 and the resulting view *(V)* is shown at the bottom of figure 4[4].

Sales			**Product.group="Computers"**			
			PC 486		Pentium II	
			33 m	66 m	100 m	200 m
Store. Country = Spain	East	Alicante	10	20	23	52
		Valencia	20	30	25	74
	South	Sevilla	10	15	30	24
		Cordoba	41	21	18	27

Sales		**Product.group="Computers"**	
		PC 486	Pentium II
Store. Country = Spain	East	80	174
	South	87	99

Figure 4. Applying *roll-up* and *drill-down* operations

Definition 3. Let *cube class* (CC) be a 6-tuple (DC,FC,A_c,C,E,CE), we define the *drill-down* operation as *drill-down $_{di}${a_2,a_1}, $_{d2}${a_2,a_1}, ..., $_{dn}${a_2,a_1} : CC → V* in which *domdrill-down* functions have been applied from a_2 to a_1 (both along an ARRP path) along the dimension *di* to find out which attributes and objects (with their values) must be included in the new view *(V)*. The resulting view *(V)* is a 6-tuple (DC,FC,A_c',C,E,CE) where all components are the same except that A_c'=A_c ∪ $d_i.a_1$ and CE'=CE / *drill-down $_{di}${a_2,a_1}*, i.e. it inherits the rest of characteristics from CC.

Example for BP 3.2.1. Let us consider the resulting *cube class* (*CC*) at the bottom of figure 4. Let us now apply the following *drill-down* operation:
drill-down $_{product}${family,type}, $_{store}${area,city}: CC → V. Then, this function will call the *domdrill-down* functions taking the values for the family and area attribute as follows:

domdrill-down:PC486,type→ 33m,66m
domdrill-down:PentiumII,type→100m,200m
domdrill-down:south,city→ Sevilla,Cordoba
domdrill-down:east,city→ Alicante,Valencia

Therefore, values for *type* attribute (33m, 66m, 100m and 200m) and for *city* attribute (Alicante, Valencia, Sevilla, and Cordoba) will be added to the cube and the de-aggregated values will be placed in the corresponding cells. This operation is applied to the objects contained in the *cube class* at the bottom of figure 4 and the resulting view *(V)* after applying the *drill-down* operation is shown at the top of the same figure 4.

Definition 4. Let *cube class* (CC) be a 6-tuple (DC,FC,A,C,E,CE), we define the *combine* operation as *combine $_{d1}${a_1,a_2}, $_{d2}${a_1,a_2}, ..., $_{dn}${a_1,a_2}: CC → V* along with each dimension d_i from a_i to a_j in which objects with the same value for a_2 must be included in the resulting view (V) as well as all different values for a_2 must be shown in V. The resulting view *(V)* is a 6-tuple (DC,FC,A_c',C,E,CE) where all components

[2] The slice/dice operation has been defined in [8].

[3] DC= Dimension Classes, FC= Fact Class, $A_{c=Set\ of\ attributes}$, C=Condition tuple, E (events) ={*new*,*delete*}, CE (Cube Events)=OLAP operations. Further information in [9]

[4] As pointed out in [20] a statistical table might be used for the visualization of the multidimensional view of data

are the same except that A_c'=$A_c \cup d_i.a_2$ and CE'=CE / *combine* $_{di}\{a_1,a_2\}$, i.e. it inherits the rest of characteristics from CC.

Example for BP 3.2.2. Let us consider the cube class (*CC*) at the top of figure 5. Let us now apply the following *combine* operation to show the characteristic of the *address* of every *store* as an example of one-to-one cardinality pattern and the *color* of different *products* as a one of a many-to-one, both of them along an ACP path. *combine* $_{store}$ *{store*[5]*, address},* $_{product}$ *{product, color} : CC* → *V*. The resulting view is shown at the bottom of figure 5[6].

Sales		Product.group="Computers"			
		PC01	PC02	PC03	PC04
Store. Country = Spain	Store1	10	20	23	52
	Store2	20	30	25	74
	Store3	10	15	30	24
	Store4	41	21	18	27

Sales			Product.group="Computers"			
			PC01	PC02	PC03	PC04
			white	black	blue	white
Store. Country = Spain	Store1	1st Ave.	10	20	23	52
	Store2	2nd Ave.	20	30	25	74
	Store3	3rd Ave.	10	15	30	24
	Store4	4th Ave.	41	21	18	27

Figure 5. Applying the *combine* and *divide* operations

Definition 5. Let *cube class* (CC) be a 6-tuple (DC,FC,A_c,C,E,CE), we define the *divide* operation as *aggregation_operator divide* $_{d1}\{a_2,a_1\}$, $_{d2}\{a_2,a_1\}$, ..., $_{dn}\{a_2,a_1\}$ *: CC* → *V* along the dimension d_i and from the attribute a_2 to a_1 such as the resulting view *(V)* is a 6-tuple (DC,FC,A_c',C,E,CE) where all components are the same except that A_c'=A_c / $d_i.a_2$ and CE'=CE / *divide* $d_i\{a_2,a_1\}$, i.e. it inherits the rest of characteristics from CC.

Note that *combine* and *divide* operations will accomplish an implicit de-aggregate and aggregate process respectively if they are applied under the third behavioral pattern, i.e. there is not any path between a_i and a_j. Consequently, these OLAP operations will have different behavior depending on the patterns between a_i and a_j.

Example for BP 3.3.2. Let us consider the cube class (*CC*) at the bottom of figure 5. Let us now apply the following *divide* operation to hide the *address* and *color* characteristics of the *store* and *product* respectively. The corresponding operation to be applied is as follows:
Divide $_{store}$ *{address,store},* $_{product}$ *{color,product} : CC* → *V*. The resulting view is shown at the top of figure 5.

Note that if the relationship between a_i and a_j is one-to-one, no aggregation operation has been applied (store dimension)

Example for BP 3.3.3. Let us consider the cube class (*CC*) at the top of figure 6. Let us now apply the following *combine* operation to show more details about *products*, concretely, the *brands* sold for each *family* of products:
combine $_{product}$ *{family, brand} : CC* → *V*. The resulting view is shown at the bottom of figure 6. It may be observed that in this particular case, an implicit de-aggregation process has been accomplished. Conversely, the *divide* operation can be applied to the cube at the bottom of figure 6 to obtain the cube at the top of figure 6 as follows: SUM *divide* $_{product}$ *{brand, family}:CC* → *V*

Sales			Product.group="Computers"	
			PC 486	Pentium II
Store. Country = Spain	East	Alicante	30	55
		Valencia	50	99
	South	Sevilla	25	54
		Cordoba	62	45

Sales			Product.group="Computers"			
			PC 486		Pentium II	
			IBM	Bull	IBM	HP
Store. Country = Spain	East	Alicante	10	20	23	52
		Valencia	20	30	25	74
	South	Sevilla	10	15	30	24
		Cordoba	41	21	18	27

Figure 6 Applying the *combine* and *divide* operations

5. CONCLUSSIONS

In this paper we have presented the concept of cardinality and behavioral patterns in our GOLD model. A predefined catalog of patterns that examine relationships between dimension attributes over multidimensional views has been presented. We have illustrated how these patterns provide a well-defined strategy to encapsulate data and OLAP operations in MDB from an OO point of view. Furthermore, an extension of the classical set of OLAP operations for navigational processing (*combine* and *divide*) when there is not any classification hierarchy defined on attributes has been presented. Although similar extensions to navigational operations have been presented in [20], the main advantage of the work presented in this paper is that the cardinality and behavioral patterns identified within our OO approach will allow us to automatically generate a prototype of our model from the conceptual level. This issue has not yet been considered by any of the models presented so far. To the best of our knowledge, the only true conceptual approaches presented up to now ([11], [4]) are focused on data static properties, and therefore, neither of them can consider the generation of a prototype automatically.

6. FUTURE WORKS

One main question arises, however, *Why an Object Oriented approach and what is the benefit provided by our OO approach with reference to other proposals (see introduction section for references)?* To sum up, our GOLD model will be supported by a formal specification language, which will allow us to automatically generate a prototype to validate and check the conceptual schema. Other future works that are currently being considered are as follows:

- The use of the specialization operator to construct dimensions when there is a great quantity of different subtypes. This will allow us to take advantage of sharing some attributes (properties) while having others that are particular to every type of different objects

[5] Store represents the key attribute of the dimension class

[6] Other way of showing the result of the operation is hiding the attribute to which the OLAP operation has been applied completely, in this particular case, the key attribute of the store class

- To consider cube classes as collections of objects in which two events, *insert* and *delete*, are defined to add and delete objects automatically after changes are detected in dimension classes

- To increase the group of operations defined on cube classes with the *pivoting* operation (to re-orientate the multidimensional view of data) as well as a formal definition of an interface (presentation model) for these cube classes.

- The use of the polymorphism feature to overwrite operations (methods) that are going to be applied to objects. Concretely, users will be able to use a small group of public methods to execute OLAP operations.

7. REFERENCES

[1] A. Datta and H. Thomas, "A conceptual Model and an algebra for On-Line Analytical Processing in Data Warehouse". In proc. of the workshop on Information Technologies and Systems (WITS'97), Atlanta, 1997

[2] C. Li and X. Wang, "A Data Model for Supporting On-Line Analytical Processing". In proc. of the 5th Int. Conf. on Information and Knowledge Management, (CIKM'96), Rockville (Maryland), USA November 1996, pp. 81-88

[3] C. Sapia, M. Blaschka, G. Höfling, and B. Dinter, "An Overview of Multidimensional Data Models for OLAP". Technical Report, http//www.forwiss.tu-muenchen.de/

[4] C. Sapia, M. Blaschka, G. Höfling, and B. Dinter, "Extending the E/R model for the Multidimensional Paradigm". In proc. of the 1st Int. Workshop On Data Warehouse and Data Mining, (DWDM'98), Singapore, November 1998. Lecture Notes in Computer Science, Springer-Verlag. Vol. 1552, pp. 105-116

[5] J. Gómez, O. Pastor, E. Insfrán, V. Pelechano. "From Object-Oriented Conceptual Modeling to Component-Based Development". In proc. of the 10th Int. Conf. on Database and Expert System Applications (DEXA'99), Florence, August 1999. Lecture Notes in Computer Science, Springer-Verlag

[6] J. Gray, A. Bosworth, A. Layman and H. Pirahesh, "Data Cube: A Relational Aggregation Operator Generalizing Group-by, Cross-Tab and Sub Totals". Data Mining and Knowledge Discovery Journal. Vol. 1, nº 1, 1997

[7] J. Trujillo and M. Palomar, "An Object Oriented Approach to Multidimensional Database Conceptual Modeling (OOMD)". In proc. of the ACM 1st Int. Workshop on Data warehousing and OLAP (DOLAP'98), Washington D.C., USA, November 1998, pp. 16-21

[8] J. Trujillo and M. Palomar, "An Object Oriented Approach to Multidimensional Databases & OLAP operations". Journal of Computer Science and Information Management, 1999. (To appear)

[9] J. Trujillo, "The GOLD model: An Object Oriented multidimensional data model for multidimensional databases". In proc. of the 9th ECOOP Workshop for Ph.D. Students in Object Oriented Systems (PhDOOS'99), Lisbon, Portugal, June, 1999

[10] L. Cabibbo and R. Torlone, "A Logical Approach to Multidimensional Databases". In proc. of the 6th Int. Conf. on Extending Database Technology, (EDBT'98), Valencia, Spain, March.1998. Lecture Notes in Computer Science. Springer-Verlag. Vol. 1377, pp. 183-197

[11] M. Golfarelli and S. Rizzi, "A methodological Framework for Data Warehouse Design". In proc. of the ACM 1st Int. Workshop on Data warehousing and OLAP (DOLAP'98), Washington D.C., The USA, Nov. 1998, pp. 3-9

[12] M. Gyssens and L. Lakshmanan "A Foundation for Multi-Dimensional Databases". In proc. of the 33rd Intl. Conf. On Very Large Database Conference (VLDB'97), Athens, Greece, August, 1997, pp. 106-115

[13] O Pastor, V Pelechano, E Insfrán, J Gómez. "From Object-Oriented Conceptual Modeling to Automated Programming in Java". In proc. of the 17th Int. Conf. on Conceptual Modeling (ER'98), Singapore, November 1998. Lecture Notes in Computer Science, Springer-Verlag. Vol. 1507, pp. 183-196

[14] O. Pastor and I. Ramos. "OASIS 2.1.1: A Class-Definition Language to Model Information Systems Using and Object-Oriented Approach". Ed. Servicio de Publicaciones Univ. Politécnica de Valencia, 3rd ed, 1995

[15] O. Pastor, F. Hayes, and S. Bear. OASIS: An Object Oriented Specification Language". In proc. of the 4th Intl. Conf. on Advanced Information Systems Engineering (CAiSE'92). Manchester, UK, May, 1992. Lecture Notes in Computer Science, Springer-Verlag. Vol. 593, pp. 348-363

[16] P. Vassiliadis. "Modeling Multidimensional Databases, cube and cube operations". In proc. of 10th Int. Conf. on Statistical and Scientific Databases (SSDBM'98), Capri, Italy, June, 1998

[17] R. Agrawal, A. Gupta and S. Sarawagi, "Modeling Multidimensional Databases". In proc of 13th Int. Conf. On Data Engineering, (ICDE'97), Birmingham, U.K., April 1997, pp. 232-243

[18] R. Kimball, "The data warehousing toolkit". John Wiley, 1996

[19] S. Chaudhuri, and U. Dayal, "An Overview of Data Warehousing and OLAP technology". ACM Sigmod Record, vol. 26, no 1, March 1997

[20] W. Lehner "Modelling Large Scale OLAP Scenarios". In proc. of the 6th Int. Conf. On Extending Database Technology, (EDBT'98), Valencia, Spain, March 1998. Lecture Notes in Computer Science, Springer-Verlag. Vol. 1377, pp. 153-167