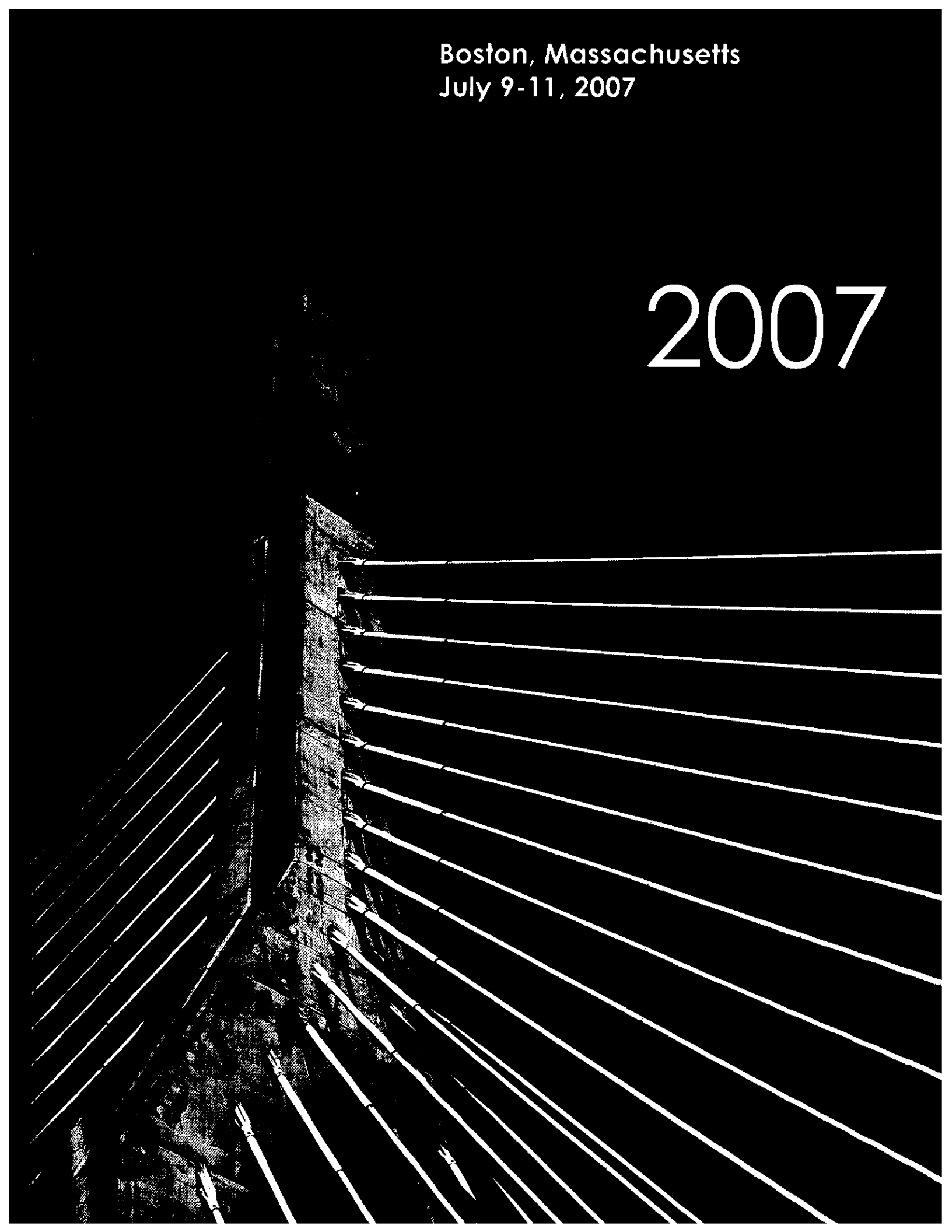
Boston, Massachusetts
July 9-11, 2007

2007



PROCEEDINGS

SEKE 2007

The 19th International Conference on Software Engineering & Knowledge Engineering

Sponsored by

Knowledge Systems Institute Graduate School, USA

Technical Program

July 9-11, 2007

Hyatt Harborside Hotel, Boston, Massachusetts, USA

Organized by

Knowledge Systems Institute Graduate School

ISBN 1-891706-20-9 (paper)

Additional Copies can be ordered from:
Knowledge Systems Institute Graduate School
3420 Main Street
Skokie, IL 60076, USA
Tel:+1-847-679-3135
Fax:+1-847-679-3166
Email:office@ksi.edu
http://www.ksi.edu

Proceedings preparation, editing and printing are sponsored by
Knowledge Systems Institute Graduate School

Printed by Knowledge Systems Institute Graduate School

Table of Contents

Software Processes and Engineering Practice

Aspect-Based Software Development

Software Testing and Quality Assurance

System Requirements Analysis, Modeling and Specification

Web-Based Applications

Web Technology and Web Engineering

Software Project and Resource Management

Software Reuse and Component Technology

Software System Maintenance

Knowledge Engineering, Natural Language Processing, and AI

Plenary Talk

Database Retrieval Methods

Data

Software Development and Design Pattern

Data Warehouse

Data Mining and Machine Learning

System and Software Architecture

Applications

Model-Driven Software Development

Agent-Based Technology and Intelligence

DB Access and Query Processing

Service-Oriented Technology and Web Technology

System Reliability and Verification

Agent Modeling/Methodology

Agent Applications

Human Interaction and GUI Development

Service

Security

Grid Technology

Software Metrics, Measurement and Evaluation

Industrial Workshop

Note: (S) means short paper.

A Proposal for a Conceptual Data Warehouse Quality Model

Manuel Serrano, Rafael Romero, José-Norberto Mazón, Juan Trujillo, and Mario Piattini

Abstract— **Data warehouses are considered a cornerstone of a decision support system, since they provide the adequate information resources for decision making in an integrated manner. Therefore, the quality of a data warehouse is a key issue in any business organization, and several works have pointed out the importance of measuring the quality in data warehouse domain. However, to the best of our knowledge, the current approaches that deal with the quality aspects of data warehouses only consider isolated quality metrics, without being part of a comprehensive quality model that allows designers to assess the quality of a data warehouse in a systematic and objective way. Having considered this limitation in current data warehouse research, in this paper we present a quality model for the data warehouse domain. This model allows designers to state the meaning of quality in the data warehouse domain, thus measuring the quality of a data warehouse conceptual model in a systematic and objective way. We have used a well-known methodology to build this quality model, and we have related the elements of this quality model to our previously defined and validated quality metrics for conceptual models of data warehouses.**

Index Terms— **Data Warehouse, Quality, Quality Model**

I. Introduction

Nowadays, data warehouses have become one of the most crucial components of decision support systems, since they efficiently and effectively allow the integrated management of data in order to provide information resources that support effective problem and opportunity identification, critical decision-making, and strategy formulation, implementation, and evaluation. Therefore, data warehouse technology has become an adequate solution, providing such resources in an environment where increasing competition, sophisticated and informed costumers, unpredictable market fluctuations, and changing regulatory environments are putting much pressure on business organizations [1].

As a result of this importance, companies are increasing their investment in the development of data warehouses. In fact, the investment in this kind of systems is around several millions a year and increases by 20% per year [2]. However, despite the potential of data warehouses and the huge amount of money invested, success is not necessarily guaranteed. As stated by several authors, a data warehouse project is a real risk [3] and more than 60% of data warehouses do not meet the user expectations [4]. Therefore, designers have to deal with quality issues to avoid these pitfalls and guarantee the success of the data warehouse project [5].

Dealing with quality issues is not an easy task, since quality is an abstract and subjective aspect, for which there is no universal definition: it is usually said that there is a quality definition for each person. In this way, it is very complex to measure or assess the quality of a software product in an objective way. Several guides have been proposed to address the complexity of data warehouse quality, but most of the time, guides are not enough, since they can help designers in their work, but they imply rather subjective decisions, and this can lead to "not-so-good" products.

In order to overcome this inherent subjectivity of quality in data warehouse projects, in this work we introduce a quality model that helps designers to assess the quality of a data warehouse in an objective way, thus guaranteeing success in designing a good data warehouse. In this paper, we focus on the quality of conceptual data warehouse models and we present the first steps towards building a data warehouse quality model. We focus on the quality of the conceptual models, as a *good* design may (or may not) lead to a *good* system, but a *bad* design will surely render a *bad* product of low quality. Thus, it is important to focus on the quality of the product from the first steps of its development, i.e. the conceptual model.

The remainder of this paper is structured as follows. The next section shows related work regarding data warehouse quality and models. In Section 3, a method for developing quality models based on ISO 9126 [6] is presented. In Section 4, the method is applied to the building of a data warehouse quality model. The last section presents some conclusions and future work arising from this work.

II. Related Work

From the beginnings of data warehousing, quality has been an important issue, from the perspectives of both data and

Manuscript received March 8, 2007. This work was supported in part by the CALIA Project supported by the University of Castilla – La Mancha and the DIMENSIONS Project (PBC-05-012-1) supported by Consejería de Educación y Ciencia – Junta de Comunidades de Castilla – La Mancha. Jose-Norberto Mazón is funded by the Spanish Ministry of Education and Science under a FPU grant (AP2005-1360)

Manuel Serrano (corresponding author, Phone: +34 926295300) and Mario Piattini are with the Alarcos Research Group, University of Castilla – La Mancha, Ciudad Real, Spain (e-mail: Manuel.Serrano@uclm.es and Mario.Piattini@uclm.es).

Rafael Romero, José-Norberto Mazón and Juan Trujillo) are with the Department of Software and Computing Systems, University of Alicante, Alicante, Spain (e-mail: romero@dlsi.ua.es, jnmazon@dlsi.ua.es and jtrujillo@dlsi.ua.es).

schema. This interest in quality is motivated by the importance of the information quality in the decision support systems.

Several techniques have been used in an attempt to improve the quality of data warehouses, such as software process improvement models. Among these techniques we can find the Capability Maturity Model [7] or SPICE [8], which relies on the belief that *"good processes deliver good software"*, but unfortunately good processes do not guarantee good software. Good processes only increase the probability that good software may be obtained.

In this search for quality, several quality standards have been applied to data warehouse development (such as ISO 9126 [6] and IEEE 1061 [9], usually complemented by GQM techniques [10, 11]) to attempt to measure the quality of the data warehouse. These standards are too general and complex and are usually not concrete enough to be useful in assessing the quality of the system.

With regard to data warehouse quality measurement we can find several proposals for measuring some quality dimensions of data warehouses such as the metrics proposed by [12] and [5]. These proposals are good approaches to data warehouse measurement but they are not complete, since they are not part of a quality model that allows designers to use them in a systematic and objective way.

Lastly, we can find the proposal of a data warehouse quality model, named DWQ [13] which attempted to assure the quality of the data stored inside the data warehouse to improve the data warehousing experience; this project had some quality dimensions and focused on data quality. We think that data warehouse quality should not only be assessed in terms of data quality and that schema quality is as important an issue as data quality. In this paper we propose a quality model for conceptual data warehouse models, based on the ISO 9126 standard.

III. Quality Model Development Method

Developing a quality model is not an easy task. Several aspects should be considered and the whole process should be carried out in an objective and methodological way. For this purpose, Franch and Carvallo [14] proposed a method to build quality models based on the ISO 9126-1 standard [6]. This method is now briefly described.

The proposed methodology comprises of six steps and also considers a preliminary activity (step 0). A graphical view of the methodology is shown in Figure 1. Although the steps seem to be sequential, these steps can be intertwined and repeated if needed.

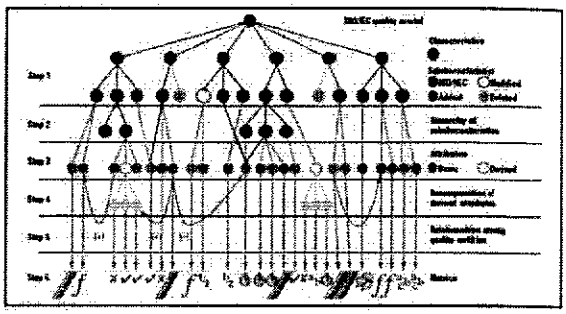


Fig. 1. Franch and Carvallo [14] six steps methodology

Step 0. Defining the domain

Previous to the application of the methodology, it is necessary to carefully examine and describe the domain of interest, with the help of experts. In order to describe the domain the use of conceptual modelling to keep track of relevant concepts is recommended.

Step 1. Determining quality subcharacteristics

The decomposition of characteristics into subcharacteristics that appear in the ISO standard is quite reasonable and should be used unless good reasons for not doing so emerge during domain analysis. In this case, new subcharacteristics are added, some of them are refined and it is even permisable to remove some of the subcharacteristics that do not apply to the domain.

Step 2. Defining a hierarchy of subcharacteristics

Typically, further decomposition of subcharacteristics with respect to some factors is needed. In this way, we obtain a hierarchy of subcharacteristics.

Step 3. Decomposing subcharacteristics into attributes

Quality subcharacteristics provide a comprehensible abstract view of the quality model. But we need to decompose these abstract concepts into more concrete ones - the quality attributes. An attribute keeps track of a particular observable feature of the packages in the domain. Attributes should be precisely defined to clarify the underlying quality concepts that they represent and to link them with the appropriate subcharacteristics.

As the standard itself mentions, it is not possible, from a practical point of view, to measure all the subcharacteristics in the entirety of a software product, but we can create a complete list of those which are most relevant.

Step 4. Decomposing derived attributes into basic ones

Some of the attibutes obtained in Step 3 can be directly measured, but others may be so abstract that they require more decomposition. In this way, attributes are classified as being 'derived' and 'basic'. Derived attributes should be decomposed until they are completely expressed in terms of those which are basic.

Step 5. Stating relationships between quality entities

If we wish to obtain a complete quality model, we must state the relationships between quality entities. The model becomes more exhaustive and the implications of quality requirements become clearer.

We can classify the relationships into three types:

- Collaboration: Growing the first quality entity implies growing the second quality entity.
- Damage: Growing the first quality entity implies decreasing the second quality entity
- Dependency: Some values of the first quality attribute require that the second one fulfils certain conditions.

Step 6. Determining metrics for attributes.

It is not only necessary to identify the attributes but also to select metrics for all the attributes. For this task, we can use the general theory of metrics.

IV. Data Warehouse Quality Model

In this section, we explain how to apply the method described in the previous section to develop a quality model for data warehouse domain. We have developed each of the proposed steps, and we have even related our previously defined metrics to this quality model.

Step 0. Defining the domain

In this step we define the domain for which we want to build a quality model, in this case the conceptual data warehouse schemata. Next, we outline an approach to data warehouse conceptual modeling, based on the UML (Unified Modeling Language). This approach has been specified by means of a UML profile[1] which contains the stereotypes necessary to carry out conceptual modeling successfully [15]. This profile contains the stereotypes which are necessary to elegantly represent main multidimensional (MD) properties at the conceptual level (see Table 1). Specifically, the structural properties of MD modeling are represented by means of a UML class diagram in which the information is clearly organized into facts and dimensions. These facts and dimensions are respectively represented by Fact and Dimension classes. Fact classes are defined as composite classes in shared aggregation relationships of n Dimension classes. The minimum cardinality in the role of the Dimension classes is 1 to indicate that every fact must always be related to all the dimensions. A fact is composed of measures or fact attributes. These are represented as attributes with the FactAttribute stereotype By default, all measures in the Fact class are considered to be additive. For non-additive measures, additive rules are defined as constraints and are included in the Fact class. Furthermore, derived measures (indicated by /) and their derivation rules can also be explicitly represented as tagged values of a FactAttribute.

[1] A *profile* is a set of improvements that extends an existing UML type of diagram to a different use. These improvements are specified by means of the extendibility mechanisms provided by UML (stereotypes, properties and restrictions) in order to be able to adapt it to a new method or model.

Our approach also allows the definition of degenerate dimensions, thereby representing other fact features in addition to the measures for analysis. These degenerated dimensions are represented as stereotyped attributes of the Fact class (DegenerateDimension stereotype).

The many-to-many relationships between a fact and a specific dimension are specified by means of the cardinality 1...n in the role of the corresponding Dimension class. In this case, we usually need to describe specific attributes to provide further features for every instance combination in this particular relationship. In doing so, the provided attributes are usually called degenerate facts. These degenerate facts are represented as an association class attached to a many-to-many aggregation relationship between a Fact class and a Dimension class. This DegenerateFact class may contain FactAttributes and DegenerateDimensions.

With respect to dimensions, each level of a classification hierarchy is specified by a Base class. Every Base class may contain several dimension attributes (DimensionAttribute stereotype), an identifying attribute (OID stereotype), and must also contain a Descriptor attribute (D stereotype). An association (represented by a stereotype called Rolls-UpTo) between Base classes specifies the relationship between two levels of a classification hierarchy. The only prerequisite is that these classes must define a Directed Acyclic Graph (DAG) rooted in the Dimension class (the DAG constraint is defined in the stereotype Dimension). The DAG structure can represent both multiple and alternative path hierarchies. A Dimension class contains a unique first hierarchy (or dimension) level called a terminal dimension level. A roll-up path is a subsequence of dimension levels, which starts in this terminal level (lower level of detail) and ends in an implicit level (not graphically represented) that represents all the dimension levels.

We use roles to represent the way in which the two Base classes see each other in a Rolls-UpTo association: role R represents the direction in which the hierarchy rolls-up, whereas role D represents the direction in which the hierarchy drills-down. Moreover, we use roles to detect and avoid cycles in a classification hierarchy, and therefore, to help us to achieve the DAG condition.

Due to the flexibility of UML, we can also consider non-strict hierarchies (an object at a hierarchy's lower level belongs to more than one higher-level object) and complete hierarchies (all members belong to one higher-class object and that object consists of those members only). These characteristics are specified, respectively, by means of the cardinality of the roles of the associations and by defining the stereotype Completeness in the association between Base classes. Lastly, the categorization of dimensions is considered by means of the generalization/specialization relationships of UML.

Our profile is formally defined and uses the Object Constraint Language (OCL) to express well-formed rules of the new defined elements (see Table 1), thereby avoiding its arbitrary use. We refer the reader to [15] for a further explanation of this profile and its corresponding OCL constraints.

TABLE 1.
MAIN STEREOTYPES OF OUR UML PROFILE FOR MD MODELING OF DW.

Stereotype	Notation
Fact	
Dimension	
Base	
DegenerateFact	
FactAttribute	FA
DegenerateDimension	DD
DimensionAttribute	DA
OID	OID
Descriptor	D
Rolls-UpTo	<<Rolls-UpTo>>
Completeness	<<Completeness>>

Step 1. Determining quality subcharacteristics

In this step the quality characteristics are decomposed into subcharacteristics following the ISO 9126 standard and the characteristics of the domain we are working with.

The ISO 9126 proposes the decomposition shown in Table 2.

TABLE 2.
ISO/IEC 9126-1 CHARACTERISTICS AND SUBCHARACTERISTICS

Characteristics	*Subcharacteristics*
Functionality	Suitability Accuracy Interoperability Security
Reliability	Maturity Fault Tolerance Recoverability
Usability	Understandability Learnability Operability Attractiveness
Efficiency	Time behaviour Resource utilization
Maintainability	Analyzability Changeability Stability Testability
Portability	Adaptability Installability Coexistence Replaceability

In our domain, data warehouse conceptual models, we firmly believe that some of the dimensions of the ISO 9126 model are not applicable, since the quality of a conceptual data warehouse schema is related to the correctness, completeness, understandability, security and stability of that schema. Those dimensions that are not related to these characteristics have been removed. In Table 3 we can find the subcharacteristics that are taken into account after removing those that are not applicable to our domain.

TABLE 3. FINAL SET OF QUALITY SUBCHARACTERISTICS

Characteristics	*Subcharacteristics*
Functionality	Suitability Accuracy Security
Usability	Understandability Learnability
Maintainability	Analyzability Changeability Stability

Step 2. Defining a hierarchy of subcharacteristics

In our context (quality of conceptual data warehouse models), there is no need to decompose the subcharacteristics stated in the previous step into smaller ones.

Step 3. Decomposing subcharacteristics into attributes

As we have previously stated, quality subcharacteristics provide a comprehensible abstract view of the quality model, but they are too abstract and difficult to measure. In this step, the abstract concepts are descomposed into attributes. For each relevant characteristic and subcharacteristic we provide a set of attributes that are related to that characteristic (see Table 4).

TABLE 4. ATTRIBUTES RELATED TO EACH QUALITY SUBCHARACTERISTIC

Characteristics	Subcharacteristics	Attribute
Functionality	Suitability	Adequacy of data modelled to requirements Completeness Well-Specified Semantic correctness
	Accuracy	Accurate to requirements Semantic correctness
	Security	Presence of security constraints Adequacy of security rules
Usability	Understandability	Ease of understanding Schema Complexity Schema Size Schema Depth
	Learnability	Ease of understanding Schema Complexity Schema Size Schema Depth
Maintainability	Analyzability	Ease of understanding Schema Complexity Schema Size Schema Depth Coupling Cohesion Normalization Modularity
	Changeability	Ease of understanding Schema Complexity Coupling Cohesion Normalization Modularity
	Stability	Schema Complexity Coupling Cohesion Normalization Modularity

Step 4. Decomposing derived attributes into basic ones

After Step 3 we may find that some attributes are too abstract to be easily measured and we should decompose them into more concrete ones. However, the attributes presented in Table 4 are concrete enough, so it is therefore unnecessary to further decompose them.

Step 5. Stating relationships between quality entities

In this step we state the relationships between quality entities, so the model becomes more exhaustive and its implications become clearer. Table 5 shows the relationships between quality attributes. In this table 'COL' means Collaboration, 'DEP' means Dependency and 'DMG' means Damage.

TABLE 5. RELATIONSHIPS BETWEEN QUALITY ATTRIBUTES

	Adequacy of data modeled to requirements	Completeness	Well-Specified	Semantic correctness	Accurate to requirements	Presence of security constraints	Adequacy of security rules	Ease of Understanding	Schema Complexity	Schema Size	Schema Depth	Coupling	Cohesion	Normalization
Completeness	COL													
Well-Specified	COL	COL												
Semantic correctness	COL	COL	COL											
Accurate to requirements	COL	COL	COL	COL										
Presence of security constraints	DEP	COL	COL	COL	COL									
Adequacy of security rules		COL		COL		DEP								
Ease of understanding		DMG	DMG	COL	COL	DMG								
Schema Complexity	DMG	DMG	DMG	DMG	DMG	DMG	COL	DMG						
Schema Size	DMG	DMG	DMG	DMG	DMG			DMG						
Schema Depth	DMG	DMG	DMG	DMG	DMG			DMG	COL					
Coupling			DMG	DMG	DMG									
Cohesion			DMG	DMG	DMG							DMG		
Normalization	DMG					DMG		COL				COL	COL	
Modularity				DMG	COL			COL	COL			COL	COL	COL

For example in Table 5, we can see that the Modularity collaborates with the Ease of understanding because if the system is modular it is probably easier to understand. This table is a summary of all the relationships between the quality attributes.

Step 6. Determining metrics for attributes.

After identifying the attributes, we must assign metrics to each attribute. Table 6 shows the proposed metric for each attribute.

TABLE 6.
METRICS ASSOCIATED TO EACH QUALITY CHARACTERISTIC.

Characteristics	*Subcharacteristics*	*Attribute*	*Metric*
Functionality	Suitability	Adequacy of data modelled to requirements	Number of redundant requirements Number of requirements mapped into the model
		Completeness	Number of missing requirements Number of missing attributes Number of missing elements Number of missing relationships Number of requirements not mapped into the model
		Well-Specified	Number of wrong specifications Number of redundant specifications
		Semantic correctness	Number of requirements mapped into the schema Adequacy to requirements
	Accuracy	Accurate to requirements	Number of missing requirements Number of elements that are not represented in the requirements
	Security	Presence of security constraints	Number of security constraints Ratio of security constraints
		Adequacy of security rules	Number of security rules Complexity of security rules
Usability	Understandability	Ease of understanding	Number of classes Number of relationships Ratio of relationships Adequacy of names
		Schema Complexity	Number of relationships Ratio of relationships
		Schema Size	Number of classes Number of relationships
		Schema Depth	Length of schema Depth of hierarchy tree
	Learnability	Ease of understanding	Number of classes Number of relationships Ratio of relationships Adequacy of names
		Schema Complexity	Number of relationships Ratio of relationships
		Schema Size	Number of classes Number of relationships
		Schema Depth	Length of schema Depth of hierarchy tree
Maintainability	Analyzability	Ease of understanding	Number of classes Number of relationships Ratio of relationships Adequacy of names
		Schema Complexity	Number of relationships Ratio of relationships
		Schema Size	Number of classes Number of relationships
		Schema Depth	Length of schema Depth of hierarchy tree
		Coupling	Schema coupling

	Cohesion	Schema cohesion
	Normalization	Normalization complaint
	Modularity	Number of modules Ratio of modules
Changeability	Ease of understanding	Number of classes Number of relationships Ratio of relationships Adequacy of names
	Schema Complexity	Number of relationships Ratio of relationships
	Coupling	Schema coupling
	Cohesion	Schema cohesion
	Normalization	Normalization complaint
	Modularity	Number of modules Ratio of modules
Stability	Ease of understanding	Number of classes Number of relationships Ratio of relationships Adequacy of names
	Schema Complexity	Number of relationships Ratio of relationships
	Coupling	Schema coupling
	Cohesion	Schema cohesion
	Normalization	Normalization complaint
	Modularity	Number of modules Ratio of modules

V. Conclusions and Future Work

Nowadays, data warehouses have become an important software product used in decision support systems. Due to the environment in which data warehouses are used, assuring the quality of these systems is a crucial issue. When dealing with data warehouse quality, it is important to assess the quality of both data and schemas, as the quality of the data and also the whole system is affected by the quality of the model that supports them.

In order to assess the quality of the data warehouse schema, it is necessary to define a quality model in order to help designers in assuring the quality of the final system as it is being developed in a systematic and objective way.

In this paper we have reviewed a method for creating a quality model based on the ISO 9126 standard and we have applied its six steps to create a first proposal of a conceptual data warehouse quality model.

As a result of this method we have obtained a set of important quality characteristics which are broken down into sub-characteristics and quality attributes. As a final step we have proposed a set of initial metrics for assessing those quality attributes, and we have related them to our previously defined and validated metrics.

The next step in this process of proposing a quality model is to validate it. We are planning to begin with the validation of the proposed metrics in order to assure their usefulness and adequacy. After, we believe that we should attempt to validate the whole quality model in an industrial environment.

References

[1] B. Shin, "An Exploratory Investigaction of System Success Factors in Data Warehousing," *Journal of Association for Information Systems*, vol. 4, pp. 141-170, 2003.

[2] T. Chenoweth, D. Schuff, and R. St. Louis, "A method for developing Dimensional data marts," *Communications of the ACM*, vol. 46, pp. 93-98, 2003.

[3] P. Vassiliadis, "Gulliver in the land of data warehousing: practical experiences and observations of a researcher," presented at International Workshop on Design and Management of Data Warehouses (DMDW'2000), Stockholm (Sweden), 2000.

[4] C. Stedman, "Warehousing Projects Hard to Finish," *Computerworld*, vol. 32, pp. 29, 1998.

[5] N. Prat and S. S.-S. Cherfi, "Multidimensional Schemas Quality Assessment," presented at The 15th Conference on Advanced Information Systems Engineering (CAiSE '03) Workshops, Klagenfurt/Velden (Austria), 2003.

[6] ISO/IEC, "9126-1: Software Engineering - Product quality - Part 1: Quality model.," 2001.

[7] M. C. Paulk, B. Curtis, M. B. Chrissis, and C. V. Weber, "The Capability Maturity Model for software," Software Engineering Institute, Carnegie Mellon University CMU/SEI-93-TR-024, February 2003 1993.

[8] K. El Emam, J. N. Drouin, and W. Melo, "Spice: The Theory and Practice of Soft-ware Process Improvement and Capability Determination," IEEE Computer Society 1998.

[9] IEEE, "IEEE Std 1061-1998 IEEE Standard for a Software Quality Metrics Methodology," 1998.

[10] V. Basili and D. Weiss, "A Methodology for Collecting Valid Software Engineering Data," *IEEE Transactions on Software Engineering*, vol. 10, pp. 728-738, 1984.

[11] V. Basili and H. Rombach, "The TAME project: towards improvement-oriented software environments," *IEEE Transactions on Software Engineering*, vol. 14(6), pp. 728-738, 1988.

[12] M. Serrano, "Definition of a set of metrics for assuring data warehouse quality," Univeristy of Castilla - La Mancha (Spain), 2004.

[13] M. Jarke, M. Lenzerini, Y. Vassiliou, and P. Vassiliadis, *Fundamentals of Data Warehouses*, second edition ed: Springer-Verlag., 2002.

[14] X. Franch and J. P. Carvallo, "Using Quality Models in Software Package Selection," *IEEE Software*, vol. 20, pp. 34-41, 2003.

[15] S. Luján-Mora, J. Trujillo, and I.-Y. Song, "A UML profile for multidimensional modeling in data warehouses," *Data & Knowledge Engineering (DKE)*, vol. 59, pp. 725–769, 2006.