STUDIA INFORMATICA
Nr 1(10) Systemy i technologie informacyjne 2008

Data warehouse design based on UML language

Andrzej Barczak, Michał Wolski

Summary. Modern management in organizations of the 21st century are efficient and dynamic decisions based on collected and well analyzed data. There are growing needs with regard of data analysis related to multiplanar work in corporations force formation of new data warehouses whose construction becomes more and more complex. In this paper we present one of the design methodologies applying to the design of data warehouses. Transformations were particularly emphasized that enable the application of UML models involving Common Warehouse Metamodel.

Key words: Data warehouses, Rationa Unified Process, modeling IT systems, UML

1 Introduction

Almost worldwide works are underway to optimize and facilitate data warehouse design. Contemporary times require not only effective, but also very quick, reliable solutions that are easy in operation [3]. In addition, continuous changes in the organization and its environment force continuous changes in the structure of warehouses. Additionally, greated and greater needs with regard of data analysis force formation of new data warehouses whose construction becomes more and more complex. It seems that key for the success of data warehouse design and construction projects as well as data warehouse change management is to use a design approach so far used commonly in software engineering. The application of a verified methodology supporting software manufacturing as well as visual modeling techniques together with Common Warehouse Metamodel (CWM) may be very helpful in the process of manufacturing a data warehouse. One of such methodologies commonly used in software engineering is Rational Unified Process (RUP) that is almost wholly connected with Unified Modeling Language (UML) modeling language [4] CWM component.

2 UML language and its extensions

UML language is defined in a four layer architecture (four levels of abstraction) [6]. Different layers define elements, notions and relationships between the notions. Subsequent layers are more general and present as follows:

- meta-metamodel layer (level M3) – most abstract level;
- metamodel layer (level M2);
- model layer (level M1);
- user layer (level M0).

UML language can be extended using stereotypes, one may use extensions, pack extensions by means of profiles as well as record features of model elements.

A stereotype defines type of the modeling element in UML. First, one should define type of the modeling element by means of a stereotype, later one may use the definition. To create a stereotype definition, class representing a stereotype should be presented, with dependence on another class representing type of the modeling element, represented by the stereotype. We use key word "stereotype" to refer to the class representing the new type of the modeling element as well as the relationship [5]. "Metaclass" keyword defines the class representing the type of the modeling element to which the stereotype shall apply. Name of a new type should be included within symbols <<>> or double square brackets before or over the model element name.

"Property" is property of the model element. We use it to define attributes and rules of a particular type of the modeling element. To record properties we use a list of comma separated text strings, in square brackets under the model name or after it. All properties may be presented in a computer or natural language. Text string may be constraint or tag.

Tag is attribute of the model element and its value. We create the tag definition when defining the stereotype. The recording is the following: attribute name, comma, attribute type. Tag defined for the given stereotype applies to all model elements to which the stereotype is used.

Constraint is a rule that may e.g. indicate the need to complete a given activity prior to the beginning of another one. It is presented as a text string that can be written in OCL language (Object Constraint Language). Other ways of recording constraints are: pseudocode as well as different programming languages (among others c++, c#, java) [2].

A profile includes: definitions of tags and constraints, definitions of stereotypes. These elements are valid in a specified domain or for a specified purpose. A profile is presented as a package marked with "profile" key word. Thanks to package structure, the diagram can be used to show profile content.

3 Data warehouses

One of the first ones and a generally acceptable definition of a data warehouse is the one presented by Bill Inmon in 1991. According to it, a data warehouse is a thematic, integrated, variable in time repository of non-volatile data, designed to support decision-making processes [6].

A data warehouse is a separated central database collecting information used to manage the organization. This database is isolated from operational databases, and its structure and the tools used to build it should be optimized for analytic

processing. In a warehouse, data are collected that are obtained from time to time from systems operating on operational data.

Every correctly designed data warehouse should consist of the following elements [3]:

- tools for extraction, transformation and loading of data (Extraction, Transformation, Loading – ETL).
- data storage layer.
- layer serving user inquiries.
- tools offering a user's interface.

ETL is the process of selecting data from the source (source system) then processing as well as loading of these data to a warehouse. An example may be DTS (Data Transformation Services) used in the MS SQL Server environment, with which data from any sources can be integrated [1].

Different kinds of data are stored in data warehouses:

- elementary – copies of up-to-date source data acquired from operational databases and adequately processed,
- materialized aggregates – namely calculated values of calculations, in various perspectives (e.g. sum of sale values in time units and territorial distribution) and at various levels of aggregation (e.g. daily, monthly, annual sums),
- historical – elementary data and/or aggregates concerning the past,
- metadata – dictionary information, describing the structure of the data warehouse and source databases from which data are acquired to the warehouse and the way of calculating aggregated data.

Data warehouse life cycle is normally as follows [2]:

- loading and consolidation – data are periodically loaded from operational databases. During loading data are merged and standardized, this involves i.e. conversion of data types and formats, translation of identifiers, transformation to another data model,
- aggregation – straightaway at the time of loading or soon after, calculation is made of materialized aggregates,
- transfer to historical data – before new elementary data version is loaded, previous data must be saved as historical. They are not however moved to a separate archive, but are usually in the same data warehouse to ensure a possibility to make efficient comparisons and time sections,
- removal – is not an operation typical of a warehouse, is performed rarely or never. Removal is likely to occur when historical data are so old that are no longer utilized or when warehouses are reconstructed and change their function.

Data warehouse properties distinguishing it from other database solutions:

- multi-dimensional conceptual perspective,
- general multidimensionality,
- unrestricted operations across dimensions,
- unrestricted dimensions and levels of aggregation,

- client-server architecture,
- operation of simultaneous access of many users,
- dynamic operation of rare matrices,
- availability,
- intuitive data handling interface,
- flexibility of generated reports,
- consistent reporting mechanism,
- transparency.

3.1 Logical data warehouse model

Logical data warehouse model consists of the following elements: fact, dimension, attribute, measure.

Facts – analyzed variables.

Dimensions – variables permitting grouping of data, are analyzed.

Attribute – basic model description unit, may determine other attributes and be determined itself. Attributes have a set of acceptable values, e.g. attribute hour may have values 11.

Measure – number value assigned to any fact (number of pieces, value of sales).

It can be presented in 2 basic diagrams: star and flake of snow.

Star scheme uses a table of facts surrounded by tables of dimensions. Table of facts includes measurable facts connected by means of keys with tables of dimensions. Tables of dimensions include descriptions of dimensions.

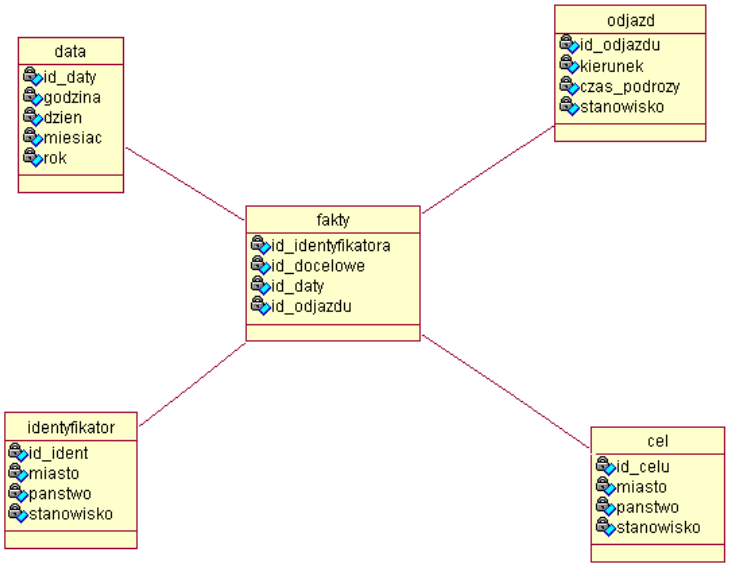

Fig. 1. Example of a logical data warehouse scheme (prepared by the author)

Flake of snow scheme varies from the star scheme in that identifiers of dimensions of the superior and subordinate level are recorded in tables of dimensions. This solution facilitates operations of selection of data related to dimensions as well as permits more precise capture of the attributes hierarchy [6].

Data warehouse architecture can be divided into the following kinds:

- centralized;
- layered;
- federational.

In the centralized architecture, data analyzed in the company are stored in one data warehouse.

The advantages of this solution are:

- simplified access to data as a result of their standardized model;
- construction as well as maintenance of such a warehouse is much simpler than that of a dispersed system
- it is the most appropriate form used in institutions where operations are centralized.

In the layered architecture, the global data warehouse is a real, physical database. Global data warehouse is supplemented by subsequent levels of local, thematic data warehouses.

Federational architecture contains logically homogeneous data that are physically stored in other databases located either in one or few computer systems. Each institution department contains local thematic warehouses. This enables local analyzing at various levels of detail [4].

An important role in construction and operation of a data warehouse is played by metadata. They are stored in a separated base, access to which is granted to all other warehouse components. Metadata base may contain the following elements:

- information about data sought by users,
- data glossaries (definitions of databases as well as relations between elements),
- information about data transformation (performed during transfer),
- stored metadata version numbers,
- names assigned to individual fields in the database,
- information about data flow,
- data use statistics (data profile),
- user rights.

3.2 Extension for data warehouses

To enable easy exchange of metadata between tools, platforms and data warehouse respositories, in heterogenous scattered environments, CWM (Common Warehouse Metamodel) has been developed. CWM is based on the following standards prepared by OMG [8]:

- UML;

- MOF (Meta Object Facility);
- XML.

MOF standard defines an expandable structure for defining metadata models. It also delivers software tools permitting storage and access to meta-data in a repository.

XML permits exchange of data in the form of standard XML files.

Key aspects of such architecture are:

- four-layer meta-model architecture used for metadata manipulation in scattered repositories;
- representation of meta-models and models using UML notation;
- use of standard UML models to describe semantics of object analysis and design models;
- application of MOF to define and manipulate meta-models using CORBA interface (Common Object Request Broker Architecture);
- use of XML for stream metadata replacement.

4 Data warehouse design

XML is used for exchange of metadata (based on CWM meta-model) from the data warehouse. CWM meta-model is described using MOF model (Meta Object Facility), which permits use of XML for:

- transformation of CWM meta-model to CWM DTD file;
- transfer of warehouse metadata instance (if meet the criteria of CWM meta-model) to XML documents based on CWM DTD;
- transformation of CWM meta-model to an XML document.

During use of all of the aforementioned guidelines one can replace warehouse metadata by means of XML.

4.1 Meta Object Facility

MOF is used to define metadata as well as to represent these data as CORBA objects. It supports these kinds of metadata that can be described using object-oriented modeling techniques. Metadata can describe any aspect of the system as well as contain other additional information. There is no limitation as to the degree of detail or precision of description.

Model according to MOF is a collection of metadata, meeting the requirements for structure and compliance (having abstract syntax)

Metadata are a kind of information. If we describe them by metadata, meta-metadata will be created (according to MOF terminology). The model containing meta-metadata is called meta-model [9].

MOF meta-model defines many kinds of metadata, therefore a four-layer architecture has been established (which I have already mentioned in chapter "UML Extensions") as shown in the table below.

Table 1. Metadata architecture according to OMG [8]

Meta level	MOF term	Example
M3	meta-metamodel	MOF model
M2	metamodel, meta-metadata	UML Meta-model, CWM Metamodel
M1	model, metadata	UML models, CWM metadata
M0	object, data	data in warehouse, system models

MOF model may be treated as an abstract language used for description of MOF meta-models. Three main elements of metadata modeling supplied by MOF, similar to UML elements, but more simplified are: class, association and package:

- Class can have attributes and operations on two levels: object and class level. Can multi-inherit after other classes.
- Associations permit binary connection of class instances. Every association has two "ends", which may have different symbols.
- Packages are a collection of associated classes and associations. Packages may be generated by importing other packages or by inheritance after them. Packages can also be nested.

Subsequent important elements of the MOF model are restrictions and data types:

- Restrictions are used for connecting meaning restrictions. Any language can be used to record them.
- Data types enable use of types other than objects as attributes or parameters.

Another important component of MOF is IDL Mapping. This is a set of templates used to replace MOF metamodel into a set of CORBA IDL interfaces. IDL mapping is a very comprehensive issue. Information important here, due to the essence of the work, are:

- class in metamodel is shown as IDL interface, which supports operations, attributes and references defined in the meta-model;
- association is shown as interface, supporting association inquiries;
- a package becomes an interface storing procedures for classes and associations.

MOF is an OMG standard used to represent metamodels. CWM metamodel has been created on the basis of these standards. As a result, you can use XMI for exchange of metadata from the warehouse (saved in CWM)

4.2 Common Warehouse Metamodel

CWM is used to define metamodel of data warehouse architecture. It also contains formal principles of modeling data warehouse instances. CWM metamodel must be saved in MOF (exchange by CORBA or XMI). CWM meta-model contains a package of an object model, based on UML metamodel. In this case, UML Meta-model has been deprived from aspects not connected with data warehouses. CWM meta-model is extension of an object model based on UML. Every meta-class in CWM inherits after meta-class from the object model. UML notation is also used in diagrams representing CWM meta-model.

CWM uses XMI as the exchange mechanism. This means that you may use all XMI possibilities during transfer of warehouse meta-data and CWM meta-model. In addition, CWM does not need additional XMI extensions.

Warehouse metadata may be recorded as XML documents using the principles of creating XMI documents.

CWM metamodel uses packages and hierarchy structure to control complexity, facilitate understanding and re-use. Elements of the model are contained in the following packages [1]:

- Object Model Package
  - Main package (core) – contains classes and associations, forming the core of CWM object model. They are used by all CWM packages together with object model packages;
  - Behavioral packages – contain classes and associations that describe behaviour of CWM objects. They provide the basis for describing calls for predefined behaviors;
  - Linkages package – contains classes and associations describing associations between CWM objects;
  - Instances package – includes classes and associations representing instances of CWM classifiers;
- Basic Package
  - Business information package – it contains classes and associations referring to business information describing elements of the model;
  - Data types package – contains classes and associations describing the structure which should be used when creating necessary data types;
  - Expression packages – include classes and associations showing expression trees;
  - Keys and indexes packages – they include classes as well as associations representing keys and indexes;
  - Software distribution packages – contain classes and associations describing the way of distribution of software in the data warehouse;
  - Type mapping package – describes classes and associations showing data types mapping between different systems;
- Source package
  - Relational package – contains classes and associations having metadata with relational data;
  - Records package – presents classes and associations representing meta-data about records;
  - Multi-dimensional package – has classes and associations describing meta-data with multi-dimensional data;
  - XML package – contains classes and associations showing metadata with XML data;
- Analysis package
  - Transformation package – contains classes and associations representing meta-data with data transformation tools;
  - OLAP package (On Line Analytical Processing) – has classes and associations showing metadata with decision making support software;

- Data mining package – describes classes and associations representing meta-data with software for data mining;
- Information visualisation package – contains classes and associations that represent metadata with information visualization tools;
- Business nomenclature package – classes and associations showing meta-data with business vocabulary;

- Management package
  - Warehouse processes package – contains classes and associations representing metadata of the data warehouse processes;
  - Warehouse operations package – presents classes and associations describing metadata about the results of data warehouse operations

4.3 Multidimensionality

Multi-dimensional data warehouse is represented by multi-dimensional CWM meta-model. It does not give full projection of all aspects of multidimensionality, but gives a sound skeleton which one can build upon with ease according to one's own demands [2].

This multidimensionality is based on the following packages:

- org.omg::CWM::ObjectModel::Core
- org.omg CWM::ObjectModel::::Instance

Classes and associations of multi-dimensional meta-model are presented on (Fig. 2).

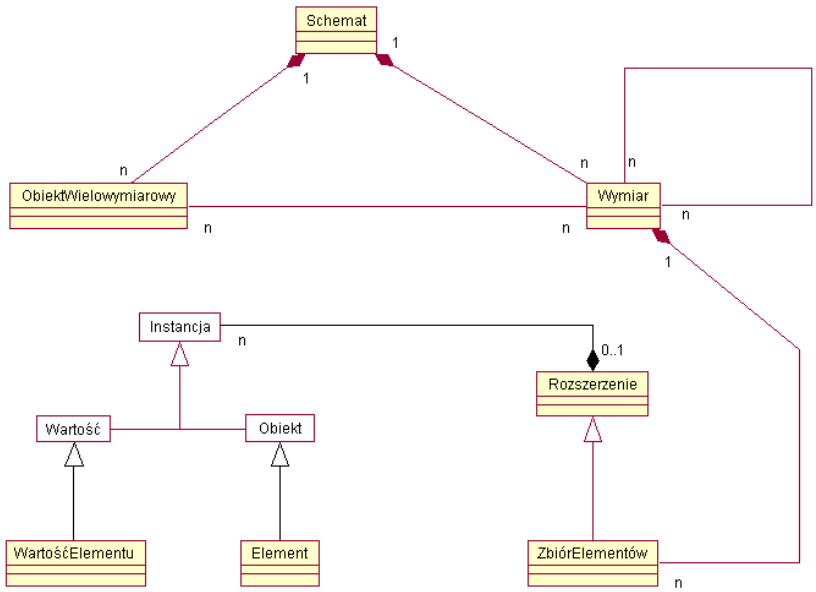

Fig. 2. Multi-dimensional meta-model: Classes and associations, has been prepared on the basis of Common Warehouse Metamodel Specification [10]

The scheme contains all elements comprising a multi-dimensional database. The dimension represents physical dimension of the database. The dimension may have references to other dimensions, what permits establishment of complex multi-dimensional structures. Multi-dimensional object represents the dimension attribute. ElementsSet means a set of elements linked with the dimension instance. The value of ElementsSet means ElementValue.

5 Data warehouse design based on RUP

Rational Unified Process is a complex methodology which can be adapted almost to every project. This goal is stimulated by such tools as Rational Method Composer [9]. The below presented short list of activities according to RUP supporting formation of a data warehouse [2].

Requirements model:

- establishment of requirements type Use Event in RequisitePro;
- making of association of descriptive models repository with graphic models repository indicating a suitable package including (in Rational Rose);
- creation of diagram of cases of use by selecting services from available services earlier prepared in the descriptive model;
- adding actors as well as relations to the model;
- performance of model validation;
- development of main scenarios for all services (as well as selected alternative scenarios) on sequence diagrams;
- sequence diagram transformation into a cooperation diagram;

Analysis model

- classes identification:
  - CRC cards;
  - linguistic analysis;
  - expert methods;
- after the end of identification making of analysis of objects present in scenarios;
- creation of class definitions;
- establishment of relationships between classes;
- making analysis of direct relationships for the classes model by analysis of cooperation diagrams (among others cross technique);
- definition of implementation for cases of use;
- transfer of implementation of cases of use to Logical View;
- transformation of the classes model into the analytical classes model;
  - for each implementation of case of use, creation of:
    - VOPC diagram;
    - limit classes (as many as actors using the service (cases of use));
    - one control class;
    - data classes

  - supplementing definitions of classes and association specifications;
- supplementing scenarios with new objects;

Design model

- development of a data warehouse component (indicating its target engine);
- establishment of working spaces (physical areas of warehouse resource allocation);
- hardening of data classes (only those that will be transformed into tables);
- making of data model:
  - indication of package including hardened classes;
  - creation of warehouse scheme;
  - transformation of classes into the data model;
  - building of data diagram
- assignment of working space to existing tables;
- generation of a data warehouse using advance engineering;
- making of necessary changes on the side of warehouse code;

code synchronization with the database model.

6 Summary

In this work, levels of abstraction of UML language have been presented. An analysis has been made of the mechanisms ensuring easy exchange of metadata between tools, platforms as well as data warehouse repositories. typology has been presented of data warehouse manufacturing processes and processes of its maintenance. A number of transformations have also been indicated that occur across metamodels and meta-metamodels, due to which a data warehouse may be presented on several levels of abstraction. A result of these transformations is the developed process of manufacturing a data warehouse based on the known software manufacturing process – RUP. To sum up, the presented CWM meta-model working in heterogeneous environments along with UML and with participation of RUP methodology is one of the most powerful solutions permitting efficient design and construction of a data warehouse.

References

1. Ambler S.W.: Agile Modeling: Effective Practices for Extreme Programming and the Unified Process, John Wiley & Sons, ISBN#: 0471202827.
2. Ambler S.W.: A RUP-based approach to developing a data warehouse. IBM. DeveloperWorks, http://WWW.ibm.com/developerworks/rational/library/dec06/ambler/ (14.01.2008).
3. Barczak A., Florek J., Sydoruk T.: Bazy danych. Wydawnictwo AP, Siedlce 2006.
4. Barczak A., Florek J., Sydoruk T.: Projektowanie zintegrowanych systemów informatycznych zarządzania. Wydawnictwo AP, Siedlce 2006.
5. Dąbrowski W., Stasiak A., Wolski M.: Modelowanie systemów informatycznych w języku UML 2.1,Wydawnictwo Naukowe PWN, Warsaw 2007, ISBN: 978-83-01-15251-2.

6. Inmon W. H.: Building a Data Warehouse. Wiley Publishing INC, Indianapolis, 2005.
7. Kimball R., Caserta J.: The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleanin, Wiley Publishing INC, Indianapolis, 2004.
8. The Object Management Group (OMG) www.omg.org
9. The Rational Method Composer and RUP, http://WWW-128.ibm.com/ developerworks/rational/products/rup (15.01.2008)
10. Common Warehouse Metamodel (CWM) Specification v 1.1 - formal/2003-03-02.
11. UML Infrastructure Specification - UML Version 2.1.2 - formal/2007-11-02.
12. UML Superstructure Specification - UML Version 2.1.2 - formal/2007-11-04.