A Comprehensive Method for Data Warehouse Design*

Sergio Luján-Mora and Juan Trujillo

Department of Software and Computing Systems
University of Alicante (Spain)
{slujan,jtrujillo}@dlsi.ua.es

Abstract. A data warehouse (DW) is a complex information system primarily used in the decision making process by means of On-Line Analytical Processing (OLAP) applications. Although various methods and approaches have been presented for designing different parts of DWs, such as the conceptual and logical schemas or the Extraction-Transformation-Loading (ETL) processes, no general and standard method exists to date for dealing with the whole design of a DW. In this paper, we fill this gap by presenting a method based on the Unified Modeling Language (UML) that allows the user to tackle all DW design phases and steps, from the operational data sources to the final implementation and including the definition of the ETL processes. The main advantages of our proposal are: the use of a standard modeling notation (UML) in the models accomplished in the different design phases, the integration of different design phases in a single and coherent framework and the use of a grouping mechanism (UML packages) that allows the designer to layer the models according to different levels of detail. Finally, we also provide a set of steps that guide the DW design.

Keywords: data warehouse, multidimensional modeling, design methods, UML

1 Introduction

In the early nineties, Inmon [1] coined the term "data warehouse" (DW): *"A data warehouse is a subject-oriented, integrated, time-variant, nonvolatile collection of data in support of management's decisions"*. A DW is "integrated" because data are gathered into the DW from a variety of sources (legacy systems, relational databases, COBOL files, etc.) and merged into a coherent whole. ETL (Extraction-Transformation-Loading) processes are responsible for the extraction of data from heterogeneous operational data sources, their transformation (conversion, cleaning, normalization, etc.) and their loading into DWs.

* This paper has been supported by a grant from the State Secretary of Education and Universities (Spanish Ministery of Education, Culture and Sport).

On the other hand, multidimensional (MD) modeling is the foundation of DWs, MD databases, and On-Line Analytical Processing (OLAP) applications. These systems provide companies with many years of historical information for the decision making process. Various approaches for the conceptual and logical design of MD systems have been proposed in the last few years, such as [2,3,4,5,6,7,8] to represent main MD structural and dynamic properties. These approaches provide their own graphical notations, which forces designers to learn a new specific model together with its corresponding MD modeling notation. Furthermore, none of these approaches has been widely accepted as a standard conceptual model for MD modeling. Due to space constraints, we refer the reader to [9] for a detailed comparison and discussion about most of these models.

In the light of this situation and due to the extensive number of different models used in the different design phases of DWs, we believe that it is absolutely essential to develop a standard method[1] that comprises all DW design phases and steps. Nowadays, some attempts to provide a DW method have been carried out (see discussion in Section 2). However, from our point of view, none of them proposes a model that can be used during all the design of a DW.

In this paper, we present a DW design method to make the developing process of DW more efficient. Our proposal is an object oriented (OO) method based on the Unified Modeling Language (UML) [10] that allows the user[2] to tackle all DW design phases and steps, from the operational data sources to the final implementation and including the definition of the ETL processes and the final users' requirements. Our objective is to combine, in a single model or related set of models, the whole DW analysis and design from the data sources to the final implementation. Unlike other methods and proposals for MD and DW design, our approach is independent of any specific implementation (relational, MD, OO, etc.). Finally, our method is based on a well-known standard modeling language (UML), thereby designers can reduce the effort needed in learning a new specific notation or language for DW design.

The rest of the paper is structured as follows. Section 2 discusses some of the most relevant proposals about the design of DWs. Section 3 presents an overview of our method and introduces the different models that are part of the method. Section 4 proposes a set of steps that help the user to apply our method. Finally, Section 5 presents the main conclusions and future works.

2 Related work

In this section, we present a brief discussion about some of the most well-known DW design methods. Some other methods that we cannot discuss due to the lack of space are [11,12,13].

[1] We use the term "method" instead of "methodology" because a methodology is the study of methods.

[2] We distinguish between final users (end users or business information consumers) and DW developers (DW designers, administrators, programmers, or, in general, information systems professionals).

In [14], different case studies of data marts (DM) are presented. The MD modeling is based in 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 from integrating the design of several DMs. Although we consider this work as a fundamental reference in the MD field (R. Kimball provides a very sound discussion of star schema design), we miss a formal method for the design of DWs. Furthermore, the conceptual and logical models coincide in this proposal, and the concepts on the BUS matrix architecture are a compilation of the personal experiences of the authors and the problems they have faced building enterprise DWs from DMs.

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 DWs, because it does not consider important aspects such as the design of ETL processes. Furthermore, the authors assume a relational implementation of the DWs and 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 [2], the authors present the *Multidimensional Model* (MD), 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 an 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 [15], 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 differs from the above-presented ones in that it does not propose a particular graphical notation for the conceptual design of the DWs, instead it uses the ER graphical notation.

Most recently, in [16] another method for the DW design is proposed. This method is based on a MD model called IDEA and it proposes a set of steps to address the conceptual, logical, and physical design of a DW. One of the most important advantages, with respect to the previous proposals, is that the conceptual schema of the DW is built taking into consideration both the operational data sources and the final users' requirements. Nevertheless, this method only considers the data modeling and does not address other relevant aspects, such as the ETL processes.

In [17], different DW development methods are analysed and a new method is proposed. This method stands out because it integrates the management of metadata. However, it lacks a model that can be used to reflect and document the DW design.

Regarding the UML, some proposals to extend the UML for database design have been presented [18,19,20], since the UML does not explicitly include a data model. However, these proposals do not reflect the peculiarities of MD modeling. There have been some approaches that uses UML for the MD modeling up to date (again, and due to space constrains, we refer the reader to [9] for a comprehensive comparison). However, none of these approaches propose a comprehensive method to cover all main DW design phases.

Therefore, and based on the previous considerations, we believe that currently there is not a standard formal method that comprises the main steps of a DW design.

3 An overview of the method

In this section we present an overview of our OO method that allows the user to tackle all DW design phases and steps, from the operational data sources to the final implementation and including the definition of the ETL processes and the final users' requirements. We have adopted the OO paradigm because it is semantically richer that others [21] and it offers numerous advantages. The design of a DW is a joint effort of DW developers (technical users) and final users (users who are only interested in the business content). Therefore, a powerful (but also easy to understand for both kinds of users) method with the corresponding models is needed and we believe the OO paradigm is the best approach for the DW design.

3.1 Design of a data warehouse

The architecture of a DW is usually depicted as various layers of data in which data from one layer is derived from data of the previous layer [22]. Following this consideration, we consider that the development of a DW can be structured into an integrated model with four different schemas as seen in Fig. 1:

- `Operational Data Schema` (`ODS`): it defines the structure of the operational and external data sources.
- `Data Warehouse Conceptual Schema` (`DWCS`): it defines the conceptual schema (facts, dimensiones, hierarchies, etc.) of the DW.
- `Data Warehouse Storage Schema` (`DWSS`): it defines the physical storage of the DW depending on the target platform (relational, MD, OO, etc.).
- `Business Model` (`BM`): it defines the different ways or views of accessing the DW from the final users' point of view. It is composed of different subsets of the `DWCS`.

From our point of view, two schema mappings are also needed in order to obtain a global and integrated DW design approach that covers the necessary schemas:

- `ETL Process`: it defines the mapping between the `ODS` and the `DWCS`.

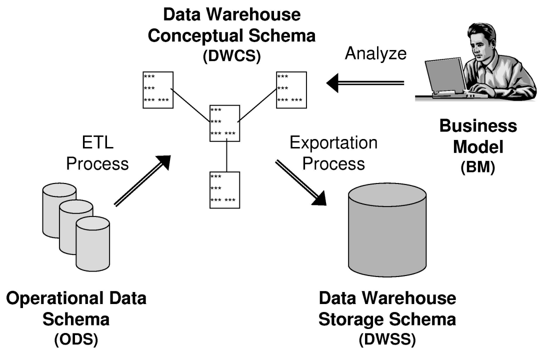


Fig. 1. Different schemas and mappings for designing data warehouses

– `Exportation Process`: it defines the mapping between the `DWCS` and the `DWSS`.

Our method accomplishes each one of the above-commented schemas and mappings in an integrated manner. We use a modeling notation based on the UML [10]. Each one of the schemas is represented by a stereotyped package. Therefore, we provide six stereotypes: «ODS», «DWCS», «DWSS», «BM», «ETL» and «Exportation». Thanks to the use of the UML packages, the design of huge and complex DWs is simplified because the DW designer can achieve the design from different levels of detail.

The different schemas (UML packages) are related to each other by means of UML dependencies. A dependency in the UML is represented as a dashed line with an arrowhead. The following dependencies are allowed in our method:

– «ETL» $\rightarrow$ {«ODS», «DWCS»}.
– «Exportation» $\rightarrow$ {«DWSS», «DWCS»}.
– «BM» $\rightarrow$ {«DWCS»}.

Bidirectional dependencies are not allowed at this level of the model as we prevent a tight coupling between packages. The designer can independently define `ODS`, `DWCS`, and `DWSS` if desired (for example, our proposal can be used to only document the conceptual model of the DW by means of the `DWCS`).

In the following sections, we present with further detail each one of the schemas and mappings we use in our method.

3.2 Operational data schema (ODS)

The `ODS` reflects the structure of the operational data sources (e.g., transaction processing systems or On-Line Transaction Processing -OLTP- systems), which record details of business transactions. We also use the `ODS` to define external sources (e.g. census data, economic data) that DWs often incorporate to support analysis.

Nowadays, there does not exist an accepted UML extension for modeling different types of data sources. Therefore, we have to use different UML extensions to model the `ODS` according to the source. For example, if the data source is a relational database, we use *Rational's UML Profile for Database Design* [20], a UML extension for the design of relational databases, but if the data source is an XML document, we use [23], a UML extension for the modeling of XML documents by means of DTDs and XML Schemas.

On the other hand, the definition of the `ODS` can require different activities, such as reverse-engineering of several data sources, the discovery of relationships in the data sources, etc., because data models are not current, offer insufficient detail, or more likely, do not exist at all; however, these issues are beyond the scope of this paper.

3.3 Data warehouse conceptual schema (DWCS)

In previous works [24,25], we have proposed a UML profile for the conceptual design of DWs following the multidimensional (MD) paradigm. The most important feature of the MD paradigm is dividing data into facts (composed of measures) and dimensions; to provide data on a suitable level of granularity, hierarchies are defined on the dimensions. Moreover, our approach easily and elegantly considers main MD properties at the conceptual level, such as the many-to-many relationships between facts and dimensions, degenerate dimensions, multiple and alternative path classification hierarchies, non-strict and complete hierarchies, etc. Our UML profile is formally defined and uses the Object Constraint Language (OCL) [10] for expressing well-formedness rules of the new defined elements, thereby avoiding an arbitrary use of the profile.

Moreover, our UML profile also includes the use of the UML packages; in this way, when modeling complex and large DW systems, we are not restricted to use flat UML class diagrams and, therefore, cluttered diagrams are avoided. We have divided the design process into three levels (Fig. 2):

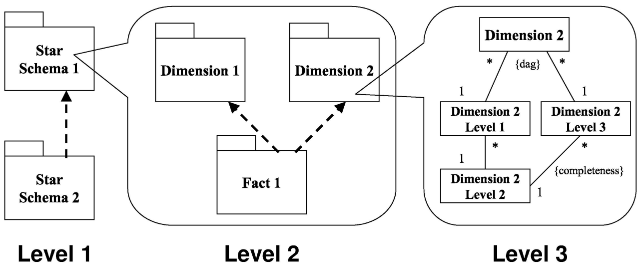


Fig. 2. The three levels of a MD model represented by means of UML packages

Level 1 : Model definition. A package represents a star schema of a conceptual MD model. A dependency between two packages at this level indicates that the star schemas share at least one dimension.

Level 2 : Star schema definition. A package represents a fact or a dimension of a star schema. A dependency between two dimension packages at this level indicates that the packages share at least one level of a dimension hierarchy.

Level 3 : Dimension/fact definition. A package from the second level is exploded into a set of classes that represent the hierarchy levels in a dimension package, or the whole star schema in the case of the fact package.

Table 1 shows the main stereotypes of our UML profile for the conceptual design of DWs.

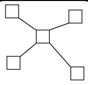
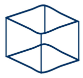
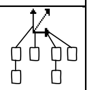
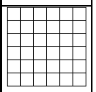
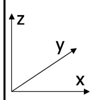
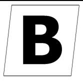
MD concept (Stereotype)	**Description**	**Icon**
`StarPackage`	Packages of this stereotype represent MD star schemas, consisting of facts and dimensions. This stereotype is used at level 1	
`FactPackage`	Packages of this stereotype represent MD facts, consisting of measures and relating facts to dimensions. This stereotype is used at level 2	
`DimensionPackage`	Packages of this stereotype represent MD dimensions, consisting of hierarchy levels. This stereotype is used at level 2	
`Fact`	Classes of this stereotype represent facts in a MD model, consisting of measures (the transactions or values being analyzed). This stereotype is used at level 3	
`Dimension`	Classes of this stereotype represent dimensions in a MD model, consisting of dimension attributes and hierarchy levels (descriptive information about the transactions or values being analyzed). This stereotype is used at level 3	
`Base`	Classes of this stereotype represent dimension hierarchy levels in a MD model. This stereotype is used at level 3	

Table 1. MD mechanisms and the representation in the UML

3.4 Data warehouse storage schema (DWSS)

Depending on the implementation of the DW (relational, MD, OO, etc.), the `DWSS` is modeled by applying different UML extensions (in the same way that the `ODS`) that allow the designer to accomplish the logical data model of the DW.

Moreover, our proposal offers two possibilities: manual and automatic definition. In the former, the DW designer needs to define the `DWSS` and the mapping

(`Exportation Process`) between the `DWCS` and the `DWSS`; in the latter, a specific transformation algorithm for each type of data storage generates the corresponding `DWSS`.

In the case of a manual definition of the `DWSS`, the DW designer can also specify some physical representations, such as summaries (also called aggregates or materialized views) and derived data.

3.5 Business model (BM)

This schema is used to adapt the DW according to the particular needs of final users. In this way, we can create different "views" from the DW schema. This is very useful because DW developers may want to provide DW structures that are easier for final users to understand and write queries against, or because there are security concerns that force to limit the access to the DW.

We use the UML importing mechanism for creating the `BM` because it allows us to define different submodels of the `DWCS`. The different models of the `BM` can be implemented as DMs, which tailors the DW to the needs of a specific group of users. A DM may be real (stored as actual tables populated from the DW) or virtual (defined as views on the DW).

Finally, the final users' initial queries (e.g., a listing of the top and bottom selling products) are also defined in the `BM`. We use the cube classes, a graphical notation we have previously presented in [7] that allows us to easily and visually represent final users' queries.

3.6 ETL process

In [26], we have proposed a UML extension that allows the designer to accomplish the conceptual modeling of ETL processes. We provide the necessary mechanisms for an easy and quick specification of the common operations defined in ETL processes such as, the integration of different data sources, the transformation between source and target attributes, the generation of surrogate keys and so on. Then, from the `ODS`, we design the ETL processes that will be responsible for the gathering, transforming and uploading data into the DW.

In our proposal, we have defined a reduced and yet highly powerful set of ETL mechanisms. We have decided to reduce the number of mechanisms in order to reduce the complexity of our proposal. We have summarized these mechanisms in Table 2. We consider that these mechanisms process data in the form of records composed of attributes, and therefore, we provide the `Wrapper` mechanism to transform any source into a record based source.

The ETL mechanisms are related to each other by means of UML dependencies. Moreover, a UML note can be attached to every ETL mechanism to (i) explain the functioning of the mechanism and, (ii) define the mappings between source and target attributes of the ETL mechanisms. These mappings conform to the following syntax: target_attribute = source_attribute. To avoid overloading the diagram with long notes, when source and target attributes'

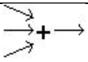
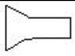
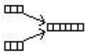
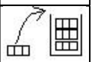
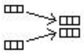
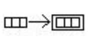
ETL Mechanism (Stereotype)	**Description**	**Icon**
`Aggregation`	Aggregates data (SUM, AVG, MAX/MIN, COUNT, etc.) based on some criteria	
`Conversion`	Changes data type and format or derives new data (derived attributes) from existing data	A → B
`Filter`	Filters and discards non-desired data, and verifies data quality by means of constraints	
`Incorrect`	Reroutes incorrect and discarded data to a different target for the later check; it can only be used with `Filter`, `Loader` and `Wrapper`	
`Join`	Joins two data sources related to each other with some attributes	
`Loader`	Loads data into the target of an ETL process (in a fact or dimension of a DW)	
`Log`	Logs activity of an ETL mechanism in order to audit the process, assist in data cleaning, performance tuning, etc.	
`Merge`	Integrates two or more data sources with compatible attributes	
`Surrogate`	Generates unique surrogate keys, which are used to replace the original keys in the data sources	123 →
`Wrapper`	Transforms a native data source into a record based data source	

Table 2. ETL mechanisms and their corresponding representation in the UML

names match, the corresponding mappings can be omitted. Furthermore, when some kind of ambiguity may exist (e.g., two attributes with the same name in different sources), the name of the source can be indicated together with the name of the attribute (e.g., Customers.Name and Suppliers.Name).

We do not impose any restriction on the content of these notes, in order to allow the designer the greatest flexibility, but we highly recommend a particular content for each mechanism. The designer can use these notes to define ETL processes at the desired level of detail. For example, the description can be general, specified by means of a natural language, or very detailed, specified by means of a programming language.

3.7 Exportation process

Finally, once the target platform has been chosen (relational, MD, OO, etc.), an exportation process is defined to map the conceptual modeling constructors used in the `DWCS` into the `DWSS`, which corresponds to the storage structures of the target platform. As we have commented in Section 3.4, this mapping can be defined manually or automatically by means of specific transformation algorithms according to the target platform.

4 How to apply the method

Providing a graphical notation is not enough to propose a method, instead a method must specify how to properly use the corresponding graphical notation. Therefore, we propose a set of steps to guide the design of a DW following our approach. Due to the lack of space, we only address the main steps of the method.

In Fig. 3, we have represented the main steps of our method by means of a UML activity diagram. The diagram is divided into two swimlanes depending on who leads the activities: DW final users (final users guide the work of DW designers and administrators) and DW designers & administrators (they do not need the help of final users because all the needed information has been previously gathered). The activities, where the models presented in this paper are applied, are shady: a light color for the activities where the schemas are created (the corresponding schema is indicated on a corner of the activity) and a dark color for the activities where the mappings between schemas are created.

We have classified the activities into four groups: analysis, design, implementation, and test. We do not address following steps such as support and enhancement because we are only interested in the main development steps of a DW.

Finally, transitions define a sequential order of activities[3] and they also show how to use information from other activities.

4.1 Analysis

- Determine initial requirements: different users have different kinds of informational needs, and therefore, the scope of the DW (the business problems to be solved) is established from interviews with final users. Different substeps can be achieved during requirements gathering; specifically, the designer has to:
  - Determine the desired data format users wish, the required detail level, and the particular data elements.
  - Classify different summaries. Some summaries represent business results that are quite significant and are widely used (e.g., annual operating results, quarterly sales figures by region, etc.), whereas others are less common.
  - Help the final users to understand what they do not know they need (anticipate data needs). The designer can trigger ideas of new form of analysis from final users.
  - Define access control and security rules.
  - Discover what are the needs of exploratory analysis (knowledge discovery, data mining, and so on).
  - Catalogue existing analytic and reporting processes.
  - Seek potential new users of the DW.

[3] Actually, some of the steps are not strictly sequential, but in many cases can be accomplished in parallel.

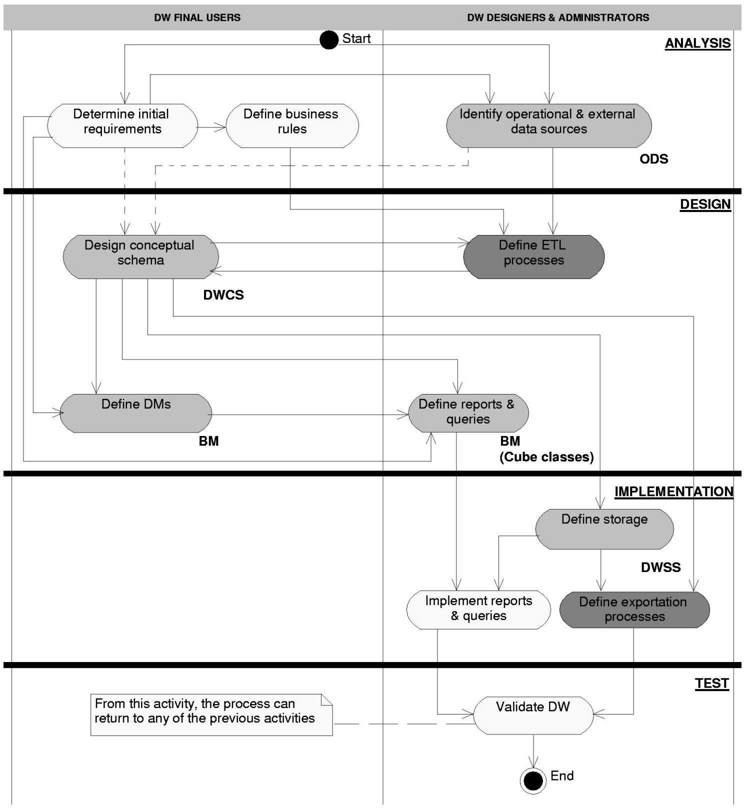


Fig. 3. UML activity diagram that represents the main steps of our method

- Define business rules: from the information collected in the previous activity, the business rules that will be applied during the construction of the DW are defined (e.g., the definition of the derived measures such as "net profit" or "product return rate"). The correct definition of the business rules avoids confusions and ambiguous calculations, and ensures standard interpretation of data.
- Identify operational & external data sources (`ODS`): the final users' requirements are taken into account in order to specify the operational and external data sources. Different questions have to be answered: Do we have the data we need?, Where is that data?, Can we access data?, What is the current data quality?, etc. According to [27], the designer has to scan the available data, dividing them into those that are:

  1. Operational and transitory: they are not valuable for the DW.
  2. Of analytic and historical value: they are candidates for the DW.

3. Of unknown use or value: it is suggested to include data in this category as candidates for DW inclusion as well, *"because a common DW failing is to not have what the consumer wants when they come shopping"*.

4.2 Design

– Design conceptual schema (`DWCS`): two "extreme" strategies (they are represented as dashed lines in Fig. 3) can be adopted in this activity: *top-down* (the definition of the `DWCS` is based on the final users' requirements) or *bottom-up* (the definition of the `DWCS` is based on the available data sources). We suggest to adopt a combined solution: the DW is designed from the final users' requirements, but bearing in mind the available data sources. Moreover, the designer has to understand the final users' requirements and properly match the DW to the intended usage. The designer has to divide data into facts (composed of measures) and dimensions (with the corresponding hierarchies). Finally, the detail level of facts (also called "level of granularity" or "grain of the fact") have to be defined; it is highly better to store data at the most grained level because less detail data can always be obtained from them.
– Define ETL processes: the ETL processes are defined as a mapping between the data sources (`ODS`) and the DW (`DWCS`); the business rules are applied in the calculation of the derived attributes, the attribute transformations, etc. This activity and the previous one define a cycle, because during the definition of the ETL processes some errors in the `DWCS` can be detected (e.g., a dimension attribute does not exist in the `ODS`) and, therefore, the `DWCS` may be modified. The design of an ETL process is usually composed of seven steps: (1) Select the sources for extraction, (2) Clean the sources, (3)Transform the sources, (4) Join the sources, (5) Select the target to load, (6) Map source attributes to target attributes, and (7) Load the data.
– Define DMs (`BM`): different models in the `BM` are defined from the final users' initial requirements and the `DWCS`; the `BM` can be implemented as real or virtual DMs.
– Define reports & queries (`BM`): the queries defined by the final users are defined by means of cube classes.

4.3 Implementation

– Define storage (`DWSS`): the target platform is selected (relational, MD, OO, etc.) and the corresponding logical schema (`DWSS`) is defined; the query performance can be improved by simplifying the data schema (so that it only contains the essential data) or by the definition of summaries (aggregates) based on the final users' requirements.
– Define exportation processes: the mappings between the `DWCS` and the `DWSS` are defined; these mappings can be manually or automatically defined.
– Implement reports & queries: the reports and queries requested by the final users are implemented in the query tool used (generally, an OLAP application).

4.4 Test

– Validate DW: the solution obtained (the DW built) is checked against the existing problem (final users' requirements). If any discrepancy exists, some corrective actions can be taken and the process can return to one of the previous activities.

5 Conclusions and future works

In this paper, we have presented a global data warehouse (DW) design method that is based on the use of the UML as the modeling language. The best advantage of our global approach is that we always use the same notation (UML) for designing the different DW schemas and the corresponding transformations between them in an integrated manner. Moreover, thanks to the use of the UML packages, our method can scale up to handle huge and complex DWs (from simple data marts to complex enterprise DWs), avoiding the DW developer using a flat approach.

Regarding future works, we are working on a set of diagramming and style guidelines (heuristics) for creating better diagrams with our method. We also plan to incorporate in our method more stages of the DW life cycle, such as the design of the refresh processes.

References

1. Inmon, W.: Building the Data Warehouse. QED Press/John Wiley (1992) (Last edition: 3rd edition, John Wiley & Sons, 2002).
2. 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., Valencia, Spain (1998) 183–197
3. Golfarelli, M., Rizzi, S.: A methodological Framework for Data Warehouse Design. In: Proc. of the ACM 1st Intl. Workshop on Data Warehousing and OLAP (DOLAP'98), Bethesda, USA (1998) 3–9
4. Sapia, C., Blaschka, M., Höfling, G., Dinter, B.: Extending the E/R Model for the Multidimensional Paradigm. In: Proc. of the 1st Intl. Workshop on Data Warehouse and Data Mining (DWDM'98). Volume 1552 of LNCS., Singapore (1998) 105–116
5. 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)
6. Husemann, B., Lechtenborger, J., Vossen, G.: Conceptual Data Warehouse Design. In: Proc. of the 2nd Intl. Workshop on Design and Management of Data Warehouses (DMDW'00), Stockholm, Sweden (2000) 3–9
7. Trujillo, J., Palomar, M., Gómez, J., Song, I.: Designing Data Warehouses with OO Conceptual Models. IEEE Computer, special issue on Data Warehouses **34** (2001) 66–75
8. Abelló, A., Samos, J., Saltor, F.: YAM2 (Yet Another Multidimensional Model): An Extension of UML. In: Intl. Database Engineering & Applications Symposium (IDEAS'02), Edmonton, Canada (2002) 172–181

9. Abelló, A., Samos, J., Saltor, F.: A Framework for the Classification and Description of Multidimensional Data Models. In: Proc. of the 12th Intl. Conf. on Database and Expert Systems Applications (DEXA'01), Munich, Germany (2001) 668–677
10. Object Management Group (OMG): Unified Modeling Language Specification 1.4. Internet: http://www.omg.org/cgi-bin/doc?formal/01-09-67 (2001)
11. Gardner, S.: Building the Data Warehouse. Communications of the ACM **41** (1998) 52–60
12. Debevoise, N.: The Data Warehouse Method. Prentice-Hall, New Jersey, USA (1999)
13. Corey, M., Abbey, M., Abramson, I., Taub, B.: Oracle8i Data Warehousing. Oracle Press. Osborne/McGraw-Hill (2001)
14. Kimball, R.: The Data Warehouse Toolkit. John Wiley & Sons (1996) (Last edition: 2nd edition, John Wiley & Sons, 2002).
15. Moody, D., Kortink, M.: From Enterprise Models to Dimensional Models: A Methodology for Data Warehouse and Data Mart Design. In: Proc. of the 3rd Intl. Workshop on Design and Management of Data Warehouses (DMDW'01), Interlaken, Switzerland (2001) 1–10
16. Cavero, J., Piattini, M., Marcos, E.: MIDEA: A Multidimensional Data Warehouse Methodology. In: Proc. of the 3rd Intl. Conf. on Enterprise Information Systems (ICEIS'01), Setubal, Portugal (2001) 138–144
17. Carneiro, L., Brayner, A.: X-META: A Methodology for Data Warehouse Design with Metadata Management. In: Proc. of the 4th Intl. Workshop on Design and Management of Data Warehouses (DMDW'02), Toronto, Canada (2002) 13–22
18. Rational Software Corporation: The UML and Data Modeling. Internet: http://www.rational.com/media/whitepapers/Tp180.PDF (2000)
19. Marcos, E., Vela, B., Cavero, J.: Extending UML for Object-Relational Database Design. In: Proc. of the 4th Intl. Conf. on the Unified Modeling Language (UML'01). Volume 2185 of LNCS., Toronto, Canada (2001) 225–239
20. Naiburg, E., Maksimchuk, R.: UML for Database Design. Object Technology Series. Addison-Wesley (2001)
21. Abelló, A., Samos, J., Saltor, F.: Benefits of an Object-Oriented Multidimensional Data Model. In: Proc. of the Symposium on Objects and Databases in 14th European Conf. on Object-Oriented Programming (ECOOP'00). Volume 1944 of LNCS., Sophia Antipolis and Cannes, France (2000) 141–152
22. Jarke, M., Lenzerini, M., Vassiliou, Y., Vassiliadis, P.: Fundamentals of Data Warehouses. 2 edn. Springer-Verlag (2003)
23. Rational Software Corporation: Migrating from XML DTD to XML-Schema using UML. Internet: http://www.rational.com/media/whitepapers/TP189draft.pdf (2000)
24. Luján-Mora, S., Trujillo, J., Song, I.: Extending UML for Multidimensional Modeling. In: Proc. of the 5th Intl. Conf. on the Unified Modeling Language (UML'02). Volume 2460 of LNCS., Dresden, Germany (2002) 290–304
25. Luján-Mora, S., Trujillo, J., Song, I.: Multidimensional Modeling with UML Package Diagrams. In: Proc. of the 21st Intl. Conf. on Conceptual Modeling (ER'02). Volume 2503 of LNCS., Tampere, Finland (2002) 199–213
26. Trujillo, J., Luján-Mora, S.: A UML Based Approach for Modeling ETL Processes in Data Warehouses. In: Proc. of the 22nd Intl. Conf. on Conceptual Modeling (ER'03). LNCS, Chicago, USA (2003) (To be presented).
27. Haisten, M.: Designing a Data Warehouse. InfoDB **9** (1995) 2–9