SOCIETY FOR SCIENCE AND EDUCATION
UNITED KINGDOM
TMLAI TRANSACTIONS ON MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE
VOLUME 5 NO 4, 2017
ISSN 2054-7390

A Model Driven Architecture Approach to Generate Multidimensional Schemas of Data Warehouses

[1]O. Betari, [1]M. Erramdani, [2]K. Arrhioui
[1]MATSI Laboratory, Superior School of Technology, Mohammed First University, Oujda, Morocco
[2]MISC Laboratory, Faculty of Sciences, Ibn Tofail University, Kenitra, Morocco
beta.oualid@gmail.com, m.erramdani@gmail.com, arr.karim@gmail.com

ABSTRACT

Over the past decade, the concept of data warehousing has been widely accepted. The main reason for building data warehouses is to improve the quality of information in order to achieve specific business objectives such as competitive advantage or improved decision-making. However, there is no formal method for deriving a multidimensional schema from heterogeneous databases that is recognized as a standard by the OMG and the professionals of the field. Which is why, in this paper, we present a model-driven approach (MDA) for the design of data warehouses. To apply the MDA approach to the Data warehouse construction process, we describe a multidimensional meta-model and specify a set of transformations from a UML meta-model which is mapped to a multidimensional meta-model. The execution of the transformation, programmed by the Query View Transformation (QVT) language, takes as input an instance of the UML Meta-Model to generate an instance of the Dimensional Meta-Model as output.

Keywords-Data Warehouse; Meta model; Model Driven Architecture; Transformation.

1 Introduction

Decision-making systems are defined by information systems that can help companies in their decision-making process. To support these decisions, it has become necessary to manage and analyze large amounts of data to obtain new and relevant knowledge. In this sense, a new database concept was created: Data Warehouse [1]. However, building a DW is still a challenging and complex task because it consists of several interrelated components with different functions, different design pitfalls, and different technologies [2].

To overcome this problem, we decided to use the Model Driven Architecture (MDA), an approach developed by Object Management Group (OMG), in building the DW. MDA addresses the challenges of constantly evolving highly interlinked systems, providing an architecture that provides portability, interoperability across platforms, platform independence, domain specificity, and productivity [3]. This framework 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 in a Platform Specific Model (PSM). Furthermore, the system requirements are specified in a Computation Independent

DOI: 10.14738/tmlai.54.2397
Publication Date: 10th May, 2017
URL: http://dx.doi.org/10.14738/tmlai.54.2397

Model (CIM). The main benefit of MDA is that less time and effort are needed to develop the whole software system, thus improving productivity [4]-[7].

Considering these aspects, the present paper describes a PIM to PIM transformation using an approach by modeling, we will automatically generate DW schema by using several UML profiles. The resulting PIM represent conceptual model of DW component, without any reference to a concrete target platform or a specific technology. We developed as well the transformation rules by using the formal QVT transformations.

The present paper is structured as follows. Section 2 presents the most relevant related works. In section 3 we outline the MDE principles. Section 4 explains the concepts of Data warehousing. Sections 5 introduces our MDA approach for generating the MD modeling of the DW repository. Finally, section 6 concludes the work and offers further perspectives.

2 Related Works

In this section, we present a brief overview of some approaches, proposed for the development of DW systems.

In [8], the authors propose a Fact-Dimension model. It is a graphical conceptual model for DW. They also show how to derive a DWH schema from the data sources described by the entity-relationship diagram.

A multidimensional conceptual Object-Oriented model for Data Warehousing is proposed in [9]. It was developed as an extension of the UML notation to represent the multi-dimensional data structure. The authors also present OLAP (OnLine Analytical Processing) tools, its structures, integrity constraints and query operation but they don't show how to get the presented conceptual model.

In [1], the authors are interested in secure XML data warehouses. They propose an approach for the model driven development of these DW in which the secure conceptual DW data model, the PIM, is transformed into a secure XML DW, as a PSM, by applying a set of transformation rules.

In [4], the authors present an MDA approach for the development of the DW repository, and then, describe how to build the different MDA models for the DW repository by using an extension of UML and the Common Warehouse Meta model (CWM). They modeled the PIM by using their UML profile for the multidimensional modeling of DW. They derive then the PSM taking into account the deployment platform. From this PSM, the necessary SQL code is derived to create data structures for the DW in a relational platform.

In [10], a well-structured approach is proposed to formalize the development of the DW repository. Authors establish a set of multidimensional normal forms in order to obtain a conceptual model of the DW repository. A relational representation from the conceptual model is informally derived, but no formal transformations are defined to obtain the logical model.

3 Model Driven Engineering

3.1 The OMG approach

In late 2000, OMG, a consortium of over 1,000 companies, first reviewed the document entitled "Model Driven Architecture" and decided to form an architecture team to produce a more formal statement of

the MDA [3]. This approach focuses on developing the highest level of abstraction models and promotes the transformation approach from one model to another.

MDA addresses the challenges of today's highly networked, constantly changing systems, providing an architecture that assures portability, cross-platform Interoperability, platform independence, domain specificity and productivity [3]. The key to the MDA approach is the importance of models in the software development process. In the MDA, the software development process is driven by the modeling activity of the software system.

In the field of software engineering, the OMG with its MDA approach classifies four types of models that it advocates for the construction of software: Computation Independent Model (CIM), Platform Independent Model (PIM), Platform Specific Model (PSM) and Code; defined as follow:

- CIM: The goal is to create a requirements model for the future application. Such a model must represent the application in its environment in order to define the services offered by the application and which other entities with which it interacts.
- PIM: This model represents the system-specific business logic or the design model. It represents the functioning of entities and services. It must be durable and lasting over time. It describes the system, but does not show the details of its use on the platform.

PSM: MDA considers that the code of an application can be easily obtained from these models. The main difference between a code model and an analysis or design model is that the code model is linked to an execution platform.

The reason for the above model organization is to develop models of the systems' business logic independently from the platforms of execution, then to transform these models automatically to models dependent of the platforms. The complexity of the platforms does no longer appear in the business logic models but it' is found in the transformation [11].

The required steps during the model-driven development with the UML approach can basically be divided into the following steps [12], at first building the CIM that acquires user requirements. Then, according to this CIM, a PIM is built. Next is, the transformation of the proposed PIM into one or more PSMs. This type of transition from CIM to PIM and PIM to PSM is called Model To Model (M2M) transformation. The final step is to transform the generated model respecting the PSM into the code of the chosen platform. This transition is called Model To Text (M2T) transformation. Figure 1, shows how the transformations are done.

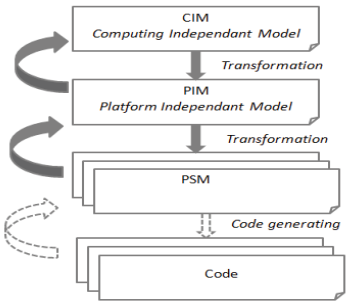

Figure 1. Model Driven Architecture levels

The OMG, in the context of MDA, gives the following definition to the transformation of models "the process of converting a model into another model of the same system". In general, it can be said that a transformation definition consists of a collection of transformation rules, which are unambiguous specifications of how a model can be used to create another model. There are three approaches in MDA to perform these transformations:

- Approach by programming: Consists in using object-oriented programming languages. The idea is to program a transformation of models in the same way that any computer application is programmed.
- Approach by template: Consists in defining the canvas of the desired target models by declaring parameters. These parameters will be substituted by the information contained in the source models. The execution of a transformation involves taking a template and replacing its parameters with the values of a source model.
- Approach by Modeling: Consists in applying the concepts of the engineering of the models to the transformations of the models themselves. The objective is to model the transformations of models and to make the transformation models perennial and productive and to express their independence vis-à-vis the execution platforms.

Using the modeling approach is designed to have a sustainable and productive models' transformation, independently of any execution platform. This is why the OMG has developed a standard for this transformation language which is the MOF 2.0 QVT [14], standing for Query View Transformation.

4 Data Warehouses

In the early 1990s, as a consequence of a competitive world, organizations realized that data analysis needs to be performed to support their decision making processes. Traditional transactional databases do not satisfy the requirements for data analysis because they are designed to support daily operations and ensure concurrent access. However, they do not include historical data and are not optimized to execute complex queries that involve large volumes of data. Therefore, data warehouses were proposed as a solution to overcome these problems and limitations [15]. A data warehouse provides an infrastructure that allows users to get efficient and accurate responses to complex queries. Different systems and tools can be used to access and analyze data stored in data warehouses.

A data warehouse is a subject-oriented, integrated, non-volatile, and time-variant collection of data in support of management's decisions [16]:

- Subject-oriented: The Data Warehouse is organized around major topics of the company such as: customer, sales, products… This organization necessarily affects the design and implementation of data in the Data Warehouse.
- Integrated: Data is integrated from various operational and external systems.
- Non-volatile: Data is accumulated from operational systems for a long period of time.
- Time-variant: In a decision-making system it is important to keep the different values of a given data, allowing comparisons and monitoring of the evolution of values over time, the Data Warehouse keeps track of how its data evolved over time.

Figure 2, describes a typical architecture of a DW system, which consist of several tiers [17]:

- The back-end tier is composed of extraction-transformation-loading (ETL) tools, used to feed data in from operational databases and other data sources, and a data staging area, which is an intermediate database.
- The DW tier is composed of an enterprise data warehouse and/or several data marts, and a metadata repository.
- The OLAP tier is an OLAP server that supports multidimensional data and operations.
- The front-end tier contains client tools such as reporting tools and data-mining tools.

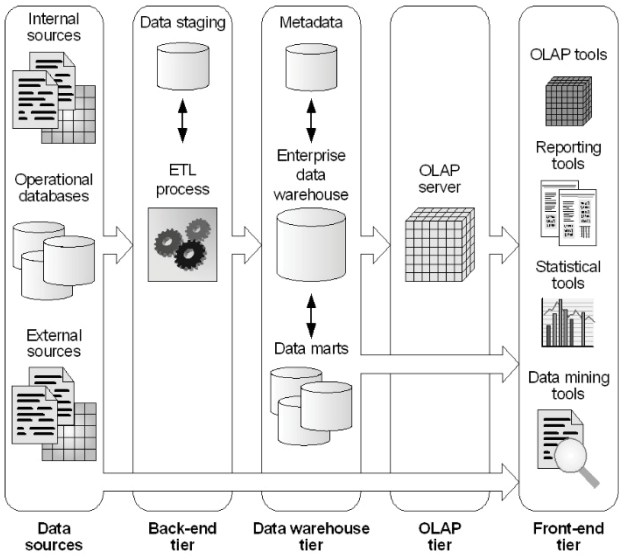

Figure 1. Typical data warehouse architecture

5 MDA for MD Modeling of DW

In this paper we proposed a meta model as a PIM to describe DW. Then, we chose to transform a UML model to PIM, with an approach by modeling using QVT. This type of transformation will allow us to automatically generate the multidimensional data warehouse schema from UML schema. We used UML in our approach, because it takes into account the structural and dynamic properties of the information system at a conceptual level. The class diagram of analysis is obtained from the data dictionary and functional dependencies. Obtaining the conceptual class diagram is based on scenarios of different use cases.

5.1 The Modeling process

5.1.1 PIM source meta model

The UML specification has the Kernel package which provides the core modeling concepts of the UML, our source Meta-model structures a simplified UML model based on a package containing the data types and classes. The classes contain structural features represented by attributes, and behavioral features represented by operations [13]. Fig. 3, presents the source meta model.

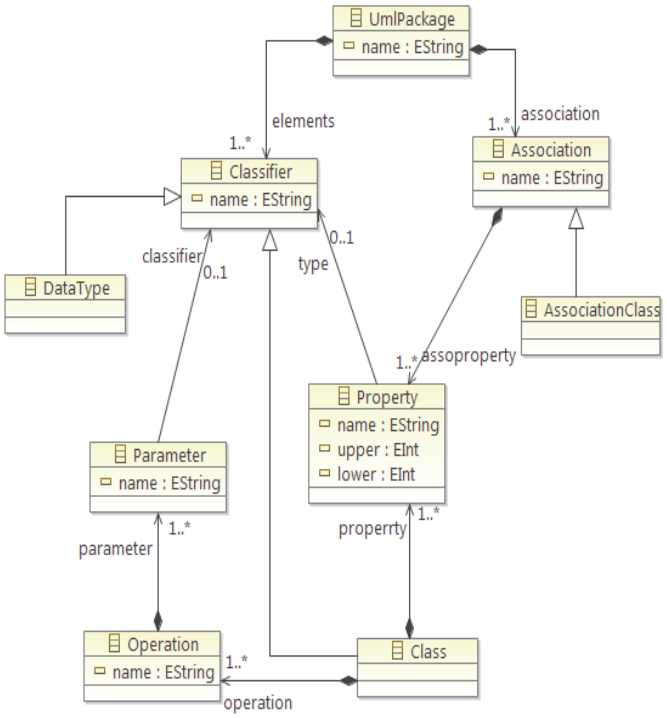

Figure 1. Simplified UML Meta-Model

- UmlPackage: is an abstract element of UML used to group together elements that are semantically related. This meta-class is connected to the meta-class Classifier.
- Classifier: this is an abstract meta-class which describes set of instances having common features. This meta-class represents both the concept of class and the concept of data type.
- Class: represents the concept of UML class.
- DataType: represents the data type of UML.
- Operation: expresses the concept of operations of a UML class.
- Parameter: describes the order, type and direction of arguments that can be given when the operation is invoked. It explains the link between Parameter meta-class and Classifier meta-class.
- Property: expresses the concept of Property of a UML class. These properties are represented by the multiplicities and meta-attributes upper and lower. A UML class is composed of properties, which explains the link between the meta-class Property and meta-class Class. These properties can be primitive type or class type. This explains the link between the meta-class and the Classifier meta-class Property.

5.1.2 PIM target meta model

Based on the approach described in [18], and inspired by this UML profile for the data warehouse, the target meta-model that we have proposed for the transformation, is a simplified meta-model containing Facts and Dimensions. A Fact contains attributes related to dimensions. This meta model is an extension of the work done by the authors in [19]. Fig. 4, shows the Target Meta Model:

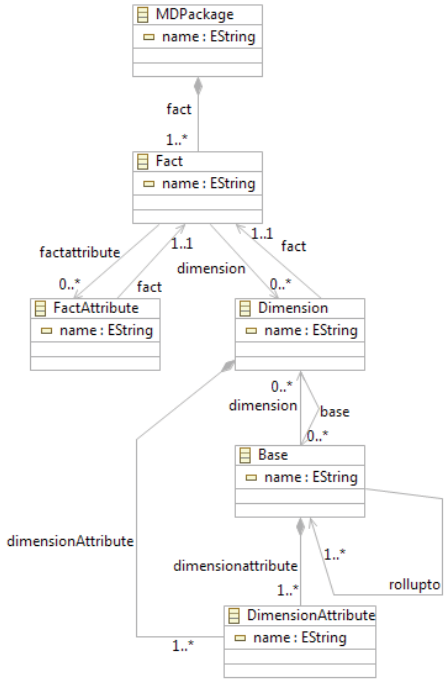

Figure 1. Simplified Meta-Model of multidimensional schema

- MDPackage: expresses the concept of the package, it contains Fact classes.
- Fact: expresses the concept of Fact in a multidimensional schema. Each fact is contained in MDPackage, which explains the link between MDPackage and Fact.
- FactAttribute: expresses the concept of attributes for the element Fact and Fact contains FactAttribute.
- Dimension: expresses the concept of dimension in a multidimensional schema.
- DimensionAttribute: expresses the concept of dimensions attributes. A dimension contains attributes, which explains the link between a Dimension Attribute and Dimension.
- Base: is the basic concept in the multidimensional schema. A base represents the hierarchical level of classification. An association (represented by the ROLLS-UPTO stereotype) between the base class specifies the relationship between two hierarchical levels of classification.

5.2 The transformation process

Once the meta models developed, the next step is to specify the correspondences between the two metamodels and automatically generateDW model from the UML model by using transformation rules written in QVT Operational Mappings. In this section we'll explain some of the programmed transformation rules.

A QVT transformation is in the form of a function with two parameters, the first corresponds to the source model, the second corresponds to the target model. The main function of our transformations is as follow:

```
transformationumlToDwhTransfo(insrc:UML, outdest:DWH);
main() {
      src.objectsOfType(UmlPackage) ->mapumlPackToDWHPack();
}
```

We consider that a simplified UML model consists of a package called UmlPackage. This package will be transformed into a multidimensional package called MDPackage, using the transformation rule defined as follows:

```
mappingUML::UmlPackage::umlPackToDWHPack() : DWH::MDPackage {
      result.name := self.name + "Dwh";
      result.fact += self.association[UML::AssociationClass] -
>mapassociationClassToFact();
}
```

The following code shows the transformation rule of a class association from a UML class diagram to a fact. The fact will carry the name of the association class, its attributes will be the properties of the association class:

```
mappingAssociationClass::associationClassToFact() : Fact {
      result.name := self.name + "Fact";
      factattribute += self.assoproperty ->mappropertyToFactAttribute();
}
```

6 Conclusion

In this paper, we have introduced our MDA approach for the development of DWs. For constructing the system, we applied transformations from a UML PIM in order to automatically obtain an analogous DW PIM. Thanks to the use of MDA and QVT, we can generate from a simplified UML model instance, a simplified Multidimensional model in just two steps: first is the development of the source and target PIMs; and second is the development of the corresponding QVT transformations.

The primary benefit of our approach is that the complex task of designing the whole DW is done in a systematic, well-structured, and standard way by using an MDA approach.

Our future intentions include implementing our MDA approach for the development of DWs into specific platforms by creating the corresponding PSMs. And transforming each generated PSM into code.

REFERENCES

[1] B. Vela, C. Blanco, E.F. Medina, E. Marcos, “A Practical Application of our MDD Approach for Modeling Secure XML Data Warehouses,” in Decision Support Systems (DSS), 54 (4), pp. 899-925, 2012.

[2] R. Kimball, M. Ross, The Data Warehouse Toolkit, 3rded., John Wiley & Sons, Inc. 2013.

[3] Object Management Group (OMG), MDA Guide 2.0. http://www.omg.org/cgi-bin/doc?ormsc/14-06-01

[4] J.N. Mazón, J. Trujillo, "An MDA Approach for the Development of Data Warehouses," in Decision Support Systems (DSS), 45 (1), pp. 41-58, 2008.

[5] D.S. Frankel, Model Driven Architecture: Applying MDA to Enterprise Computing, Wiley, 2003.

[6] A.G. Kleppe, J. Warmer, W. Bast, MDA Explained: The Model Driven Architecture: Practice and Promise, Addison-Wesley, 2003.

[7] S. Mellor, K. Scott, A. Uhl, D. Weise, MDA Distilled: Principles of Model-driven Architecture,Addison-Wesley, 2004.

[8] M. Golfarelli, D. Maio, S. Rizzi, "The Dimensional Fact Model: a Conceptual Model for Data Warehouses," in International Journal of Cooperative Information Systems (IJCIS), 7 (2&3), pp. 215-247, 1998.

[9] A. Abelló, J. Samos, F. Saltor, "YAM2: a Multidimensional Conceptual Model Extending UML," in Information Systems (IS), 31 (6), pp. 541-567, 2006.

[10] J. Lechtenbörger, G. Vossen, "Multidimensional Normal Forms for Data Warehouse Design," in Information Systems (IS), 25 (5), pp. 415-434, 2003.

[11] X. Blanc, MDA en Action: Ingénierie Logicielle Guidée par les Modèles, Eyrolles, 2005.

[12] S. Roubi, M. Erramdani, S. Mbarki, "Generating Graphical User Interfaces Based on Model Driven Engineering," in International Review on Computers and Software (IRECOS), 10 (5), pp. 520-528, 2015.

[13] O.Betari, M. Erramdani, S. Roubi, K. Arrhioui, and S. Mbarki, "Model transformations in the MOF meta-modeling architecture: from UML to codeIgniter PHP framework," Europe and MENA Cooperation Advances in Information and Communication Technologies, vol. 520, pp. 227-234, 2016.

[14] Object Management Group (OMG), MOF 2.0 QVT. http://www.omg.org/spec/MOF/2.0/PDF

[15] A. Vaisman, E. Zimányi, Data Warehouse Systems: Design and Implementation, Springer Berlin Heidelberg, 2014.

[16] W. H. Inmon, Building the Data Warehouse, John Wiley & Sons, 2002.

[17] E. Malinowski, E. Zimányi, Advanced Data Warehouse Design: From Conventional to Spatial and Temporal Applications, Springer-Verlag Berlin Heidelberg, 2008.

[18] S.L. Mora, J. Trujillo, I.Y. Song, "A UML Profile for Multidimensional Modeling in Data Warehouses," in Data & Knowledge Engineering, 59 (3), pp. 725-769, 2006.

[19] I.Arrassen, A.Meziane, R.Sbai, M.Erramdani, "QVT Transformation by Modelling - From UML Model to MD Model," in International Journal of Advanced Computer Science and Applications (IJACSA), 2 (5), pp. 7-14, 2011.

[20] R. . S. Kaplan, D. P. Norton, "The Balanced Scorecard – Measures that Drive Performance," in Harvard Business Review (HBR), 69 (1), pp. 71-79, 1992.