Paper to seminar

Business Intelligence

Model-Driven Architecture (MDA) and Data Warehouse Design

Prof. Dr. Marc Scholl, Prof. Dr. Harald Reiterer, Assistant Svetlana Mansmann

Written by

Eduard Schibrowski

(01/506856)

Table of Contents:

1. Definition and concept of Model Driven Architecture

The concept of Model Driven Architecture was introduced by OMG (Object Management Group) as an approach to system specification and interoperability and is inspired by the use of several formal models.

One of the most accepted definitions of MDA is the one published in the paper "*Applying MDA to the Development of Data Warehouses*", written by Jose-Norberto Mazón, Juan Trujillo, Manuel Serrano and Mario Piattini which refers to MDA as 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.*"[1]

1.1 Key concepts of the MDA Framework

The key concepts of the MDA architecture are in fact the default view points on a system specified by the MDA: computation independent, platform independent and a platform specific.
In MDA, *platform-independent models* (PIMs) are initially expressed in a platform-independent modelling language, such as UML. The platform-independent model is subsequently translated to a platform-specific model (PSM) by mapping the PIM to some implementation language or platform (Java) using formal rules.

1.1.1 PIM (*platform-independent model*)

A PIM has a sufficient degree of independence so as to enable its mapping to one or more platforms. This is commonly achieved by defining a set of services in a way that abstracts out technical details. That means that PIM doesn't contain any information specific to the platform or the technology that is used to realise it. Other models then specify a realization of these services in a platform specific manner.

1.1.2 PSM (*platform-specific model*)

A PSM combines the specifications in the PIM with the details required to stipulate how a system uses a particular type of platform. If the PSM does not include all of the details necessary to produce an implementation of that platform it is considered abstract (meaning that it relies on other explicit or implicit models which do contain the necessary details).

1.1.3 CIM (*computation-independent model*)

A CIM is also often referred to as a business or domain model because it uses a vocabulary that is familiar to the subject matter experts. It presents exactly what the system is expected to do, but hides all information technology related specifications to

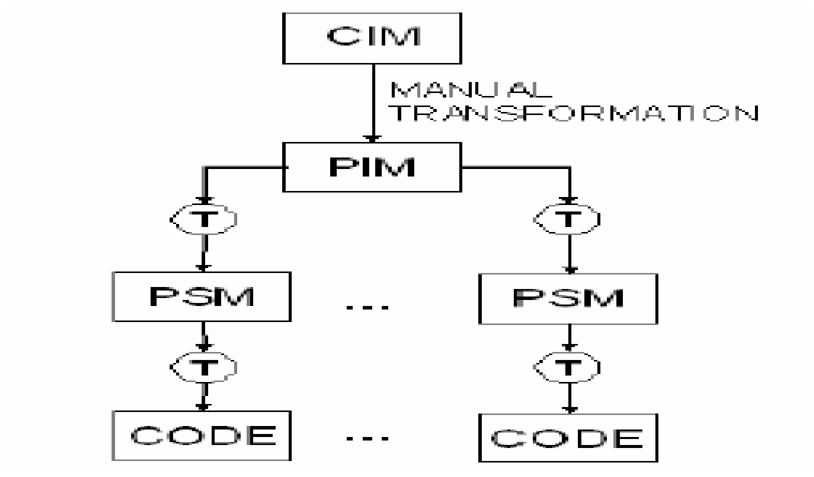
[1] Jose-Norberto Mazon, Juan Trujillo, Manuel Serrano, Mario Piattini: *"Applying MDA to the development of data warehouses"*, in DOLAP '05: Proceedings of the 8th ACM international workshop on Data warehousing and OLAP. 2005, page 58

remain independent of how that system will be (or currently is) implemented. The CIM plays an important role in bridging the gap which typically exists between these domain experts and the information technologists responsible for implementing the system.
In an MDA specification the CIM requirements should be traceable to the PIM and PSM constructs that implement them (and vice-versa).

1.2 Short Framework description

The requirements of the system are addressed in a CIM. Afterwards, this CIM is refined into a PIM (normally this refinement is done by hand. 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.

In the picture below we can see how different models interact:



[2]

As we can observe, the core issue is the transformation between PIM and SPM. While the transformation between CIM and PIM is still done by hand, the PIM – PSM transformation is an automatic process in continuously development. Furthermore, the Object Management Group defines MOF2 (Meta Object Facility) Query/View/Transformation (QVT), an approach for expressing model transformations.
I will discus in more detail about the PIM – PSM automatic transformation, in the context of the development of data warehousing. But before proceeding into the core of the Model Driven Architecture with the focus on CWM (Common Warehouse Metamodel), I would like to highlight at this point the twofold vision of the development of MDA. This consists of a short and a long term vision, delimited by the level of abstraction and automatisation of the (CIM – PIM, PIM - PSM) transformation processes.

[2] Jose-Norberto Mazon, Juan Trujillo, Manuel Serrano, Mario Piattini: *“Applying MDA to the development of data warehouses”*, in DOLAP '05: Proceedings of the 8th ACM international workshop on Data warehousing and OLAP. 2005, page 59

The near term vision (i.e., nearly seamless interoperability, based on formal PIM-PSM translations and shared metadata) is achievable right now. The supporting technologies are largely specified and implementations are currently being built by a number of organizations. ***The long-term vision*** (i.e., the wide spread deployment of AOMs, as an evolution of the MDA), on the other hand, is still being conceptualized.[3]

2. The core of the MDA concept.

A keyword in describing the concept of MDA is ***interoperability***. This is the reason why, it is interesting to see how this problem was solved in the past. For a very long time, interoperability had been based largely on CORBA (Common Object Request Broker Architecture) standards and services. Heterogeneous software systems interoperate at the level of standard component interfaces. CORBA “wraps” program code into a bundle containing information about the capabilities of the code inside and how to call it. The resulting wrapped objects can then be called from other programs (or CORBA objects) across a network.

On the other hand the MDA solves the problem of interoperability by using conceptualized (formal) system models. That’s why, the key to successful integration and interoperability lies in the intelligent use and management of metadata across all applications, platforms, tools, and databases. This can be realised through the use of the MDA standards or technologies: UML, MOF, XMI and CWM. These technologies are used to describe PIMs. In the next section of this chapter I will introduce each of these standards, with one exception: an entire chapter will be dedicated to CWM, as the standard for modelling metadata in a warehousing environment. Given the importance of CWM, the last chapter will be dedicated to the benefits of applying MDA to DW development.

2.1 UML – The Unified Modelling Language

UML is a general-purpose modeling language that includes a standardized graphical notation used to create an abstract model of a system.

UML addresses the modelling of architecture, objects, interactions between objects, data modelling aspects of the application life cycle, as well as the design aspects of component based development including construction and assembly.[4]

[3] J. Poole: *“Model-driven architecture: Vision, standards and emerging technologies”*, in ECOOP 2001: Workshop on Adaptive Object-Models and Metamodeling. 2001.
[4] Joaquin Miller and Jishnu Mukerji „ *Model-driven architecture: Document number ormsc/2001-07-01 Architecture Board ORMSC* “

At this point may be important to notice that UML represents the notational basis for definition of CWM, but CWM also extends a subset of the core UML metamodel with data warehousing concepts. The capacity of visual UML models to automatic translate to other languages facilitates the interchange of CWM models in various platforms and tool independent formats (translation of a CWM relational model into SQL DDL statements).

2.2 MOF - Meta Object Facility

The Meta Object Facility (MOF), an adopted OMG standard, (latest revision MOF 1.4) provides a metadata management framework, and a set of metadata services to enable the development and interoperability of model and metadata driven systems. A number of technologies standardized by OMG, including UML, MOF, CWM, SPEM, XMI, and various UML profiles, use MOF and MOF derived technologies (specifically XMI and more recently JMI which are mappings of MOF to XML and Java respectively) for metadata-driven interchange and metadata manipulation. [5]

In other words, MOF is another OMG standard originated in UML, defining a common, abstract language for the specification of metamodels. It defines the essential elements, syntax, and structure of metamodels that are used to construct object-oriented models of discrete systems. MOF serves as a bridge between the CWM and UML metamodels.

The Meta-Object Facility (MOF) technology provides a model repository that can be used to specify and manipulate models, thus encouraging consistency in manipulating models in all phases of the use of MDA.[6]

2.3 XMI - XML Metadata Interchange

XMI is a standard interchange mechanism used between various tools, repositories and Middleware. XMI can also be used to automatically produce XML DTDs from UML and MOF models. MOF-based metamodels are translated to XML Document Type Definitions (DTDs) and models are translated into XML Documents that are consistent with their corresponding DTDs.

The XMI technology represents the world of modelling (UML), metadata (MOF and XML) and middleware (UML profiles for Java, EJB, IDL, etc.) In essence XMI adds modeling and *Architecture* to the world of XML.

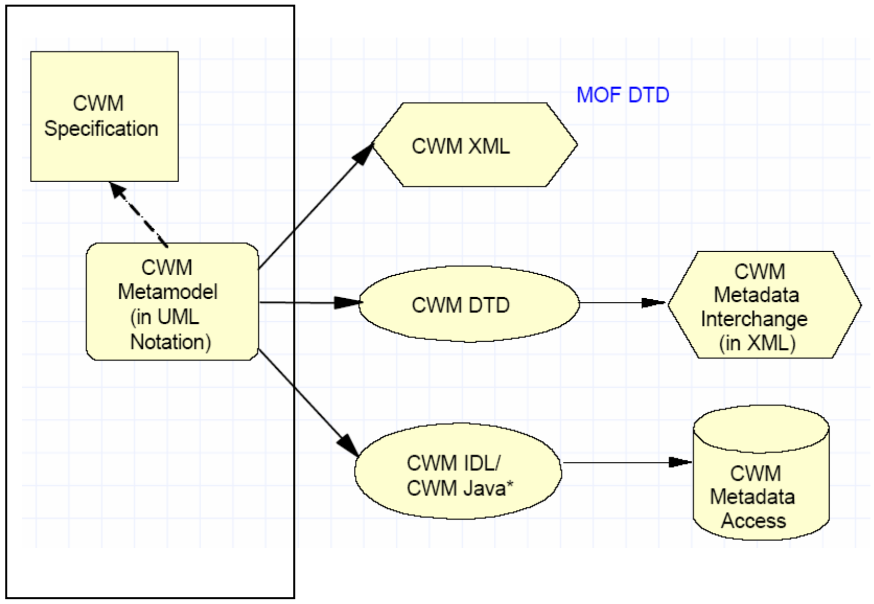
[5] Meta Object Facility (MOF) 2.0 Core Specification, http://www.omg.org/docs/ptc/04-10-15.pdf

[6] Joaquin Miller, Jishnu Mukerji, MDA Guide Version1.0.1, omg/2003-06-01, http://doc.omg.org/formal/02-04-03

3. CWM – Common Warehouse Metamodel

3.1 From the general MDA to the more specific CWM Metamodel

Analogous with the general MDA in the more specific approach of the CWM model architecture we also distinguish between Platform independent (in the left rectangle) and a Platform Dependent perspective (the right side of the picture) comprising technical details for implementation.



[7]

The Common Warehouse Metamodel (CWM) is a specification for modeling metadata for relational, non-relational, multi-dimensional, and most other objects found in a data warehousing environment.

CWM provides objects that describe where the data came from and when and how the data has been created.

Metadata shared between products is formulated in terms of data models that are consistent with one or more CWM metamodels. A product *exports* metadata by formulating a model of its internal metadata structures in a format prescribed by CWM. Similarly, a product *imports* metadata by consuming a CWM-compliant model and mapping it to its internal metadata.

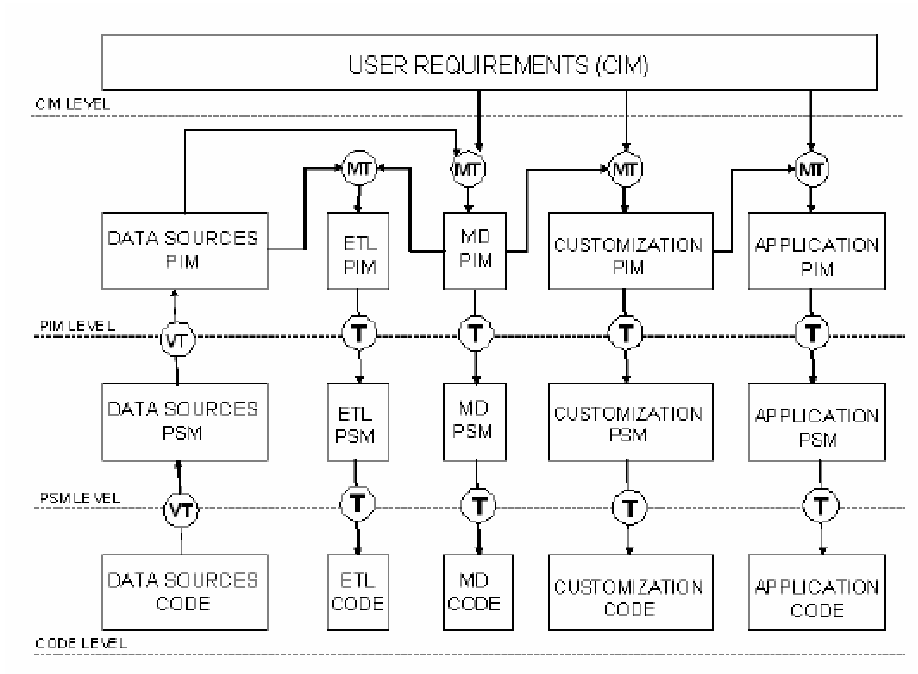
[7] Dr. Daniel T. Chang, IBM Database Technology Institute, CWM: Model Driven Architecture, UML Forum/Tokyo, 3/21/01

3.2 CWM Model Driven Architecture

In the previous sections we could consider the Model Driven Architecture from three view points: computational independent, platform independent and platform specific.
In addition to these viewpoints the DW system architecture introduces several layers of abstraction, and each layer represents a part of the SW that must be designed. Briefly these layers are:

- source layer: which defines internal and external data sources
- integration layer: mapping between data sources and the DW
- DW layer: structure of the data warehouse repository
- customisation layer: middleware used by user applications to access the DW
- application layer: tools that allow analyse data from DW by the end user.

At each level of abstraction, here viewpoints, every of the five layer play an important role in the general consideration of the architecture.



[8]

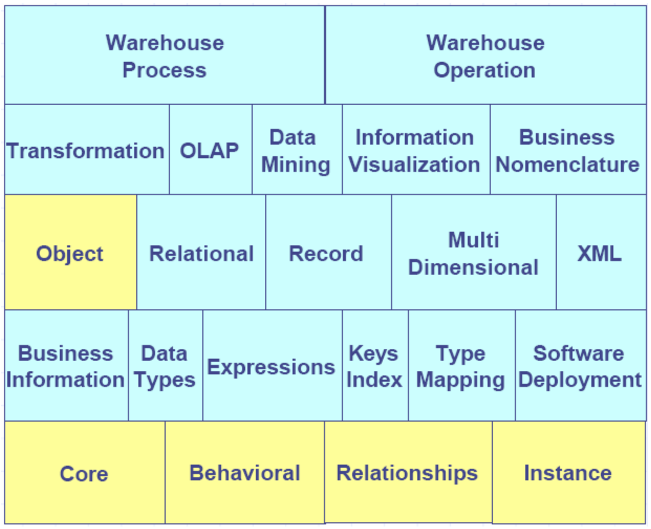
Model Driven Data Warehouse (MDDW) represents the approach developed for aligning the design of DWs with the general MDA paradigm. This approach is based on the Common Warehouse Metamodel (CWM).

CWM 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 modelling, relational implementation of a DW, and so on.).

[8] Jose-Norberto Mazon, Juan Trujillo, Manuel Serrano, Mario Piattini: *"Applying MDA to the development of data warehouses"*, in DOLAP '05: Proceedings of the 8th ACM international workshop on Data warehousing and OLAP. 2005, page 58

The Common Warehouse Metamodel consists of:

- a common metamodel of the data warehousing and business intelligence domain
- a platform-independent metamodel definition
- an XML-based inter-change format for metadata
- a mapping to a platform-independent API specification (CORBA IDL)
- tools that standardize on CWM which can readily share metadata via CWM-compliant XML files



[9]

The left picture provides an overview of the Metamodel with five levels, from top to bottom: Management, Analysis, Resource, Foundation and Object Model

An interesting point at this stage of discussion is the transformation of MD PIM into MD PSM and code. For this purpose we outline a UML profile that contains the necessary stereotypes in order to carry out conceptual MD modelling successfully.

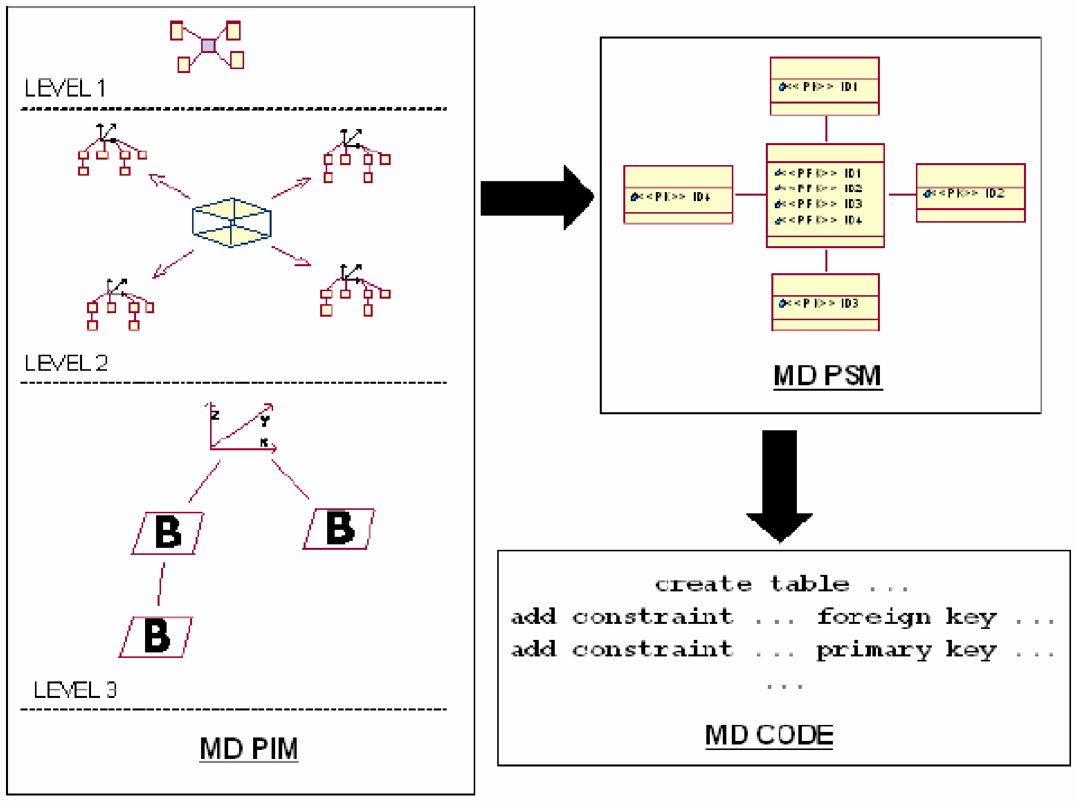
The main features of the considered MD modelling are the *many-to-many* relationships between facts and one specific dimension, degenerated dimensions, multiple classification and alternative path hierarchies, and the non strict and complete hierarchies.[10]

The structural properties of MD modelling are represented by an UML class diagram in which the information is organized into facts and dimensions. These facts and dimensions are represented by fact and dimension classes respectively.

[9]Dr. Daniel T. Chang, IBM Database Technology Institute, CWM: Model Driven Architecture, UML Forum/Tokyo, 3/21/01

[10] idem

[11]



```
create table ...
add constraint ... foreign key ...
add constraint ... primary key ...
...
```

The upper image illustrates the described process of transformation, passing through the three levels of abstraction until the code is generated. I have to specify that there are no new UML design models in the MDA DW architecture (alternatives to class or state diagrams), but the existing ones are enriched with convention specifications for the more specific field of Data Warehousing.

[11] Jose-Norberto Mazon, Juan Trujillo, Manuel Serrano, Mario Piattini: *"Applying MDA to the development of data warehouses"*, in DOLAP '05: Proceedings of the 8th ACM international workshop on Data warehousing and OLAP. 2005, page 59

4. Benefits of the MDA DW Framework[12]

Considering the facts described in section three of this paper we can conclude that the whole development of a data warehouse is structured into an integrated framework with five layers and three viewpoints. There is an impetuous necessity to discuss the advantages of such a framework by highlighting the reasons for his appliance further development.

Obviously the main benefit is that the DW is automatically generated from PIM's resulting in a cost decrease and improvement of productivity.

Other important benefits deserve to be taken into consideration:

- *Business Perspective* – focus on developing conceptual models for each layer, and shifting attention from logical and technical decisions. This improves the way of solving problems and concentrates on the business needs.

- *Portability* – The PIM's can be automatically transformed in PSM's for different DW technologies.

- *Integration and interoperability -* Using MDA transformations, concepts from one platform can be converted into concepts used in another platform in an automatic way.

- Reusability – this is in fact the same concept like the one applied in every object oriented software development – that means that once a transformation is developed, we can use it in every instance of PIM's.

- 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 – the changes in the technology doesn't represent a menace any more. The changes in the requirements are reflected in the change of CIM and PIM. These changes are automatically overtaken in PSM and visible in the DW.

[12] Inspired from: Jose-Norberto Mazon, Juan Trujillo, Manuel Serrano, Mario Piattini: *"Applying MDA to the development of data warehouses"*, in DOLAP '05: Proceedings of the 8th ACM international workshop on Data warehousing and OLAP. 2005

5. References:

1. http://www.cwmforum.org/paperpresent.htm

2. http://www-128.ibm.com/developerworks/rational/library/3100.html

3. http://www.omg.org/

4. http://www.wiley.com/legacy/compbooks/poole/CWM_Guide/patterns.html

5. http://www.omg.org/mda/presentations.htm

6. Meta Object Facility (MOF) 2.0 Core Specification, http://www.omg.org/docs/ptc/04-10-15.pdf

7. Jose-Norberto Mazon, Juan Trujillo, Manuel Serrano, Mario Piattini: *“Applying MDA to the development of data warehouses”*, in DOLAP '05: Proceedings of the 8th ACM international workshop on Data warehousing and OLAP. 2005

8. J. Poole: *“Model-driven architecture: Vision, standards and emerging technologies”*, in ECOOP 2001: Workshop on Adaptive Object-Models and Metamodeling. 2001

9. Joaquin Miller and Jishnu Mukerji „ *Model-driven architecture: Document number ormsc/2001-07-01 Architecture Board ORMSC* “

10. Joaquin Miller, Jishnu Mukerji, MDA Guide Version1.0.1, omg/2003-06-01, http://doc.omg.org/formal/02-04-03

11. John Poole, Hyperion Solutions Corporation: „Model Driven Data Warehousing“ Integrate.2003, Burlingame, CA

12. Dr. Daniel T. Chang, IBM Database Technology Institute, CWM: Model Driven Architecture, UML Forum/Tokyo, 3/21/01

13. Frank Truyen: The Fast Guide to Model Driven Architecture, The Basics of Model Driven Architecture, Cephas Consulting Corp, January 2006.