Towards Quality-Oriented Data Warehouse Usage and Evolution

Panos Vassiliadis[1], Mokrane Bouzeghoub[2], Christoph Quix[3]

[1] National Technical University of Athens, Greece, pvassil@dbnet.ece.ntua.gr
[2] University of Versailles and INRIA, France, Mokrane.Bouzeghoub@prism.uvsq.fr
[3] Informatik V, RWTH Aachen, Germany, quix@informatik.rwth-aachen.de

Abstract. As a decision support information system, a data warehouse must provide high level quality of data and quality of service. In the DWQ project we have proposed an architectural framework and a repository of metadata which describes all the data warehouse components in a set of metamodels to which is added a quality metamodel, defining for each data warehouse metaobject the corresponding relevant quality dimensions and quality factors. Apart from this *static* definition of quality, we also provide an *operational* complement, that is a methodology on how to use quality factors and to achieve user quality goals. This methodology is an extension of the Goal-Question-Metric (GQM) approach, which allows to capture (a) the inter-relationships between different quality factors and (b) to organize them in order to fulfil specific quality goals. After summarizing the DWQ quality model, this paper describes the methodology we propose to use this quality model, as well as its impact on the data warehouse evolution.

1. Introduction

Many researchers and practitioners share the understanding that a data warehouse (DW) architecture can be formally understood as layers of materialized views on top of each other. A DW architecture exhibits various layers of data in which data from one layer are derived from data of the lower layer. *Data sources*, also called *operational databases*, form the lowest layer. They may consist of structured data stored in open database systems and legacy systems, or unstructured or semi-structured data stored in files. The central layer of the architecture is the *global* (or *primary*) *DW*. The global DW keeps a historical record of data that result from the transformation, integration, and aggregation of detailed data found in the data sources. Usually, a data store of volatile, low granularity data is used for the integration of data from the various sources: it is called *Operational Data Store (ODS)*. The Operational Data Store, serves also as a buffer for data transformation and cleaning so that the DW is populated with clean and homogeneous data. The next layer of views are the *local*, or *client* warehouses, which contain highly aggregated data, directly derived from the global warehouse. There are various kinds of local warehouses, such as the *data marts* or the *OLAP databases* which may use relational database systems or specific multidimensional data structures.

All the DW components, processes and data are -or at least should be- tracked and administered from a *metadata repository*. The metadata repository serves as an aid both to the administrator and the designer of a DW. Indeed, the DW is a very complex system, the volume of recorded data is vast and the processes employed for its extraction, transformation, cleansing, storage and aggregation are numerous, sensitive to changes and time-varying. The metadata repository serves as a maproad which provides a trace of all design choices and a history of changes performed on its architecture and components. For example, the new version of the *Microsoft Repository* [1] and the *Metadata Interchange Specification* (MDIS) [9] provide different models and application programming interfaces to control and manage metadata for OLAP databases. In figure 1, a generic architecture for a DW is depicted.

As a decision support information system, a DW must provide high level quality of data and quality of service. Coherency, freshness, accuracy, accessibility, availability and performance are among the quality features required by DW users. The metadata repository plays a central role in this concern, as it provides the necessary knowledge to understand, evaluate and analyze current DW architecture in order to predict its behaviour and the resulting quality of service and quality of data.

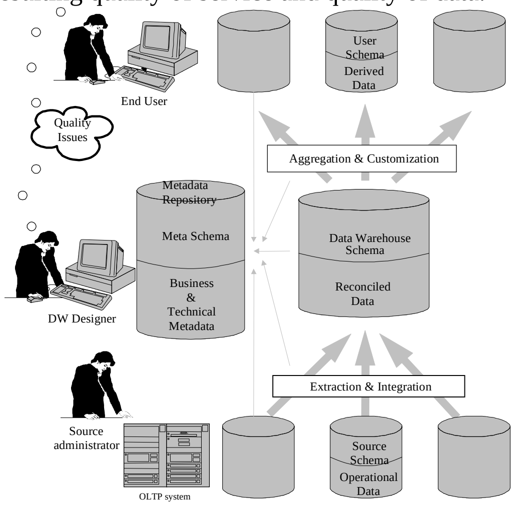


Fig. 1. A generic architecture for a DW

Data quality has been defined as the fraction of performance over expectancy, or as the loss imparted to society from the time a product is shipped [3]. We believe, though, that the best definition is the one found in [13,11,17,14]: data quality is defined as "fitness for use". The nature of this definition directly implies that the concept of data quality is relative. For example, data semantics (the interpretation of information) is different for each distinct user. As [11] mentions "the problem of data quality is fundamentally intertwined in how [...] users actually use the data in the system", since the users are actually the ultimate judges of the quality of the data produced for them: if nobody actually uses the data, then nobody will ever take care to improve its quality.

From the previous it follows that, on one hand, the quality of data is of highly *subjective* nature and should ideally be treated differently for each user. But, on the other hand, the reasons for data deficiencies, non-availability or reachability problems are definitely *objective*, and depend mostly on the information system definition and implementation. Furthermore, the prediction of data quality for each user must be based on objective quality factors which are computed and compared to users' expectations. The question that arises, then, is how to tune the design choices in such a way that all the different, and sometimes opposing, user requirements can be simultaneously satisfied. As the number of users and the complexity of DW systems do not permit to reach total quality for every user, the subsidiary question is how to prioritize these requirements in order to satisfy them with respect to their importance. This problem is typically illustrated by the physical design of the DW where the problem is to find a set of materialized views which optimize user requests response time and the global DW maintenance cost.

In [6] a metadata modeling approach is presented that enables the capturing of all the crucial parts of the architecture of a DW, along with information over different quality dimensions of these components. In this paper, we have refined the quality metamodel with a more detailed linkage between objective quality factors and user-dependent quality goals. Moreover, we have extended the Goal-Question-Metric (GQM) methodology [2] in order (a) to capture the inter-relationships between different quality factors with respect to a specific quality goal, and (b) to define an appropriate lifecycle which deal with quality goal evaluation and improvement.

Our methodology comprises a set of steps aiming, in one hand, to map a high-level subjective quality goal into the measurement of a set of interrelated quality factors, and, in the other hand, to propose improvement actions which may help in achieving the target quality goal. These steps involve the *design* of the quality goal, the *evaluation* of the current status, the *analysis* and *improvement* of this situation, and finally, the *re-evaluation* of the achieved plan. The metadata repository together with this quality goal definition methodology constitute a decision support system which helps DW designers and administrators to take relevant decisions to achieve reasonable quality level which fits the best user requirements. This work is being integrated in a methodology for DW quality design, that is developed in the European DWQ project (Foundations of Data Warehouse Quality) [8].

We want to stress out that we do not follow the ISO 900x paradigm in our approach; rather we try to present a computerized approach to the stakeholder, for both the storage and exploitation of information relevant to the quality of the DW. The objective of this paper is to show how subjective quality goals can be evaluated using more objective quality factors, following an extended GQM approach.

The paper is organized as follows: section 2 describes the DWQ quality metamodel and an example for its instantiation. In section 3, we detail the DWQ methodology for quality management. Section 4 presents some hints on DW evolution. Section 5 summarizes related work and finally section 6 we discusses our results.

2. Metadata Repository and Quality Model

This section summarizes the nature of metadata used in the DWQ framework and gives an overview of the DWQ quality model. The section parlicularly focuses on the quality dimensions and quality factors associated to the main DW meta objects. As an example, the refreshment process is taken with its corresponding quality factors, because it is an crucial process for the quality of a DW.

2.1 Architecture Components

In the DWQ project we have advocated the need for enriched metadata facilities for the exploitation of the knowledge collected in a DW. In [6], it is shown that the DW metadata should track both architecture components and quality factors.

The proposed categorization of the DW metadata is based on a 3x3 framework, depicted in Figure 2: we identified three *perspectives* (conceptual, logical and physical) and three *levels* (source, DW, client). We made the observation, that the conceptual perspective, which represents the real world of an enterprise, is missing in DW projects, with the risk of incorrectly representing, or interpreting the information found in the DW.

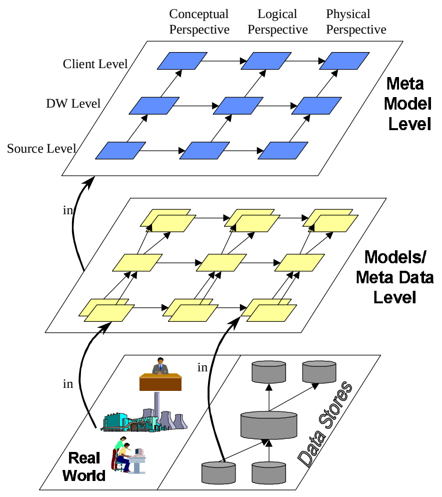


Fig. 2. The Data Warehouse Architecture Meta-Model

The proposed *metamodel* (i.e. the topmost layer in Figure 2) provides a notation for DW generic entities, such as schema, or agent, including the business perspective. Each box shown in figure 2 is decomposed into more detailed DW objects in the metamodel of [6]. This metamodel is instantiated with the *metadata* of the DW (i.e. the second layer in Figure 2), e.g. relational schema definitions or the description of the conceptual DW model. The lowest layer in Figure 2 represents the real world where the actual processes and data reside: in this level the metadata are instantiated with data instances, e.g. the tuples of a relation or the objects of the real world which are represented by the entities of the conceptual model.

2.2 Quality metamodel

Each object in the three levels and perspectives of the architectural framework can be subject to quality measurement. Since quality management plays an important role in DWs, we have incorporated it in our meta-modeling approach. Thus, the quality model is part of the metadata repository, and quality information is explicitly linked with architectural objects. This way, stakeholders can represent their quality goals explicitly in the metadata repository, while, at the same time, the relationship between the measurable architecture objects and the quality values is retained.

The DWQ quality metamodel [7] is based on the Goal-Question-Metric approach (GQM) of [10] originally developed for software quality management. In GQM, the high-level user requirements are modeled as goals. Quality metrics are values which express some measured property of the object. The relationship between goals and metrics is established through quality questions.

The main difference in our approach resides in the following points: (i) a clear distinction between subjective quality goals requested by stakeholder and objective quality factors attached to DW objects, (ii) quality goal resolution is based on evaluation of the composing quality factors, each corresponding to a given quality question, (iii) quality questions are implemented and executed as quality queries on the semantically rich metadata repository.

Figure 3 shows a simplified conceptual view of the DWQ Quality Model. The class ‘DW object type’ refers to any meta-object of the DWQ framework depicted in the first layer of figure 2. A *quality goal* is an abstract requirement, defined on DW object types, and documented by a purpose and the stakeholder interested in. This roughly expresses natural language requirements like ‘improve the availability of source s1 until the end of the month in the viewpoint of the DW administrator’. *Quality dimensions* (e.g. ‘availability’) are used to classify quality goals and factors into different categories. Furthermore, quality dimensions are used as a vocabulary to define quality factors and goals; yet each stakeholder might have a different vocabulary and different preferences in the quality dimensions. Moreover, a quality goal is *operationally* defined by a set of *questions* to which *quality factor* values are provided as possible answers. As a result of the goal evaluation process, a set of improvements (e.g. design decisions) can be proposed, in order to achieve the expected quality. A quality factor represents an actual measurement of a quality value, i.e. it relates quality values to measurable objects. A quality factor is a special property or characteristic of the related object wrt. the quality dimension of the quality factor. It also represents the expected range of the quality value, which may be any subset of the quality domain. Dependencies between quality factors are also stored in the repository.

The quality meta-model is not instantiated directly with concrete quality factors and goals, it is instantiated with patterns for quality factors and goals. The use of this intermediate instantiation level enables DW stakeholders to define templates of quality goals and factors. For example, suppose that the analysis phase of a DW project has detected that the availability of the source database is critical to ensure that the daily online transaction processing is not affected by the loading process of the DW. A source administrator might later instantiate this template of a quality goal with the expected availability of his specific source database. Thus, the programmers of the DW loading programs know the time window of the update process.

Based on the meta-model of DW architecture, we have developed a set of quality factor templates which can be used as a initial set for DW quality management. The exhaustive list of these templates can be found in [12]. The following section gives an intuition of some of them which are associated to the DW refreshment process.

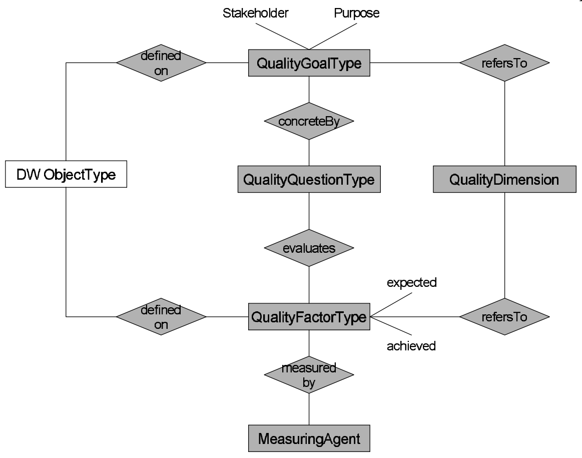


Fig. 3. The DWQ Quality Metamodel (simplified)

2.3 Quality metamodel instantiation: the refreshment case

As shown in [6], it is not sufficient to describe a DW as layers of materialized views on top of each other. For example, a view definition is not sufficient to capture the semantics of the refreshment process. Indeed, a view definition does not include the information whether this view operates on a history or not, how this history is sampled, which transformations are necessary during the refreshment, whether the changes of a given source should be integrated each hour or each week, and which data timestamp should be taken when integrating changes of different sources. Consequently, based on the same view definitions, a refreshment process may produce different results depending on all these extra parameters which have to be fixed independently, outside the queries which define the views. A detailed description for the refreshment process can be found in [4].

As mentioned before, the refreshment process is one of the main DWs processes for which the quality is an important issue. The associated quality template includes quality dimensions such as coherence, completeness and freshness.

- *Data coherence*: the respect of (explicit or implicit) integrity constraints from the data. For example, the conversion of values to the same measurement unit allows also to do coherent computations.
- *Data completeness*: the percentage of data found in a data store, with respect to the necessary amount of data that should rely there.
- *Data freshness*: the age of data (with respect to the real world values, or the date when the data entry was performed).

Quality Dim.	**DW objects**	**Primary Quality Factors**	**Derived Quality Factors**	**Design Choices**
Coherence	• Sources • ODS • Views	• Availability window of each source • Expected response time for a given query	• Extraction frequency of each source • Estimated response time of extraction for each source	• Granularity of data • Extraction and cleaning policy • Integration policy
Complete-ness	• Sources • ODS	• Availability window of each source • History duration for each DW store	• Extraction frequency of each source	• Extraction policy • Integration policy
Freshness	• Sources • ODS • Views	• Availability window of each source • Expected freshness for a given query • Estimated response time of extraction for each source, of integration and of propagation • Volume of data extracted and integrated	• Extraction frequency of each source • Actual freshness for a given query • Actual response time for a given query	• Extraction policy • Integration policy • Update policy

Table 1. Different levels of abstraction for the management of quality in a data.

Given a quality dimension, several low level quality factors of this dimension may be defined in a DW. For example, one can define quality factors like the *availability window* or the *extraction frequency* of a source, the *estimated values for the response time* of an algorithm or the *volume of the data extracted each time*, etc. However, the quality factors are not necessarily independent of each other, e.g., completeness and coherence may induce a certain accuracy of data. We discriminate between *primary* and *derived* quality factors as well as *design choices*. A primary quality factor is a simple estimation of a stakeholder or a direct measurement. For example, the completeness of a source content may be defined with respect to the real world this source is supposed to represent. Hence, the completeness of this source is a subjective value directly assigned by the DW administrator or the source administrator. On the other hand, derived quality factors are computed as formulae over some other quality factors: for example, the completeness of the operational data store content can be defined as a formula over the completeness of the sources. The design choices are a special kind of quality factors, expressed as parameter values and control strategies which aim to regulate or tune the algorithm followed for the performance of each task in the DW.

We believe that quality dimensions, quality factors and design choices are tightly related. For example, in the case of the refreshment process, the design choice 'extraction policy' is related to the derived quality factor 'extraction frequency' of each source, which is computed from the corresponding primary quality factor 'availability window' of each source. In table 1, we mention several quality factors which are relevant to the refreshment process and link them to the corresponding DW objects and quality dimensions. One can also notice that some quality factors may belong to more than one dimension. Some of them are primary quality factors, arbitrarily assigned by the DW administrator, others are derived. Deriving procedures can be either mathematical functions, logical inferences or any *ad hoc* algorithms. The values of derived quality factors depend on design choices which can evolve with the semantics of the refreshment process. Underlying the design choices are design techniques, that are all rules, events, optimizations and algorithms which implement the strategies on which refreshment activities are based.

3. Exploitation of the Metadata Repository and the Quality Metamodel

This section describes the DWQ methodology to exploit the quality metamodel on the basis of the meta-data repository. We first present the different phases and steps of our methodology.

Our approach extends GQM, based on the idea that a goal is operationally defined over a set of questions. Thus, we provide specific "questions" for the full lifecycle of a goal: this way the DW metadata repository is not simply defined statically, but it can be actually exploited in a systematic manner. Underlying our methodology, we exploit:

Our approach for quality management is based on two folds:

- A metadata repository which provides all the necessary knowledge to understand quality goals, quality factors and their related DW objects. This repository allows to trace design decisions, and to report on the history of quality goals with their successive evaluations and improvements.
- A computational engine composed of all the deriving procedures of quality factors. The techniques underlying this engine can be simple functions and procedures or more sophisticated reasoning mechanisms. In the case of performance evaluation of a given query, a mathematical function is generally sufficient while in the case of coherence validation of a conceptual schema we need a more sophisticated inference mechanism.

Based on this, the DWQ methodology for quality management is composed of three main phases: (i) the design phase which elaborates a quality goal by defining its purpose, the set of questions to solve it and the set of quality factors which answer to these questions; (ii) an evaluation phase which deals with the computation of quality factors; (iii) an analysis and improvement phase which gives an interpretation to the quality goal evaluation and suggests a set of improving actions.

In Figure 4, we graphically present our methodological approach for quality management. This methodology is influenced by the TQM paradigm, which has also been adopted by other approaches such as TDQM [14]. The DWQ methodology suggests four major *steps* (or else *phases of a lifecycle*) when dealing with a quality goal: *design, current status evaluation, analysis and improvement* and *re-evaluation* of a quality goal. In the sequel we provide a detailed presentation of a set of questions corresponding to each step of each process. Before proceeding, we would like to mention that the proposed methodology does not consist of a strict algorithm: one may choose to ignore several steps, according to the specific situation he is tackling.

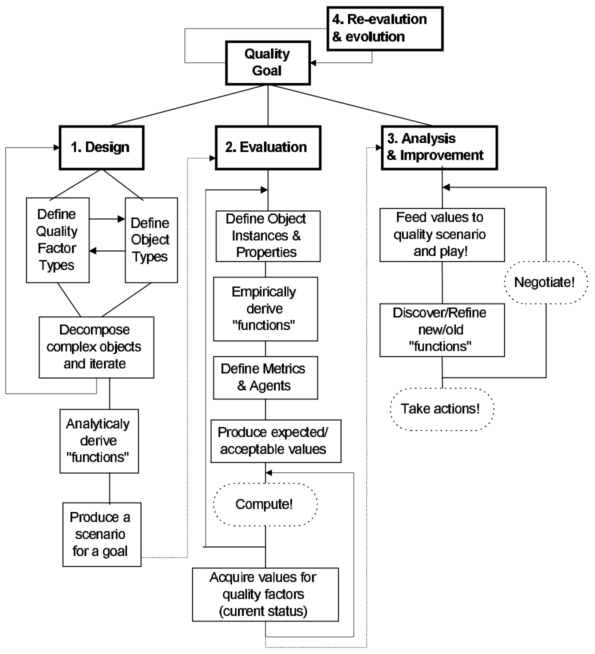

4. Re-evalution & evolution
Quality Goal
1. Design
2. Evaluation
3. Analysis & Improvement
Define Quality Factor Types
Define Object Types
Decompose complex objects and iterate
Analyticaly derive "functions"
Produce a scenario for a goal
Define Object Instances & Properties
Empirically derive "functions"
Define Metrics & Agents
Produce expected/ acceptable values
Compute!
Acquire values for quality factors (current status)
Feed values to quality scenario and play!
Negotiate!
Discover/Refine new/old "functions"
Take actions!

Fig. 4. DW Quality management

3.1 The Design Phase

When dealing with a quality goal, we assume that there is always a first time when the *stakeholder* defines the goal. The *design process* of the goal is the first phase of its interaction with the stakeholder and should result in the selection of the involved object types and their quality factors.

There are two steps which can take place at the same time: *the identification of the object types which are related to the goal* and the respective low level *quality factors.* The identification of the object types, tries to reuse the experience stored in the metadata repository: the metamodel is powerful enough to model the relationships not only at the instance but at the type level as well.

Nevertheless, these steps alone are not sufficient to characterize a quality goal. Since the identified object types are most probably composite (e.g. a schema is composed from several relations) one has to *decompose them at a satisfactory level of detail*. For example, if the examined type is the *refreshment process*, one can try to decompose it into more refined objects such as *data extraction, data cleaning and transformation, data integration and high level aggregation*.

The next step deals with *identification of the inter-relationships between objects and quality factors*. Each object can be viewed as a node of graph. Every node in the graph has input and output arcs, determining the interdependencies of the DW components with respect to their quality factors. Several design choices are by default encapsulated in the figure (e.g. the simple fact that the data of a materialized view stem from source data). The input arc to the graph is the high level quality dimension expressed by the user. The output is the set of specific quality factors which measure this quality dimension.

The goal of this process is, not only to set up a list of the "ingredients" of the problem, but also, to come up with a list of "functions", determining the outcome of the quality of an object, in terms both of its own characteristics and of the quality of other objects affecting it. We call the outcome of the process, the *scenario* of the quality goal.

More specifically, to produce the list of functions, one has to try and analytically define the inter-relationships by inspecting the peculiarities of the problem. For example, one could use either a Pareto diagram [3] to determine the timeliness of a materialized view, or a function taking into respect the availability of the sources, the frequency of updates and queries and the capacity of the propagators.

We do not advocate that these functions can always be derived or discovered in an analytic form: yet (a) as we will show there are truly cases where this can happen, (b) in the absence of an analytical form we can use empirical knowledge and (c) it is important to note that even the existence of an interdependency link can be used as a boolean function to denote dependency.

Example. In the example of Figure 5, we try to quantify a quality dimension: the *believability* of the information delivered to the final user. To achieve this goal, we decide that we have to measure a specific quality factor: the *accuracy* of the data in the views used by the final users. The scenario is composed from all the components participating in the refreshment of a view: the *source database* (which in terms is decomposed to a set of *source relations*), the *transformation agents* converting the data to the desired format and the *DW/ODS views*, each one possibly defined on top of another view. We also provide an analytical function for the accuracy of a view, calculating it from the size and the accuracy of the input data.

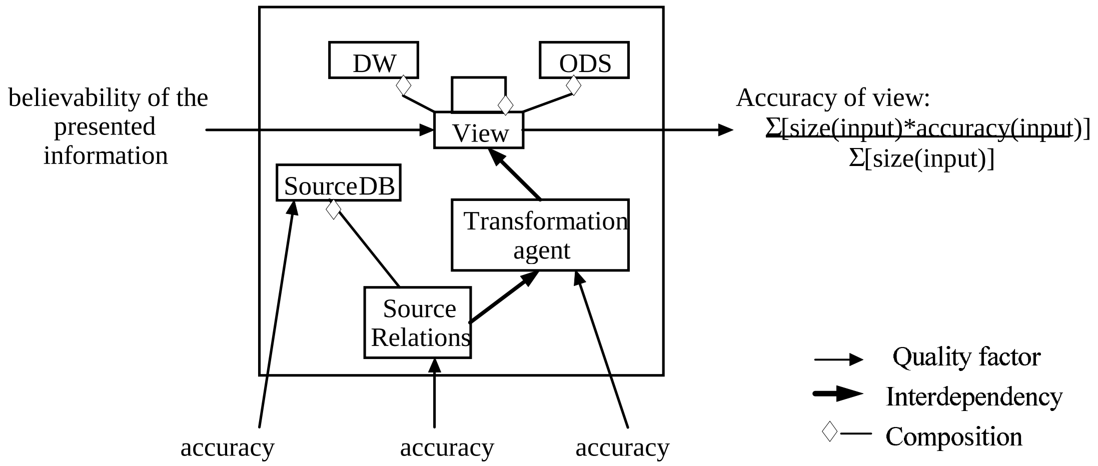


Fig. 5. The scenario of a quality goal

A fascinating feature of a scenario is the tracking of the *inverse relationships* between the quality factors. In other words, by describing the interdependencies of the quality factors, not only do we get a clear view of the way the overall quality of our final "product" is influenced, but also we get a first insight of how to remedy an undesired situation. For example, in Figure 5, we can improve the believability of our information by increasing its accuracy, something which in terms can be achieved through the improvement of the accuracy of the transformation agents or the source relations. In the case of redundant information, one can also increase the volume of the utilized data from a source with higher accuracy. In any case, to generalize this observation, the inverse functions can be either analytical relationships or inverse interdependency path on the scenario.

3.2 The Evaluation Phase

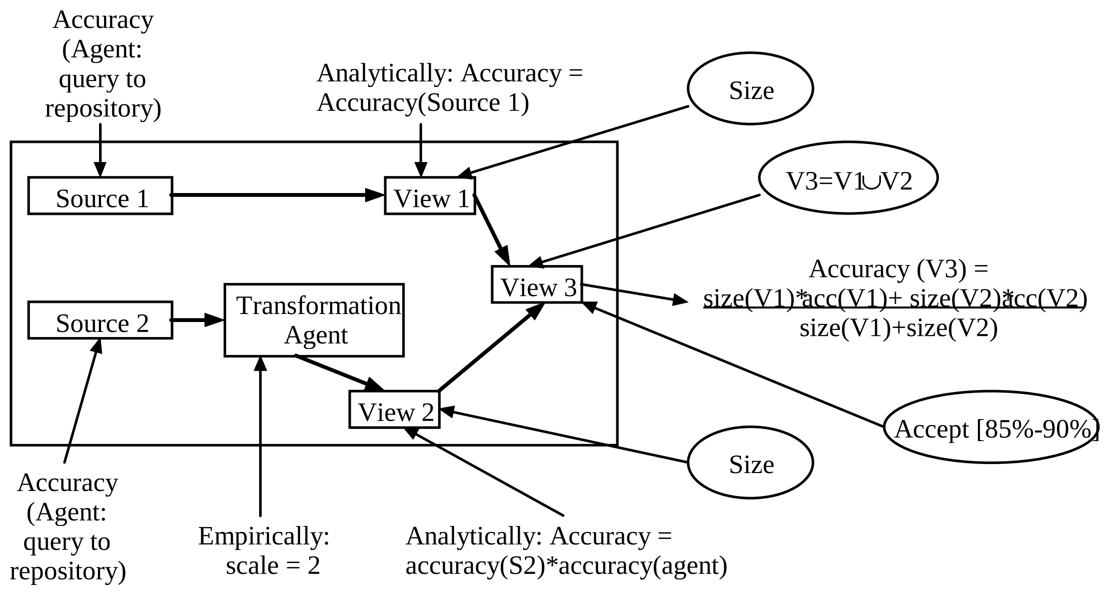
After the design process, the following step is the *evaluation of the current status*. The purpose of the evaluation phase is to construct a detailed *map* based on the constructed scenario, which describes accurately all the interplaying components and factors. This can also be the first step, when a goal has already been defined in the past and a scenario has been developed and is currently reused.

First, we must *determine the specific object instances* of the specific evaluation through a query to the metadata repository. In the example of Figure 5, one can identify two source relations (S_1, S_2), pumping data to two views in the ODS (V_1, V_2), through a respective transformation agent and a final view (V_3), the accuracy of which we have to quantify (Figure 6).

Next, one must take into account several *design choices*, i.e. the properties of the interplaying objects which influence the quality of the outcome. In our example, one can take into account the size of the propagated data, the time windows of the sources, the regularity of the refreshment, etc. For reasons of simplicity of the presentation and since we deal only with the accuracy factor, we retain only the size of the propagated data and the view definitions.

Apart from the component refinement, we can also refine the interrelationships between the quality factors. The refinement can be performed either through the use of analytical formulae or direct instantiations in the scenario, based on the empirical knowledge of a specific situation. Empirical knowledge can be obtained from simple observation or through the use of well-tested techniques such as statistical process control (SPC), concurrent engineering, etc. [3]. In our example, simple sampling could show that in the past, the transformation agent increased the accuracy of the data by 2.

Then, for each quality factor one should also determine the *metrics* and *measuring agents*. If no measuring agent(s) has ever been defined, one must determine the computation procedure for the actual values of the quality factors. Also, the parameters of the measuring procedures should be set accordingly.



Accuracy (Agent: query to repository)
Analytically: Accuracy = Accuracy(Source 1)
Size
V3=V1∪V2
Source 1
View 1
View 3
Accuracy (V3) = size(V1)*acc(V1)+ size(V2)*acc(V2) / size(V1)+size(V2)
Source 2
Transformation Agent
View 2
Accept [85%-90%]
Size
Accuracy (Agent: query to repository)
Empirically: scale = 2
Analytically: Accuracy = accuracy(S2)*accuracy(agent)

Fig. 6. The map of a quality goal

The final step is the addition of *acceptable/expected values for each quality factor*, wherever necessary. This is a crucial step for the evaluation of the current status later on. The accepted range of values will be the basic criterion for the objective judgment of a subjective quality goal. The outcome of this step should provide the stakeholder with a well defined *map* of the problem (see also Figure 6).

With respect to the scenario of Figure 5, the map is enriched with (a) agents for the computation of primary quality factors (e.g. the queries at the metadata repository), (b) formulae for the computation of the derived quality factors, (c) properties of the components such as the view definition or the size of the propagated data and (d) acceptable ranges of values (e.g. accuracy of view 3).

After that, the only thing left is the acquisition/calculation of the specific values of the selected quality factors, though the necessary computation. In Figure 7, a certain instance of our exemplary map is depicted.

The acquisition of these values is performed through the use of the already defined measuring agents. In fact, we anticipate that if the values are regularly (i.e. not on-demand) computed and stored in the metadata repository, then their acquisition can be done through a simple query to the metadata repository.

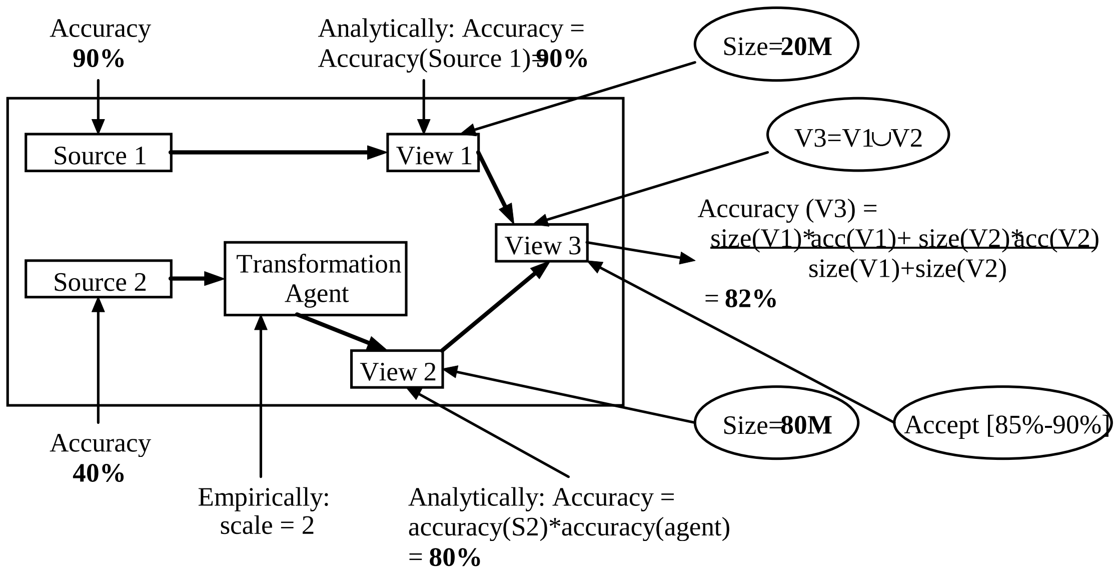


Fig. 7. Instance of a quality map

Here we must clarify again, that several steps can be omitted. In fact, if we consider that the metadata repository is regularly refreshed through an external agent, then some of the intermediate steps of this process can be avoided.

3.3 The Analysis and Improvement Phase

At the end of the second phase, the map of Figure 7 is fully instantiated with actual values. Yet, if the situation is not obviously satisfactory, the stakeholder may choose to react against it. Although this is a process with different characteristics each time, we can still draw some basic guidelines for the steps that can be taken. Consider for example the case in Figure 7, where the computed accuracy for view 3 is not within the accepted range. Obviously there must be some reaction against this undesired situation.

One of the main advantages of our approach is that if we have an understanding of the mechanism that produces the problem, we can attack the problem directly through the use of the inverse quality functions, which have been derived during the design phase or detected during the evaluation phase. Again, by 'inverse functions' we mean both the possible analytical functions and the inverse interrelationships in the map of the problem.

The inverse functions in our example suggest that an increase of 10% for the accuracy of view 3 calls for one of the following actions:

a) Use the analytical formulae directly: increase of 10% to the accuracy of views 1 and 2 (directly through the formula), which in terms implies:

- increase of the accuracy of source 1 by 10%;
- increase of the accuracy of source 2 by 5% or the accuracy of the agent by 10% or a combination of the two.

b) Customize the reaction to the specific characteristics of the situation: Through the use of the specific measurements one could also try to derive a plan taking into account the sizes of the input views. For example, elementary calculations prove that it suffices to increase the accuracy of source 2 to 45%, for the quality of the view 3 to be in the accepted range.

Nevertheless, there is always the possibility that this kind of approach is not directly feasible. If our understanding of the problem is not full, then steps must be taken so that we deepen our knowledge. Moreover, it is possible that the derived solution is not feasible -or is too costly to achieve.

In the first case we must go all the way back to the design process and try to *refine the steps of the function discovery*. In the second case we must try to use the inverse functions in order to *determine which are the feasible limits of values* which we can negotiate. The negotiation process is a painful task, since one has to deal with contradictory goals and priorities. Yet, several specific techniques exist which can be applied to the negotiation problem. In section 5, we present as example the QFD and the Statistical Process Control methodologies. Other examples are the experimental design, the Taguchi quality engineering etc. [3].

4 The Data Warehouse Evolution

A DW is a very complex system whose components evolve frequently independently of each other. Users can create new views or update old ones. Some sources may disappear while others are added. The enterprise model can evolve with the enterprise objectives and strategies. The technical environment changes with products evolution and updates. Design choices at the implementation level can also evolve to achieve users requirements and administration requirements.

In this evolving context, the re-evaluation of a goal and of the strategy to achieve is a strict contingency in a DW environment. There are 3 main reasons for this:

(a) *evolution reasons*: there are natural changes happening in such a complex environment;

(b) *failure in the achievement of the desired quality*, and

(c) *meta-quality:* we can never be sure for the quality of our measuring processes.

In the sequel we will detail the first of these reasons. DWs can evolve in many different ways. The *data stores* can produce changes due to reasons of schema evolution in logical and conceptual perspective, changes to the physical properties of the source (e.g. location, performance etc.), insertions or deletions of a data store, specific reasons due to their nature (e.g. in the sources, the time window for extraction or the data entry process can change). The *software components* can be upgraded, completed, debugged, etc. The *propagation agents* of all types (loaders/refreshers/ wrappers/mediators/source integrators) can obtain new schedules, new algorithms, rules, physical properties, etc. Needless to say that the *user requirements* continuously change, too. New requirements arise, while old ones may become obsolete, new users can be added, priorities and expected/acceptable values change through the time, etc. Moreover, the *business rules* of an organization are never the same, due to real world changes.

As a result of evolution and errors, our goals, components, scenarios and maps are never to be fully trusted. Each time we reuse previous results we must always consider cases like: lack of measurement of several objects, errors in the measurement procedure (e.g. through an agent which is not appropriate), outdated information of the repository with respect to the DW, etc.

A way to control this evolution is to provide a complementary meta-data which tracks the history of changes and provides a set of consistency rules to enforce when a quality factor has to be re-evaluated. To do so, it is necessary to link quality factors to evolution operators which affect them. The idea behind this is to enrich the meta-data repository in order to ease the impact analysis of each evolution operator and its consequences on the quality factor measures.

5. Related Work

There has ben much research on the definition and measurement of data quality dimensions [18,15,17,13]. A very good review is found in [16]. The GQM methodology is best presented in [10,2].

The *TDQM* methodology [14] follows the Total Quality Management approach, adapted for the evaluation of data quality in an information system (by assuming that each piece of produced information can be considered a product). The TDQM methodology also follows the TQM cycle: *Definition, Measurement, Analysis, Improvement*. The Definition part identifies the important quality dimensions and the corresponding requirements. The Measurements step produces the quality metrics. The Analysis step identifies the roots of any quality problems and their interplay, while the Improvement step provides techniques for improving the quality of information.

Negotiation techniques enable the negotiation over the desired quality of a system. *Statistical Process Control* (SPC) is one of the best tools for monitoring and improving product and service quality [3]. SPC comprises of several techniques, such as Pareto diagrams (used to identify the most important factors of a process), process flow diagrams, cause and effect (or Ishikawa) diagrams, check sheets, histograms and control charts.

Quality Function Deployment, (QFD) [5,3] is a team-based management technique, used to map customer requirements to specific technical solutions. This philosophy is based on the idea that the customer expectations should drive the development process of a product. The basic tool used in QFD is the so-called "House of Quality", mapping user expectations to technical solutions, taking into account priorities and conflicts. However, while QFD certainly has a useful role in rough quality planning and cross-criteria decision making, using it alone would throw away the richness of work created by research in measuring, predicting, or optimizing individual DW quality factors.

6. Conclusions

In this paper, we deal with the problem of quality-oriented design, usage and evolution of DWs. As explained also in previous papers we store semantically rich meta information of a DW in our DWQ repository concerning the conceptual, logical and physical perspective of a DW. In addition, the information on the quality of the stored objects is recorded in the repository.

In our view, the quality of the DW is an aggregated view of the metadata and data of the warehouse. For example, the quality of the DW depends on the quality of the sources, the quality of the extraction process and the quality of the DW components itself. One can think of the quality factors as materialized views over the metadata and data of the warehouse; thus the evolution of the DW can be seen as a view maintenance problem on the aggregated quality views. For example, if a source is changed -- either its data or its properties (metadata) -- the quality of the source must be recomputed. All objects and their quality which depends on this source must be adapted to the new situation.

The application of our GQM-like methodology helps us to (a) design and (b) maintain the knowledge about this evolution efficiently. We make extensive use of our metadata repository, so that the information is obtained in a controlled, efficient fashion. We have elaborated on our quality metamodel, in order to track the basic primitives of the interrelationships between DW components and quality factors. We also extend GQM so that we can both take advantage of the interrelationships of components and track the full lifecycle of a stakeholders' requirement.

We exploit the dependencies and computation functions of quality factors. For example, in the case a new source is integrated, we just have to compute the quality of this source and recompute the DW and data mart quality using the information of the process quality, which describes how the data is transformed and what improvement or debasement to the data quality has been made.

In addition to the maintenance process of the quality, the inverse of the computation functions for the quality factors can be used, to find the DW object which has to be improved to reach a certain quality goal. This process can be compared with the view update process in databases systems, where updates to views (here: quality views) are translated to updates in base data (here: quality factors).

As of now, our metamodel and methodology have been validated only partially. As future work, we plan to fully validate and refine them through the development of software tools and application to major case studies.

Acknowledgments. This research is sponsored by the European Esprit Project "DWQ: Foundations of Data Warehouse Quality", No. 22469. We would like to thank all our DWQ partners who contributed to the progress of this work, and especially Prof. Matthias Jake, Manfred Jeusfeld and Maurizio Lenzerini. Many thanks are also due to the anonymous reviewers, for useful comments.

References

1. P.A. Bernstein, Th. Bergstraesser, J. Carlson, S. Pal, P. Sanders, D. Shutt. Microsoft Repository Version 2 and the Open Information Model. To appear in *Information Systems* 24(2), 1999.
2. V. R. Basili, G.Caldiera, H. D. Rombach. The Goal Question Metric Approach. Encyclopedia of Software Engineering - 2 Volume Set, pp 528-532, John Wiley & Sons, Inc., available at http://www.cs.umd.edu/users/basili/papers.html, 1994
3. D. H. Besterfield, C. Besterfield-Michna, G. Besterfield, M. Besterfield-Sacre, Total Quality Management, Prentice Hall, 1995
4. M. Bouzeghoub, F. Fabret, M. Matulovic, E. Simon. Data Warehouse Refreshment: A Design Perspective from Quality Requirements. Technical Report D 8.5, DWQ Consortium, 1998.
5. E. B. Dean, "Quality Functional Deployment from the Perspective of Competitive Advantage", available at http://mijuno.larc.nasa.gov/dfc/qfd.html, 1997
6. M. Jarke, M.A.Jeusfeld, C. Quix, P. Vassiliadis: Architecture and quality in data warehouses, Proceedings CaiSE 98, Pisa, Italy, 1998.
7. M.A. Jeusfeld, C. Quix, M. Jarke: Design and Analysis of Quality Information for Data Warehouses. In Proc. of the 17th International Conference on the Entity Relationship Approach (ER'98), Singapore, 1998.
8. M. Jarke, Y. Vassiliou. Foundations of data warehouse quality – a review of the DWQ project. In *Proc. 2nd Intl. Conference Information Quality (IQ-97)*, Cambridge, Mass., 1997.
9. Metadata Coalition: Meta Data Interchange Specification, (MDIS Version 1.1), August 1997, available at http://www.he.net/~metadata/standards/ .
10. M. Oivo, V. Basili: Representing software engineering models: the TAME goal-oriented approach. IEEE Transactions on Software Engineering, 18, 10, 1992.
11. K. Orr. Data quality and systems theory. In Communications of the ACM, 41, 2, Feb. 1998.
12. C. Quix, M. Jarke, M. Jeusfeld, M. Bouzeghoub, D. Calvanese, E. Franconi, M. Lenzerini, U. Sattler, P. Vassiliadis. Quality Oriented Data Warehouse Evolution. Technical Report D9.1, DWQ Consortium, 1998.
13. G. K. Tayi, D. P. Ballou: Examining Data Quality. In Communications of the ACM, 41, 2, Feb. 1998.
14. R. Y. Wang. A product perspective on total data quality management. In Communications of the ACM, 41, 2, Feb. 1998.
15. R.Y. Wang, H.B. Kon, S.E. Madnick. Data Quality Requirements Analysis and Modeling. In Proc. of 9 th International Conference on Data Engineering, pp. 670-677, IEEE Computer Society, Vienna, Austria, 1993.
16. R.Y. Wang, V.C. Storey, C.P. Firth. A Framework for Analysis of Data Quality Research. IEEE Transactions on Knowledge and Data Engineering, Vol. 7, No. 4, August 1995.
17. R.Y. Wang, D. Strong, L.M. Guarascio. Beyond Accuracy: What Data Quality Means to Data Consumers. Technical Report TDQM-94-10, Total Data Quality Management Research Program, MIT Sloan School of Management, Cambridge, Mass., 1994.
18. Y. Wand, R.Y. Wang. Anchoring Data Quality Dimensions in Ontological Foundations. Communications of the ACM, Vol. 39, No. 11, November 1996.