

Applying MDA to the Development of Data Warehouses

Jose-Norberto Mazón, Juan Trujillo
Dept. of Software and Computing Systems
University of Alicante, Spain
Apto. Correos 99. E-03080
{jnmazon,jtrujillo}@dlsi.ua.es

Manuel Serrano, Mario Piattini
Alarcos Research Group
University of Castilla-La Mancha
Paseo Universidad, 4; 13071 Ciudad Real, Spain
{Manuel.Serrano,Mario.Piattini}@uclm.es

ABSTRACT

Different modeling approaches have been proposed to overcome every design pitfall of the development of the different parts of a data warehouse (DW) system. However, they are all partial solutions which deal with isolated aspects of the DW and do not provide designers with an integrated and standard method for designing the whole DW (ETL processes, data sources, DW repository and so on). On the other hand, the Model Driven Architecture (MDA) is a standard framework for software development that addresses the complete life cycle of designing, deploying, integrating, and managing applications by using models in software development. In this paper, we describe how to align the whole DW development process to MDA. Then, we define MD²A (MultiDimensional Model Driven Architecture), an approach for applying the MDA framework to one of the stages of the DW development: multidimensional (MD) modeling. First, we describe how to build the different MDA artifacts (i.e. models) by using extensions of the Unified Modeling Language (UML). Secondly, transformations between models are clearly and formally established by using the Query/View/Transformation (QVT) approach. Finally, an example is provided to better show how to apply MDA and its transformations to the MD modeling.

Categories and Subject Descriptors

H.2.7 [Database management]: Database Administration – *Data warehouse and repository*; D.2.0 [Software Engineering]: General – *Standards*.

General Terms

Design.

Keywords

Data warehouse, multidimensional modeling, MDA, QVT.

1. INTRODUCTION

Data warehouses (DW) still pay a central role in current decision support systems since they provide crucial business information to improve strategic decision making processes [4]. However, building a DW is still a challenging and complex task because it concerns many organizational units and can often involve many

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

DOLAP '05, November 4–5, 2005, Bremen, Germany.

Copyright 2005 ACM 1-59593-162-7/05/0011...\$5.00.

people with different skills and targets. Moreover, the architecture of a DW is usually depicted as a multi-tier system in which data from one layer is derived from data of the previous layer [6] as we present in figure 1. Such architecture has prompted that different methods and approaches have been presented for designing such different parts of DWs. Nevertheless, these methods do not deal with the whole design of a DW in an integrated fashion. Instead, they provide partial solutions to certain issues, such as ETL (Extraction-Transformation-Load) processes or multidimensional (MD) modeling, therefore, many problems derived from interoperability and integration between layers may arise. To overcome these problems, in a previous work [11] a DW development method based on the UML (Unified Modeling Language) [17] and the UP (Unified Process) [5] has been defined. This method addresses the design of the whole DW from operational data sources to the final implementation by using several UML profiles which have been developed to adapt UML to certain aspects of the DW design, such as MD modeling [9,10,22], modeling of ETL processes [23], modeling data mappings between data sources and targets [12], and physical modeling of the DW [13].

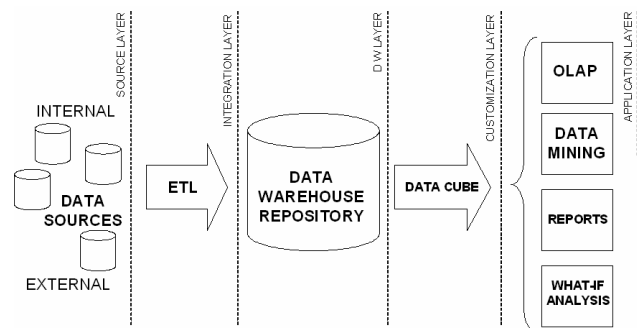


Figure 1. Multi-tier data warehouse.

This method has a clear benefit: using the same standard notation (i.e. UML) to tackle the design of every layer of the DW in an integrated manner in order to provide easier interoperability among every DW subsystem. Unfortunately, this development method may result a little complex for non-expert designers as they can use up to 15 diagrams. Furthermore, the process starts with a MD conceptual model and not enough attention is paid to the requirement analysis phase.

On the other hand, a new standard that addresses the complete life cycle of developing applications by using models in software development is arising: Model Driven Architecture (MDA) [19]. This standard separates the specification of system functionality in a Platform Independent Model (PIM) from the specification of the implementation of that functionality on a specific technology

platform in a Platform Specific Model (PSM). Besides these models, a Computation Independent Model (CIM) is provided by MDA as means of modeling requirements. This MDA framework provides not only the possibility to construct these models in a formal way (i.e. using UML), but also to specify automatic transformations between models in order to obtain the final software product, so less time and effort is needed to develop the whole software system, improving productivity. Furthermore, MDA provides support for system evolution, integration, interoperability, portability, adaptability and reusability [2,8,16].

In this paper, we have aligned the whole process of the DW development with MDA. First, a CIM must be developed in order to acquire user requirements. Secondly, each PIM is designed by using the several UML profiles above-mentioned. These PIMs are conceptual models of each part of the DW itself without considering any aspect of the implementation in a concrete target platform. Normally, PIMs are manually generated from CIM [8,16]. Then, the resulting PIM can be automatically transformed into several PSMs depending of the required target platform. These transformations are formally specified by using a standard called QVT (Query/View/Transformation) [20]. Finally, the necessary code to implement the DW is obtained from PSMs. The main advantage is that once every needed PIM has been developed, we automatically can obtain the PSM and code by using a set of clear and formal transformations, so less time and effort is needed to develop the whole DW. Due to space constraints, we focus on defining MD²A (MultiDimensional Model Driven Architecture), an approach to apply MDA to DW repository development.

The rest of this paper is structured as follows. Section 2 presents the most relevant related work for DW development. An overview of MDA and QVT is presented in section 3. We introduce our framework for DW development with MDA in section 4. In section 5 our approach for applying MDA to MD modeling is outlined. An example is shown in section 6. Finally, in Section 7 we present our conclusions and sketch some future works.

2. RELATED WORK

Various approaches for the conceptual and logical design of DW systems have been proposed in the last few years. In this section, we present a brief discussion about some of the most well-known approaches.

In [7], different case studies of data marts (DM) are presented. The DW design is based on the use of the star schema and its different variations (snowflake and fact constellation). Moreover, the BUS matrix architecture is proposed to build a corporate DW by integrating the design of several DMs. Although we consider this work as a fundamental reference in the MD field, we miss a formal method for the design of DWs. Furthermore, the authors focus on logical model, and no conceptual model is developed.

In [3], the authors propose the Dimensional-Fact Model (DFM), a particular notation for the DW conceptual design. Moreover, they also propose how to derive a DW schema from the data sources described by Entity-Relationship (ER) schemas. From our point of view, this proposal is only oriented to the conceptual and logical design of the DWs repository, because it does not consider important aspects such as the design of ETL processes. Furthermore the authors assume the existence of all the ER

schemas of the data sources, which unfortunately occurs in few occasions. Finally, we consider that the use of a particular notation makes difficult the application of this proposal.

In [1], the authors present the Multidimensional Model, a logical model for OLAP systems, and show how it can be used in the design of MD databases. The authors also propose a general design method, aimed at building a MD schema starting from an operational database described by an ER schema. Although the design steps are described in a logic and coherent way, the DW design is only based on the operational data sources, what we consider insufficient because the final users' requirements are very important in the DW design.

In [24], the building of star schemas (and its different variations) from the conceptual schemas of the operational data sources is proposed. Once again, it is highly supposed that the data sources are defined by means of ER schemas. This approach does not propose a particular graphical notation for the conceptual design of the DWs, instead it uses the ER graphical notation.

Every of the above-described approaches presents the following drawbacks: (i) they do not define a standard notation for conceptual modeling (e.g. UML), and therefore, there is a need for learning a new notation for DWs, (ii) some of the transformations to generate logical and physical models from a conceptual model are not totally automatic in a formal way, (iii) they are based on a specific implementation (e.g. star schema in relational databases), (iv) they do not provide methods to design every stage in DW (ETL, mapping, cubes, data mining...), and (v) they do not provide an approach to model requirements for DWs.

On the other hand, to the best of our knowledge, only one effort has been developed for aligning the design of DWs with the general MDA paradigm, the Model Driven Data Warehouse (MDDW) [21]. This approach is based on the Common Warehouse Metamodel (CWM) [18], which is a platform-independent metamodel definition for interchanging DW specifications between different platforms and tools. Basically, CWM provides a set of metamodels that are comprehensive enough to model an entire DW including data sources, ETL processes, MD modeling, relational implementation of a DW, and so on. These metamodels are intended to be generic, external representations of shared metadata. The proposed MDDW is based on the CWM as the PIM for the DW design. However, CWM metamodels are (i) too generic to represent all peculiarities of MD modeling in a conceptual model (i.e. PIM) and (ii) too complex to be handled by both final users and designers [15], so we deeply believe that it is more reliable to design a PIM by using an enriched conceptual modeling approach easy to be handled (e.g. [9,10]), and then transform this PIM into a CWM-compliant PSM in order to assure the interchange of DW metadata between different platforms and tools.

3. MODEL DRIVEN ARCHITECTURE OVERVIEW

In this section we summarize MDA, QVT and their main characteristics. However, for a more detailed explanation, we refer the reader to [2,8,16,19,20].

Model Driven Architecture (MDA) is an Object Management Group (OMG) standard [19] that addresses the complete life cycle of designing, deploying, integrating, and managing applications by using models in software development. MDA separates the specification of system functionality from the specification of the implementation of that functionality on a specific technology platform. Thus, MDA encourages specifying a Platform Independent Model (PIM) which contains no information specific to the platform or the technology that is used to realize it. Then, this PIM can be transformed into a Platform Specific Model (PSM) in order to include information about the specific technology that is used in the realization of it on a specific platform. Later, each PSM is transformed into code to be executed on that platform and obtain the final software product. On the top of these models, MDA also presents a Computation Independent Model (CIM). This model describes the system within its environment (i.e. business domain) and shows what the system is expected to do without showing details about how it is constructed. Therefore, the requirements for the system are modeled by using a CIM. These requirements are traceable to the PIM and PSM that realize them.

CIM can be modeled by using UML (e.g. use cases diagrams) or other languages more specific to requirement analysis, such as i* [25]. PIM and PSM can be developed by using any specification language, but typically UML is used since it is a standard modeling language for general purpose and, at the same time, it can be extended to define specialized languages for certain domains (i.e. metamodel extensibility or profiles).

In figure 2, we can see how the different models are related to each other. The requirements of the pretended system are addressed in a CIM. Afterwards, this CIM is refined into a PIM (normally this refinement is done by hand [8,16]). Then, this PIM can be automatically transformed into different PSMs. Finally, code is generated from each PSM in order to obtain the final software product.

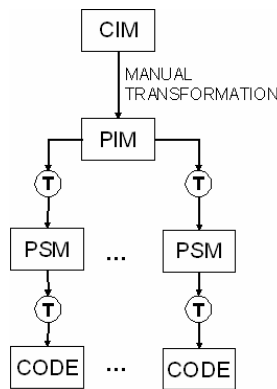


Figure 2. Model Driven Architecture framework.

Nowadays the most crucial issue in MDA is the transformation between a PIM and a PSM [8,19]. In fact, OMG defines MOF 2.0 Query/View/Transformation (QVT), an approach for expressing model transformations [20]. This is a standard under development for defining transformation rules between MOF-compliant models (e.g. UML models). This standard defines a hybrid specification for transformations. On the one hand, there is a declarative part, which provides mechanisms to define transformations as a set of relations that must hold between the model elements of a set of

candidate models (i.e. source and target models). This declarative part can be split into two layers according to the level of abstraction: the relational layer that provides graphical and textual notation for a declarative specification of relations, and the core layer that provides a more formal definition of transformations. On the other hand, QVT also defines an imperative part which defines operational mappings that extend the declarative part with imperative implementations. This part is used when it is difficult to provide a purely declarative specification of a relation.

We wish to point out that QVT is not restricted to PIM-PSM transformations, but also PIM-PIM or PSM-PSM transformations can be developed, provided their metamodels are MOF-compliant.

4. MDA ORIENTED DATA WAREHOUSE DEVELOPMENT FRAMEWORK

As we have previously presented in the introduction, the architecture of a DW is usually depicted as a system with several layers which have different characteristics (see figure 1). On the other hand, MDA presents several viewpoints (i.e. computational independent, platform independent, and platform specific). Following these considerations we present an MDA oriented DW development framework. In this framework, each DW layer represents a part of the whole DW system that must be designed. Each layer is situated within three different kinds of viewpoints according to MDA. Therefore, we consider that the whole development of a DW can be structured into an integrated framework with five layers and three viewpoints. For each layer and viewpoint, different kinds of models are required as we can see in figure 3. The main advantage of this framework is that the DW is automatically generated from defined PIMs. Therefore, the productivity is improved and development time and cost decrease. Other benefits of applying MDA to DW development are the followings:

- **Business perspective:** DW developers shift focus from logical and technical details (i.e. PSM) to CIM and PIM, so they pay more attention to develop conceptual models of each layer of the DW, trying to solve business problems in a better way. Therefore the developed DW fits much better with the needs of the end users. Then, it is clear that MDA helps to improve quality of the developed DW by focusing on business needs.
- **Portability:** the same PIM can be automatically transformed into multiple PSMs for different DW technologies. Therefore, everything which is specified at the PIM level is completely portable to different target platforms.
- **Integration and interoperability:** DWs are heterogeneous systems, thus its development claims for the need of manage metadata in an efficient and effective way. Using MDA transformations, concepts from one platform can be converted into concepts used in another platform in an automatic way.
- **Reusability:** since transformations represent repeatable design decisions, once a transformation is developed, we can use it in every instance of PIMs. Therefore, we can include MD modeling best practices in MDA transformations and reuse them in every project to assure high quality DWs.
- **Adaptability:** if new DW technologies arise for certain parts of the developed DW, then we do not have to change the whole DW. Since the PIM is platform independent, it is still

valid, and DW developers have only to concern about the transformation between the old PIM and the new PSM.

- **Support for system evolution:** every new requirement in DW system is addressed by changing the CIM and the PIM. Then, these changes are automatically reflected in the PSM [2,8,16] and, of course, in the DW system.

In this section we outline our MDA development framework to build the whole DW system. Afterwards, in the next section, we focus on the Data Warehouse layer (i.e. MD PIM, MD PSM and MD code).

4.1 Layers

We distinguish five layers to take into account the definition of a DW development framework (see figure 1):

- Source layer, that defines the data sources of the DW, such as internal data sources (e.g. OLTP systems), or external data sources (e.g. syndicated data).
- Integration layer, that defines the mapping between the data sources and the DW. This layer corresponds with ETL processes and data mapping between the data sources and the DW.
- Data Warehouse layer, that defines the structure of the DW repository. MD modeling is the basis for designing this repository [4,6,7,9].
- Customization layer, that defines special structures (i.e. data cubes) that are used by the end-user applications to access the DW. This is a kind of middleware layer that provide end-user applications with the necessary information from the DW.
- Application layer, that defines the implementation of end-user applications. These applications allow users to analyze data from DW (provided by the customization layer). This analysis is carried out by OLAP, data mining, or reporting tools.

4.2 Viewpoints

Each of the layers above-presented can be analyzed at three different levels according to MDA viewpoints [19]:

- CIM, that defines the requirements for the DW. It is a viewpoint of the DW within its business environment, so it plays an important role in bridging the gap between those that are experts about the domain and its requirements on the one hand (i.e. decision makers), and those that are experts of the design and construction of the DW which satisfies the requirements (i.e. DW developers), on the other.
- PIM, that defines the DW from a conceptual viewpoint. The major aim at this level is to represent the main DW properties without taking into account any specific technology detail.
- PSM, that addresses aspects of the DW design from a certain platform view. For example, a DW repository can be implemented according to different platforms, such as ROLAP (Relational OLAP), MOLAP (Multidimensional OLAP) or HOLAP (Hybrid OLAP).

Besides these DW viewpoints, code is generated from PSM in order to create data structures according to a specific platform. For example, if the chosen platform for implementing the DW repository is a relational one (i.e. ROLAP), then we must create tables, primary keys, foreign keys, and so on.

4.3 Models

Each of the layers previously described is concerned about different issues of the whole DW development which must be represented at different levels of abstraction according to each MDA viewpoint, thus each layer and viewpoint requires different modeling formalisms. In our approach we use UML [17] as the modeling language for constructing PIMs and PSMs, since we can use its extension mechanisms to adapt it to specific domains. In previous works several UML profiles have been developed to adapt UML to certain aspects of DW design: MD modeling [9,10,22], modeling of the ETL processes [23], modeling data mappings between data sources and targets [12], or physical modeling of the DW [13]. Furthermore, some metamodels from CWM [18] can be used to model PSMs. In order to build the CIM, requirements from DW users have to be modeled. A previous work has been developed to start addressing this issue [14], however describing how to build a CIM is out of the scope of this paper.

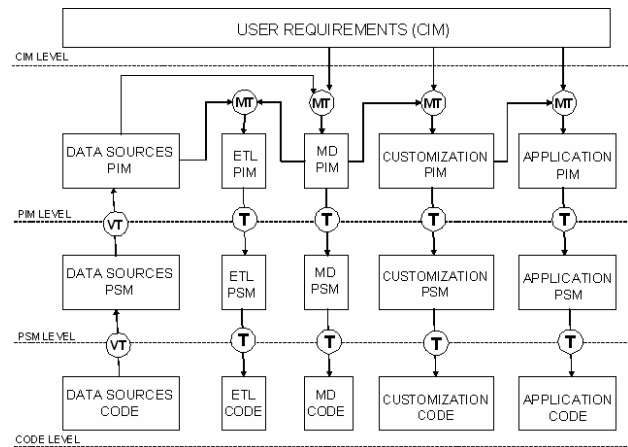


Figure 3. MDA oriented DW development framework.

4.4 Transformations

The main benefit of our framework is that once we have developed each PIM, we can automatically derive the rest of models from them by applying the corresponding transformations until obtaining the final implementation of the DW. These are vertical transformations since a source model is transformed into a target one at a different level of abstraction. However, we can also apply other kind of transformations [16], i.e. horizontal transformations (a source model is transformed into a target model that is at the same level of abstraction) and merging transformations (multiple source models are combined into a single target model) in order to automatically obtain a certain PIM from other PIMs. In figure 3 we can see how transformations are applied to our framework. The central part of this framework is the MD PIM (i.e. MD conceptual model). First a tentative PIM is manually constructed from user requirements (CIM). Then, this first model is enriched by data sources (DATA SOURCES PIM) using a merging transformation¹ to obtain the MD PIM (merging

¹ Please, note that the process of obtaining the MD PIM is out of the scope of this paper; however a previous work has been developed to start addressing this issue [14].

transformation are labeled as MT in figure 3). We wish to point out that a conceptual schema of data sources must be available. However, it is possible that we only have a logical schema or, even worst, only a collection of structured data. Then, it is possible to apply MDA to start from code or a PSM of data sources and obtain the corresponding PIM (transformations labeled as VT in figure 3). Once, the MD PIM has been constructed we can derive the other PIMs by using merging transformations (MT in figure 3) in the following way:

- ETL PIM can be built from DATA SOURCES PIM and from MD PIM.
- CUSTOMIZATION PIM can be constructed from user requirements (CIM) and from MD PIM.
- APPLICATION PIM: from user requirements (CIM) and from CUSTOMIZATION PIM.

Once we have developed every PIM, we can automatically obtain every corresponding PSM and code from them by using vertical transformations (denoted as T in figure 3). Note that the whole DW system can be designed from CIM, MD PIM and DATA SOURCES by only developing every necessary transformation. However, from now on and due to space constraints, we focus on the relational layer of QVT to define formal transformations between instances of PIMs and PSMs in the DW layer (i.e. MD PIM and MD PSM).

5. MD²A: MULTIDIMENSIONAL MODEL DRIVEN ARCHITECTURE

Since the main issue in DW design is the MD modeling and due to space constraints, in this paper we focus on describing MD²A (MultiDimensional Model Driven Architecture), an approach to apply MDA to DW repository development, i.e. Data Warehouse layer within our framework described in section 4. The basis for designing this repository is the MD modeling, thus we apply MDA to MD modeling.

The hottest issue in MDA is the definition of transformations between PIMs and PSMs, so the following subsections explain our MD PIM, MD PSM, and transformations between them (how the MD PIM is constructed is out of the scope of this paper). In figure 4, we show a symbolic diagram that will help to understand how a MD PSM is constructed from a MD PIM:

- On the left hand side of this figure we have represented the MD conceptual schema (i.e. MD PIM).
- From the MD PIM, we develop the logical model (i.e. MD PSM) using a relational approach to build a star schema. This model is represented on the top right side of figure 4.
- Finally, from the MD PSM we can derive the necessary code (i.e. MD CODE) to create data structures for DW in the platform which indicates the MD PSM. In this case, the MD PSM follows a relational approach so the MD code is SQL. This is represented on the bottom right side of figure 4.

PIMs and PSMs are modeled by using the UML. Although it is a modeling language for general purposes, it can be extended for specific domains by using either UML metamodel extensions or UML profiles. Extending the metamodel is more expressive and allows to use transformation standards like QVT, but creating a UML profile provides compatibility for using it in modeling tools

(i.e. Rational Rose). Therefore, in order to combine both advantages, we use our profile for MD modeling as PIM [9,10,22], however we use the corresponding metamodel (see figure 5) to apply QVT transformations. Regarding PSM, we use the Relational CWM metamodel (see figure 6). Transformations between PIMs and PSMs are modeled using the visual and textual notation of the relational layer of QVT (i.e. declarative part).

5.1 PIM definition

A PIM is a view of a system from the platform independent viewpoint [19]. This means that this model describes the system hiding the details necessary for a particular platform. This point of view corresponds with the conceptual level of our framework. The major aim at this level is to represent the main MD properties without taking into account any specific technology detail, so the specification of DW repository is independent from platform in which the DW runs. A PIM is developed following a general purpose modeling language or a language specific to the area in which the system will be used. In this section, we outline a UML profile [9,10] that contains the necessary stereotypes in order to carry out conceptual MD modeling successfully. This profile corresponds with the metamodel of the figure 5. The main features of the considered MD modeling are the *many-to-many* relationships between facts and one specific dimension, degenerated dimensions, multiple classification and alternative path hierarchies, and the non strict and complete hierarchies. In this approach, the structural properties of MD modeling are represented by means of a UML class diagram in which the information is clearly organized into facts and dimensions. These facts and dimensions are represented by fact and dimension classes respectively.

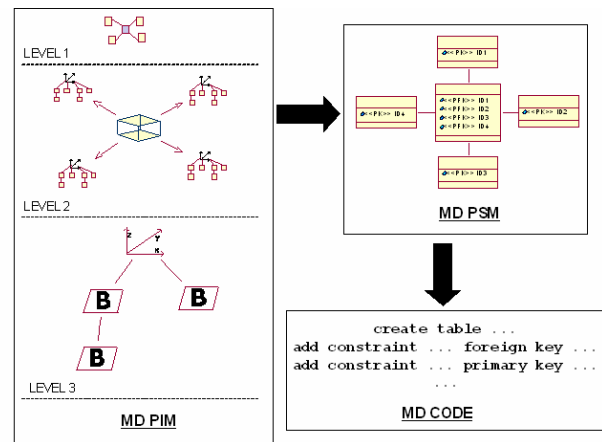


Figure 4. Obtaining MD PSM and code from a MD PIM.

Fact classes are defined as composite classes in shared aggregation relationships of n dimension classes. The minimum cardinality in the role of the dimension classes is 1 to indicate that every fact must always be related to all the dimensions. The *many-to-many* relationships between a fact and a specific dimension are specified by means of the cardinality 1...n in the role of the corresponding dimension class.

A fact is composed of measures or fact attributes. By default, all measures in the fact class are considered to be additive. For non-additive measures, additive rules are defined as constraints and

- Two or more domains: each domain identifies one candidate model (i.e. PIM or PSM metamodels), and a set of elements to match in that model by means of patterns. A domain pattern can be considered a template for objects, their properties and their associations that must be located, modified, or created in a candidate model in order to satisfy the relation.
- A relation domain: it specifies the kind of relation between domains, since it can be marked as *checkonly* (labeled as C) or as *enforced* (labeled as E). A *checkonly* domain is checked to see if there exists a valid match in the model that satisfies the relationship (without transforming any model if the domain patterns does not match); whereas for a domain that is *enforced*, when the domain patterns do not match, model elements may be created, deleted or modified in the target model for the execution in order to satisfy the relationship. Moreover, for each domain the name of its underlying metamodel is specified.
- When clause: it specifies the conditions that must be satisfied to carry out the transformation (i.e. pre-condition).
- Where clause: it specifies the condition that must be satisfied by all model elements participating in the relationship (i.e. post-condition).

Therefore, we have developed every transformation to obtain every PSM from its corresponding PIM by using the diagrammatic (i.e. transformation diagrams) and textual notation of QVT. Due to space constraints we can only describe some of the defined transformations.

5.3.1 Dimension2Table

Dimension2Table transformation is shown in figure 7. On the left hand side we can see the part of the PIM metamodel (labeled as MD) that has to match with the part of the PSM metamodel (labeled as REL) on the right hand side. In this case, a dimension match with a certain collection of objects, links and values, i.e. a table with the name of the dimension. This table must have a column with a certain name (specified in the where clause) and type, which is also the primary key of the table. This relationship between patterns determines the transformation in the following way: it is checked (C arrow) that there is a dimension in the PIM, then if this pattern is found, the transformation enforces (E arrow) that a new table is created with its corresponding column and primary key according to PSM metamodel (see figure 6). The textual notation that corresponds to this transformation can be viewed in figure 8. Once this transformation is done, the transformation *RootBase2Column* must be done (according to the where clause).

5.3.2 RootBase2Column

In figure 9 the *RootBase2Dimension* transformation is shown. Here, a base (with dimension attributes) that is directly linked to a dimension is matching with a table and its corresponding columns (one column per each dimension attribute). The textual notation for this transformation is shown in figure 10. According to the where clause, the transformation *Base2Column* must be performed in order to include attributes for each base in the table previously created by the *Dimension2Table* transformation. The function *MDType2RELType* converts a data type of the source

metamodel (i.e. UML) into a certain data type of the target metamodel (i.e. CWM Relational metamodel).

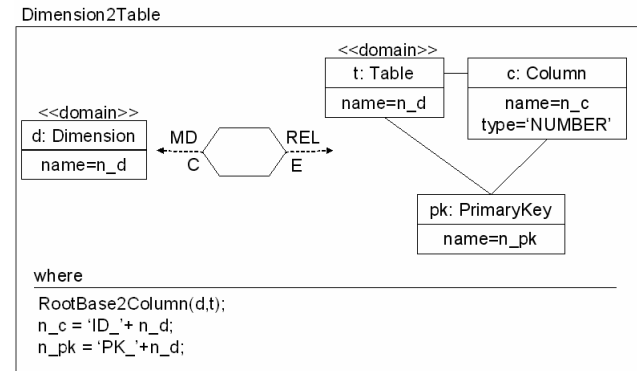


Figure 7. Transforming dimensions into tables.

```

relation Dimension2Table
{
  n_d, n_c, n_pk: String;
  checkonly domain MD d:Dimension {name=n_d};
  enforce domain REL t:Table
  {
    feature=c:Column {name=n_c, type='NUMBER',
      uniqueKey=pk},
    ownedElement=pk:PrimaryKey {name=n_pk,
      feature=c}
  };
  where
  {
    RootBase2Column(d,t);
    n_c = 'ID_'+n_d;
    n_pk = 'PK_'+n_d;
  }
}

```

Figure 8. Textual notation for Dimension2Table.

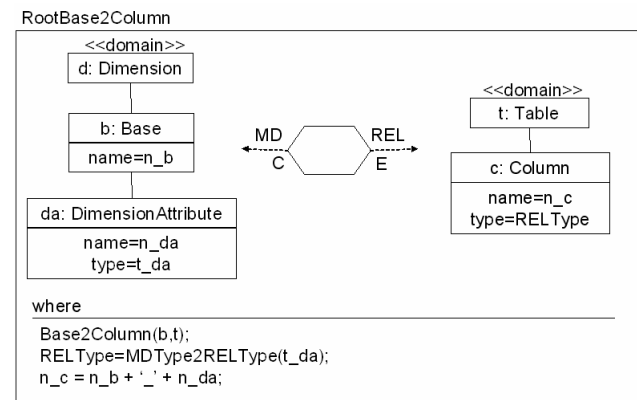


Figure 9. Transforming dimension attributes of a root base into columns.

```

relation RootBase2Column
{
  n_b, n_da, t_da, n_c, RELType: String;
  checkonly domain MD d:Dimension
  {
    base=b:Base
    {
      name=n_b,
      ownedElement=da:DimensionAttribute
      {name=n_da, type=t_da}
    }
  };
  enforce domain REL t:Table
  {
    feature=c:Column{name=n_c, type=RELType}
  };
  where
  {
    Base2Column(b, t);
    RELType = MDType2RELType(t_da);
    n_c = n_b + '_' + n_da;
  }
}

```

Figure 10. Textual notation for RootBase2Column.

5.3.3 Base2Column

The *Base2Column* transformation is shown in figure 11 (textual form in figure 12). Each pair of linked bases matches with a table in order to include all attributes of the base with the role *roll* in the same table (since we are constructing a star schema). This transformation presents a way to recover every base in a hierarchy. We wish to point out that the following transformation depends on the kind of attribute (i.e. DimensionAttribute, OID or Descriptor). Due to space constraints, we only show the *DimensionAttribute2Column* transformation.

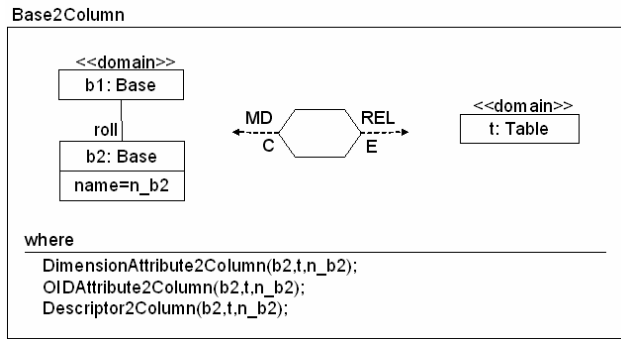


Figure 11. Transforming a base into a table.

```

relation Base2Column
{
  n_b2: String;
  checkonly domain MD b1:Base{roll=b2:Base{}};
  enforce domain REL t:Table{};
  where
  {
    DimensionAttribute2Column(b2,t,n_b2);
    OIDAttribute2Column(b2,t,n_b2);
    Descriptor2Column(b2,t,n_b2);
  }
}

```

Figure 12. Textual notation for Base2Column.

5.3.4 DimensionAttribute2Column

In figure 13 the *DimensionAttribute2Column* transformation is shown. Here, each dimension attribute is matching to a column of a table. Please, note that a prefix is needed in order to complete the name of the column. This prefix is the name of the base (see figure 11).

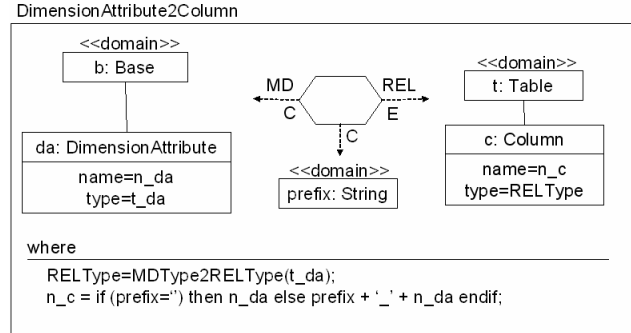


Figure 13. Transforming dimension attributes to columns.

```

relation DimensionAttribute2Column
{
  n_da, t_da, n_c, RELType: String;
  checkonly domain MD b:Base
  {
    ownedElement=da:DimensionAttribute
    {name=n_da, type=t_da}
  };
  enforce domain REL t:Table
  {
    feature=c:Column
    {name=n_c, type=RELType}
  };
  primitive domain prefix: String;
  where
  {
    RELType = MDType2RELType(t_da);
    n_c = if (prefix='') then n_da else prefix
    + '_' + n_da endif;
  }
}

```

Figure 14. Textual notation for DimensionAttribute2Column

6. EXAMPLE

In this section we provide an example to show how to apply the previously presented transformations to a PIM in order to obtain the corresponding PSM. The example can be tracked by figures 15, 16 and 17. The result of the first transformation is given by *Dimension2Table* transformation (see figure 15): the dimension is transformed into elements from CWM Relational metamodel according to this transformation. Once this transformation is executed, the following transformation, i.e. *RootBase2Column* is shown in figure 16. The final transformations (i.e. *Base2Column* and *DimensionAttribute2Column*) can be viewed in figure 17.

As we can see in this example, the corresponding PSM is automatically obtained from the PIM by using our set of formal defined QVT transformations.

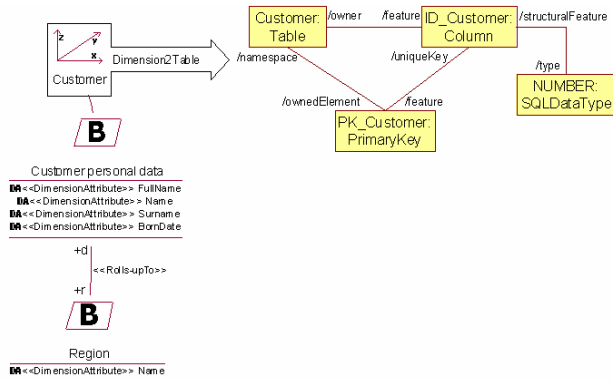


Figure 15. Applying Dimension2Table transformation.

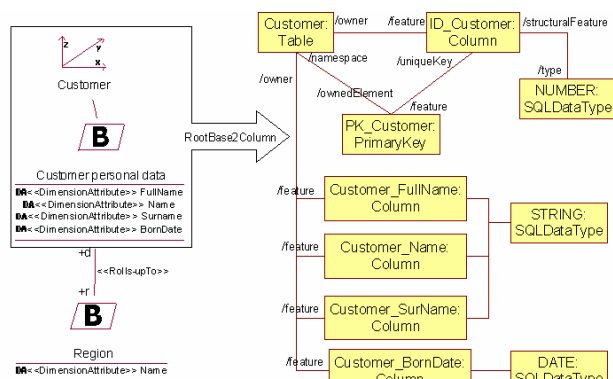


Figure 16. Applying RootBase2Column transformation.

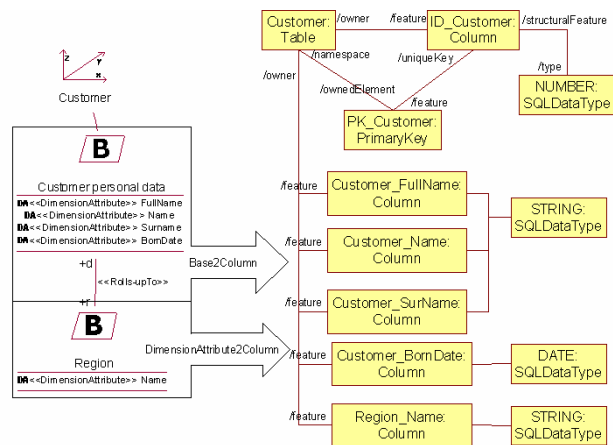


Figure 17. Applying Base2Column and DimensionAttribute2Column transformation.

7. CONCLUSION AND FUTURE WORK

In this paper, we have introduced our MDA oriented framework for DW development. This framework addresses the design of the whole DW system by aligning every layer of the DW with the different MDA viewpoints. Then, for each layer we have three different viewpoints (i.e. CIM, PIM, and PSM). The whole system

is constructed by means of transformations applied to PIM in order to automatically obtain PSMs. From PSM is possible to obtain code in a straightforward way. The development of the DW is reduced to create the PIM of each layer and its corresponding transformations.

In this paper, we have also presented MD²A (MultiDimensional Model Driven Architecture), an approach to apply MDA to DW repository development. We have defined the MD PIM, the MD PSM and the necessary transformations. Our PIM have been modeled by using our MD modeling profile [9,10] as a PIM and the CWM Relational package [18] as a PSM, while the transformations are formally and clearly established by using QVT [20].

We plan to add quality metrics in the transformations by means of marked models [19], i.e. we mark the PIM with metrics in order to generate the most high quality PSM and code. In this way we will be able to adapt transformations in order to generate a pure star schema or a snowflake schema defining the most adequate level of normalization. Therefore, the most suitable DW schema will be obtained in an automatic way.

8. ACKNOWLEDGMENTS

This work has been partially supported by the METASIGN project (TIN2004-00779) from the Spanish Ministry of Education and Science, by the DADASMECA project (GV05/220) from the Valencia Government, and by the MESSENGER (PCC-03-003-1) and DIMENSIONS (PBC-05-012-2) projects from the Regional Science and Technology Ministry of Castilla-La Mancha (Spain).

9. REFERENCES

- [1] Cabibbo L., Torlone R. A Logical Approach to Multidimensional Databases. In: Proc. Of the 6th Intl. Conf. on Extending Database Technology (EDBT'98). Volume 1377 of LNCS, pp. 183-197. Valencia, Spain. 1998.
- [2] Frankel D.S. Model Driven Architecture. Applying MDA to Enterprise Computing. Indianapolis, Indiana. Wiley. 2003.
- [3] Golfarelli M., Maio D., Rizzi S. The Dimensional Fact Model: A Conceptual Model for Data Warehouses. Int. J. Cooperative Inf. Syst. 7(2-3): 215-247. 1998.
- [4] Inmon, W. Building the Data Warehouse (3rd Edition). New York. Wiley & Sons. 2002.
- [5] Jacobson I., Booch G., Rumbaugh J. The Unified Software Development Process. Object Technology Series. Addison-Wesley. 1999.
- [6] Jarke, M., Lenzerini, M., Vassiliou, Y., Vassiliadis, P. Fundamentals of Data Warehouses. Springer. 2000.
- [7] Kimball, R., Ross, M. The Data Warehouse Toolkit, second edition, John Wiley & Sons. 2002.
- [8] Kleppe A., Warmer J., Bast W. MDA Explained. The Practice and Promise of The Model Driven Architecture. Addison Wesley. 2003.
- [9] Luján-Mora S., Trujillo J., Song I-Y. Multidimensional modeling with UML package diagrams. 21st Intl. Conference on Conceptual Modeling (ER2002). Volume 2503 of LNCS. pp. 199-213. Springer-Verlag. 2002.

- [10] Luján-Mora S., Trujillo J., Song I-Y. Extending UML for Multidimensional Modeling. 5th International Conference on the Unified Modeling Language (UML 2002), LNCS 2460, 290-304. 2002.
- [11] Luján-Mora S., Trujillo J. A Data Warehouse Engineering Process. In Proceedings of the 3rd Biennial International Conference on Advances in Information Systems (ADVIS'04). Lecture Notes in Computer Science, Izmir, Turkey, October 2004. Springer-Verlag.
- [12] Luján-Mora S., Vassiliadis P., Trujillo J. Data Mapping Diagrams for Data Warehouse Design with UML. In Proceedings of the 23rd International Conference on Conceptual Modeling (ER'04), Lecture Notes in Computer Science, Shanghai, China, November 2004. Springer-Verlag.
- [13] Luján-Mora S., Trujillo J. Physical Modeling of Data Warehouses using UML. In: Proc. Of the ACM 7th Intl. Workshop on Data Warehousing and OLAP (DOLAP'04), Washington, D.C., USA. 2004.
- [14] Mazón J-N, Trujillo J., Serrano M., Piattini M. Designing data warehouses: from business requirement analysis to multidimensional modeling. 13th IEEE International Requirements Engineering Conference Workshop on Requirements Engineering for Business Needs and IT Alignment (REBNITA). Paris. August, 2005. http://homepage.mac.com/karlalancox/documents/MazonCR_C_000.pdf
- [15] Medina E., Trujillo J. A Standard for Representing Multidimensional Properties: The Common Warehouse Metamodel (CWM). In proceedings of the 6th East-European Conference on Advances in Databases and Information Systems (ADBIS'02), volume 2435 of Lecture Notes in Computer Science, pages 232-247, Bratislava, Slovakia. September, 2002. Springer-Verlag.
- [16] Mellor S., Scott K., Uhl A., Weise D. MDA distilled: principles of Model-Driven Architecture. Addison-Wesley. 2004.
- [17] Object Management Group (OMG). Unified Modeling Language Specification 1.5. <http://www.omg.org/cgi-bin/doc?formal/03-03-01>
- [18] OMG Common Warehouse Metamodel (CWM) Specification 1.0.1. <http://www.omg.org/cgi-bin/doc?formal/03-03-02>
- [19] OMG, Object Management Group: Model Driven Architecture (MDA). 2004. <http://www.omg.org/cgi-bin/doc?formal/03-06-01>
- [20] OMG 2nd Revised Submission: MOF 2.0 Query/Views/Transformations. <http://www.omg.org/cgi-bin/doc?ad/05-03-02>
- [21] Poole J. Model Driven Data Warehouse (MDDW). www.cwmforum.org/POOLEIntegrate2003.pdf
- [22] Trujillo J., Palomar M., Gómez J., Song I-Y. Designing data warehouses with OO conceptual models. IEEE Computer, special issue on Data Warehouses 34, 66-75. 2001.
- [23] Trujillo J., Luján-Mora S. A UML based Approach for Modeling ETL Processes in Data Warehouses. In proceedings of the 22nd International Conference on Conceptual Modeling (ER'03), volume 2813 of Lecture Notes in Computer Science, pages 307-320, Chicago, USA, October 2003. Springer-Verlag.
- [24] Tryfona N. Busborg F., Christiansen J. starER: A Conceptual Model for Data Warehouse Design. In: Proc. Of the ACM 2nd Intl. Workshop on Data Warehousing and OLAP (DOLAP'99), Kansas City, USA. 1999.
- [25] Yu. E. "Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering", Proc. 3rd Int. Symp. on Requirements Engineering, Annapolis, 1997, pp. 226-235.