From: CAiSE 1998, LNCS 1413 © Springer-Verlag Berlin Heidelberg 1998

Architecture and Quality in Data Warehouses[1]

Matthias Jarke[(1)], Manfred A. Jeusfeld[(2)]
Christoph Quix[(1)], Panos Vassiliadis[(3)]

(1) RWTH Aachen, Germany, {jarke,quix}@informatik.rwth--aachen.de

(2) Tilburg University, The Netherlands, jeusfeld@kub.nl

(3) National Technical University of Athens, Greece, pvassil@dbnet.ece.ntua.gr

Abstract. Most database researchers have studied data warehouses (DW) in their role as buffers of materialized views, mediating between update-intensive OLTP systems and query-intensive decision support. This neglects the organizational role of data warehousing as a means of centralized information flow control. As a consequence, a large number of quality aspects relevant for data warehousing cannot be expressed with the current DW meta models. This paper makes two contributions towards solving these problems. Firstly, we enrich the meta data about DW architectures by explicit enterprise models. Secondly, many very different mathematical techniques for measuring or optimizing certain aspects of DW quality are being developed. We adapt the Goal-Question-Metric approach from software quality management to a meta data management environment in order to link these special techniques to a generic conceptual framework of DW quality. Initial feedback from ongoing experiments with a partial implementation of the resulting meta data structure in three industrial case studies provides a partial validation of the approach.

1 Introduction

Data warehouses provide large-scale caches of historic data. They sit between information sources gained externally or through online transaction processing systems (OLTP), and decision support or data mining queries following the vision of

[1] This research was partially supported by the European Commission in ESPRIT Long Term Research Project DWQ (Foundations of Data Warehouse Quality), by the General Secretariat of Research and Technology (Greece) under the PENED program; and by the Deutsche Forschungsgemeinschaft through Graduiertenkolleg "Informatik und Technik".

online analytic processing (OLAP). Three main arguments have been put forward in favor of this caching approach:

1. *Performance and safety considerations*: The concurrency control methods of most DBMSs do not react well to a mix of short update transactions (as in OLTP) and OLAP queries that typically search a large portion of the database. Moreover, the OLTP systems are often critical for the operation of the organization and must not be under danger of corruption of other applications.
2. *Logical interpretability problems*: Inspired by the success of spreadsheet techniques, OLAP users tend to think in terms of highly structured multi-dimensional data models, whereas information sources offer at best relational, often just semi-structured data models.
3. *Temporal and granularity mismatch*: OLTP systems focus on current operational support in great detail, whereas OLAP often considers historical developments at a somewhat less detailed granularity.

Thus, quality considerations have accompanied data warehouse research from the beginning. A large body of literature has evolved over the past few years in addressing the problems introduced by the DW approach, such as the trade-off between freshness of DW data and disturbance of OLTP work during data extraction; the minimization of data transfer through incremental view maintenance; and a theory of computation with multi-dimensional data models.

However, the heavy use of highly qualified consultants in data warehouse applications indicates that we are far from a systematic understanding and usage of the interplay between quality factors and design options in data warehousing. The goal of the European DWQ project [JV97] is to address these issues by developing, prototyping and evaluating comprehensive Foundations for Data Warehouse Quality, delivered through *enriched meta data management facilities* in which specific analysis and optimization techniques are embedded.

This paper develops the DWQ architecture and quality management framework and describes first steps towards its implementation and validation. The main contributions include an extension of the standard DW architecture used in the literature by enterprise modeling aspects, and a strategy for embedding special-purpose mathematical reasoning tools in the architecture which will enable a computationally tractable yet rich quality analysis or quality-driven design process.

Interaction with DW tool vendors, DW application developers and administrators has shown that the standard framework used in the DW literature is insufficient to capture in particular the business role of data warehousing. A DW is a major investment made

to satisfy some business goal of the enterprise; quality model and DW design should reflect this business goal as well as its subsequent evolution over time. In section 2, we discuss this problem in detail; our new architectural framework separates (and links) explicitly the concerns of conceptual enterprise perspectives, logical data modeling (the main emphasis of DW research to date), and physical information flow (the main concern of commercial DW products to date).

In section 3, we first build on literature frameworks for data and software quality to come up with a suitable set of DW quality dimensions, as perceived by different groups of stakeholders. We then adapt a variant of the so-called Goal-Question-Metric approach used in software quality management. Through materialized quality views, we link conceptual quality goals to specific analysis techniques developed in DW research and practice, and enable trade-offs between heterogeneous quality goals. Initial experiences with a prototypical implementation of the resulting meta database using the ConceptBase deductive object manager have been gained in cooperation with industrial case studies. Section 4 relates our approach to other work in data warehousing, data and software quality, while section 5 provides a summary and conclusions.

2 The Architecture of a Data Warehouse

Physically, a data warehouse system consists of databases (source databases, materialized views in the distributed data warehouse), data transport agents that ship data from one database to another and a data warehouse repository which stores all kinds of meta data about the system. The content of the repository determines to a large extent how the data warehouse system can be used and evolved. The main goal of our approach is therefore to define a meta database schema which can capture and link all relevant aspects of DW architecture and quality. We shall tackle this very difficult undertaking in several steps.

2.1 Three Perspectives of Data Warehouse Meta Data

Almost all current research and practice understand a data warehouse architecture as a stepwise information flow from information sources through materialized views towards analyst clients, as shown in figure 2.1. Our key observation is that this architecture covers only partially the tasks faced in data warehousing and is therefore unable to even express, let alone support, a large number of important quality problems and management strategies.

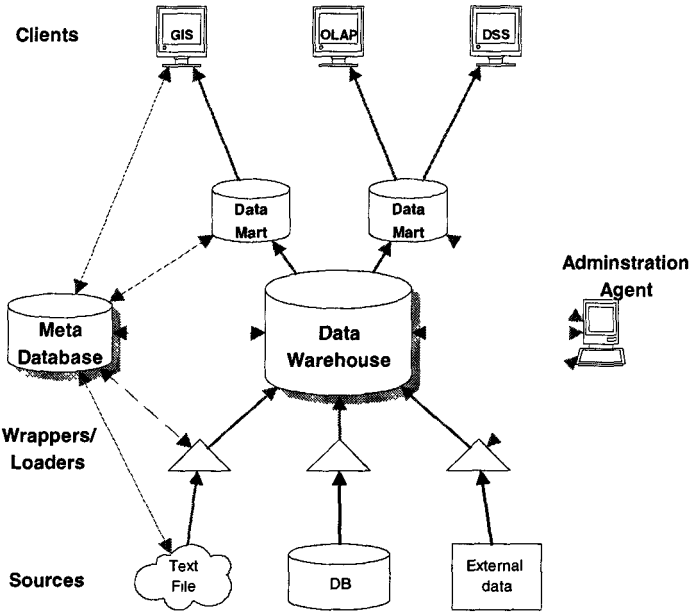

Figure 2.1: Current Understanding of a Data Warehouse

As a consequence, we propose a separation of three perspectives as shown in figure 2.2: a conceptual enterprise perspective, a logical data modeling perspective, and a physical data flow perspective.

The main argument we wish to make is the need for a *conceptual enterprise perspective*. To explain, consider the left two columns of figure 2.2. Suppose an analyst wants to know something about the business -- the question mark in the figure. She does not have the time to observe the business directly but must rely on existing information gained by operational departments, and documented as a side effect of OLTP systems. This way of information gathering implies already a bias which needs to be compensated when selecting OLTP data for uploading and cleaning into a DW where it is then further pre-processed and aggregated in data marts for certain analysis tasks. Considering the long path the data has taken, it is obvious that also the last step, the formulation of conceptually adequate queries and the conceptually adequate interpretation of the answers presents a major problem to the analyst.

The traditional DW literature only covers two of the five steps in figure 2.2. Thus, it has no answers to typical practitioner questions such as "how come my operational departments put so much money in their data quality, and still the quality of my DW is terrible?" (answer: the enterprise views of the operational departments are not easily compatible with each other or with the analysts view), or "what is the effort required

to analyze problem X for which the DW currently offers no information?" (could simply be a problem of wrong aggregation in the materialized views, could require access to not-yet-integrated OLTP sources, or even involve setting up new OLTP sensors in the organization).

An adequate answer to such questions requires an explicit model of the conceptual relationships between an enterprise model, the information captured by OLTP departments, and the OLAP clients whose task is the decision analysis. We have argued that a DW is a major investment undertaken for a particular business purpose. We therefore do not just introduce the enterprise model as a minor part of the environment, but demand that *all other models are defined as views on this enterprise model.* Perhaps surprisingly, even information source schemas define views on the enterprise model -- not vice versa as suggested by figure 2.1!

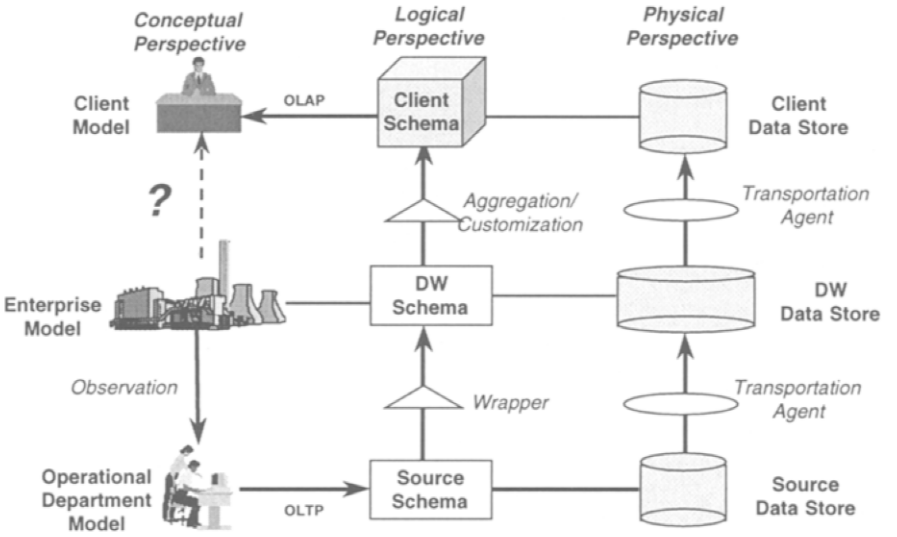

Figure 2.2 The Data Warehouse Meta Data Framework

The wrapping and aggregation transformations performed in the (traditionally discussed) logical perspective can thus be checked for interpretability, consistency or completeness with respect to the enterprise model -- provided an adequately powerful representation and reasoning mechanism is available. At the same time, the logical transformations need to be implemented safely and efficiently by physical storage and transportation -- the third perspective in our approach. It is clear that physical quality aspects require completely different modeling formalisms than the conceptual factors, typical techniques stemming from queuing theory and combinatorial optimization.

There is no single decidable formalism that could handle all of these aspects uniformly in a meta database. We have therefore decided to capture the architectural framework in a deductive object data model in a comprehensive but relatively shallow manner. Special-purpose reasoning mechanisms such as the ones mentioned above can be linked to the architectural framework as discussed in section 3, below.

2.2 A Notation for Data Warehouse Architecture

We use the meta database to store an abstract representation of data warehouse applications in terms of the three-perspective scheme. The architecture and quality models are represented in Telos [MBJK90], a metadata modeling language. Its implementation in the ConceptBase system [JGJ+95] provides query facilities, and definition of constraints and deductive rules. Telos is well suited because it allows to formalize specialized modeling notations by means of meta classes. Preloaded with these metaclasses, the ConceptBase system serves as the meta database for quality-oriented data warehouses.

A condensed graphical overview of the architecture notation is given in Figure 2.3. Bold arrows denote specialization links. The most general meta class is *DW_Object*. It subsumes objects at any perspective (conceptual, logical, or physical) and at any level (source, data warehouse, or client).

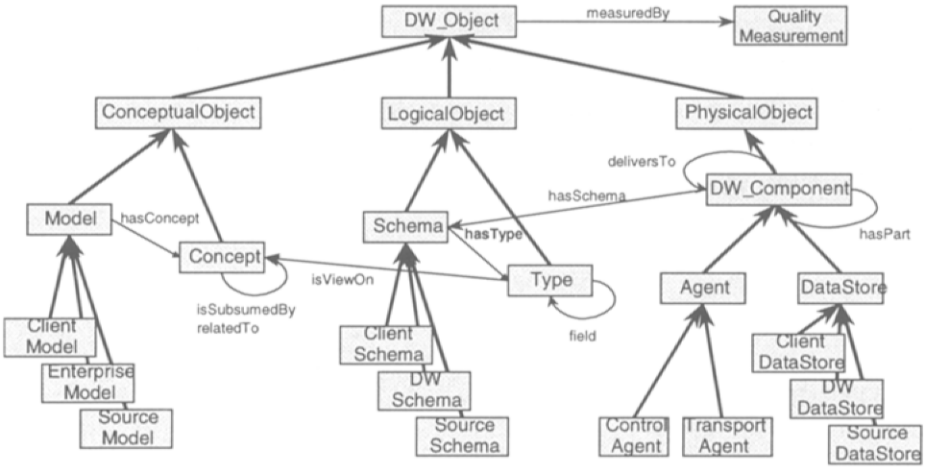

Figure 2.3: Overview of the Architecture Notation

Within each perspective, we distinguish between the modules it offers (e.g. client model) and the kinds of information found within these modules (e.g. concepts and their subsumption relationships). The horizontal links *hasSchema* and *isViewOn* establish the way how the horizontal links in Figure 2.2 are interpreted: the types of a

schema (i.e., relational or multidimensional structures) are defined as logical views on the concepts in the conceptual perspectives. On the other hand, the components of the physical perspective get a schema from the logical perspective as their schema.

Each object can have an associated set of materialized views called *QualityMeasurements*. These materialized views (which can also be specialized to the different perspectives -- not shown in the figure) constitute the bridge to the quality model discussed in section 3.

The horizontal levels of the objects are coded by the three subclasses attached to *Model*, *Schema*, and *DataStore*. We found this notation adequate to represent physical data warehouse architectures of commercial applications, such as the SourcePoint tool marketed by Software AG [SAG96] or the DW architecture underlying a data mining project at Swiss Life [SKR97]. The logical perspective currently supports relational schema definitions whereas the conceptual perspective supports the family of extended entity-relationship and similar semantic data modeling languages. Note that all objects in Figure 2.3 are meta classes: actual conceptual models, logical schemas, and data warehouse components are represented as instances of them in the meta database. In the following subsections, we elaborate on the purpose of representing each of the three perspectives.

2.3 Conceptual Perspective

The conceptual perspective describes the business models underlying the information systems of an enterprise. The central role is played by the enterprise model, which gives an integrative overview of the conceptual objects of an enterprise. The models of the client and source information systems are views on the enterprise model, i.e. their contents are described in terms of the enterprise model. One goal of the conceptual perspective is to have a model of the information independent from physical organization of the data, so that relationships between concepts can be analyzed by intelligent tools, e.g. to simplify the integration of the information sources. On the client side, the interests of user groups can also be described as views on the enterprise model.

In the implementation of the conceptual perspective in the meta database, the central class is *Model*. A model is related to a source, a client or the relevant section of the enterprise, and it represents the concepts which are available in the corresponding source, client or enterprise. The classes *ClientModel*, *SourceModel* and *EnterpriseModel* are needed, to distinguish the models of several sources, clients and the enterprise itself. A model consists of *Concepts*, each representing a concept of the

real world, i.e. the business world. If the user provides some information about the relationship between concepts in a formal language like description logics, a reasoner can check for subsumption of concepts [CDL97].

The results of the reasoning process are stored in the model as attribute *isSubsumedBy* of the corresponding concepts. Essentially, the repository can serve as a cache for reasoning results. Any tool can ask the repository for containment of concepts. If the result has already been computed, it can directly be answered by the repository. Otherwise, a reasoner is invoked by the repository to compute the result.

2.4 Logical Perspective

The logical perspective conceives a data warehouse from the view point of the actual data models involved, i.e. the data model of the logical schema is given by the corresponding physical component, which implements the logical schema. The central point in the logical perspective is *Schema.* As a model consists of concepts a schema consists of *Types.* We have implemented the relational model as an example for a logical data model; other data models such as the multi-dimensional or the object-oriented data model are also being integrated in this framework.

Like in the conceptual perspective, we distinguish in the logical perspective between *ClientSchema, DWSchema* and *SourceSchema* for the schemata of clients, the data warehouse and the sources. For each client or source model, there is one corresponding schema. This restriction is guaranteed by a constraint in the architecture model. The link to the conceptual model is implemented by the relationship between concepts and types: each type is expressed as a view on concepts.

2.5 Physical Perspective

Data warehouse industry has mostly explored the physical perspective, so that many aspects in the physical perspective are taken from the analysis of commercial data warehouse solutions such as Software AG's SourcePoint tool [SAG96], the data warehouse system of RedBrick [RedB97], Informix's MetaCube[Info97], Essbase of Arbor Software [Arbo96] or the product suite of MicroStrategy [MStr97]. We have observed that the basic physical components in a data warehouse architecture are *agents* and *data stores. Agents* are programs that control other components or transport data from one physical location to another. *Data stores* are databases which store the data that is delivered by other components.

The basic class in the physical perspective is *DW_Component*. A data warehouse component may be composed out of other components. This fact is expressed by the attribute *hasPart*. Furthermore, a component *deliversTo* another component a *Type*, which is part of the logical perspective. Another link to the logical model is the attribute *hasSchema* of *DW_Component*. Note that a component may have a schema, i.e. a set of several types, but it can only deliver a type to another component. This is due to the observation that agents usually transport only "one tuple at a time" of a source relation rather than a complex object.

One type of component in a data warehousing environment is an *Agent*. There are two types of agents: *ControlAgent* which controls other components and agents, e.g. it *notifies* another agent to start the update process, and *TransportationAgent* which transports data from one component to another component. An *Agent* may also notify other agents about errors or termination of its process.

Another type of component is a *DataStore*. It physically stores the data which is described by models and schemata in the conceptual and logical perspective. As in the other perspectives, we distinguish between *ClientDataStore*, *DW_DataStore* and *SourceDataStore* for data stores of clients, the data warehouse and the sources.

3 Managing Data Warehouse Quality

In this section, we discuss how to extend the DW architecture model by explicit quality models and their support. There are two basic issues to be resolved. On the one hand, quality is a subjective phenomenon so we must organize quality goals according to the stakeholder groups that pursue these goals. On the other hand, quality goals are highly diverse in nature. They can be neither assessed nor achieved directly but require complex measurement, prediction, and design techniques, often in the form of an interactive process. The overall problem of introducing quality models in meta data is therefore to achieve breadth of coverage without giving up the detailed knowledge available for certain criteria. Only if this combination is achieved, systematic quality management becomes possible.

3.1 Stakeholders in Data Warehouse Quality

There exist different roles of users in a data warehouse environment. The *Decision Maker* usually employs an OLAP query tool to get answers interesting to him. A decision maker is usually concerned with the *quality of the stored data*, their *timeliness* and the *ease of querying* them through the OLAP tools. The *Data*

Warehouse Administrator needs facilities like *error reporting, metadata accessibility* and knowledge of the *timeliness* of the data, in order to detect changes and reasons for them, or problems in the stored information. The *Data Warehouse Designer* needs to measure the *quality of the schemata* of the data warehouse environment (both existing or newly produced) and the *quality of the metadata* as well. Furthermore, he needs *software evaluation standards* to test the software packages he considers purchasing. The *Programmers of Data Warehouse Components* can make good use of *software implementation standards* in order to accomplish and evaluate their work. *Metadata reporting* can also facilitate their job, since they can avoid mistakes related to schema information.

Based on this analysis, we can safely argue that different roles imply a different collection of quality dimensions, which a quality model should be able to address in a consistent and meaningful way. In the following, we summarize the quality dimensions of three stakeholders, the data warehouse administrator, the programmer, and the decision maker. A more detailed presentation can be found in [DWQ97b].

Design and Administration Quality. The design and administration quality can be analyzed into more detailed dimensions, as depicted in Figure 3.1. The *schema quality* refers to the ability of a schema or model to represent adequately and efficiently the information. The *correctness* dimension is concerned with the proper comprehension of the entities of the real world, the schemata of the sources (models) and the user needs. The *completeness* dimension is concerned with the preservation of all the crucial knowledge in the data warehouse schema (model). The *minimality* dimension describes the degree up to which undesired redundancy is avoided during the source integration process. The *traceability* dimension is concerned with the fact that all kinds of requirements of users, designers, administrators and managers should be traceable to the data warehouse schema. The *interpretability* dimension ensures that all components of the data warehouse are well described, so as to be administered easily. The *metadata evolution* dimension is concerned with the way the schema evolves during the data warehouse operation.

Software Implementation Quality. Software implementation and/or evaluation is not a task with specific data warehouse characteristics. We are not actually going to propose a new model for this task, but adopt the ISO 9126 standard [ISO91]. The quality dimensions of ISO 9126 are *Functionality* (Suitability, Accuracy, Interoperability, Compliance, Security), *Reliability* (Maturity, Fault tolerance, Recoverability), *Usability* (Understandability, Learnability, Operability), *Software Efficiency* (Time behavior, Resource Behavior), *Maintainability* (Analyzability, Changeability, Stability, Testability), *Portability* (Adaptability, Installability, Conformance, Replaceability).

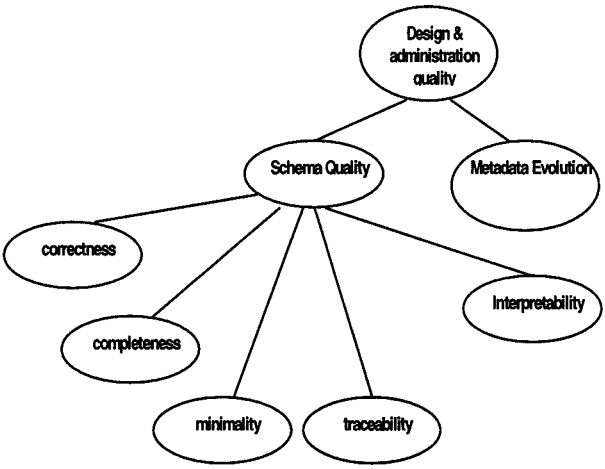

Figure 3.1 Design and administration quality dimensions

Data Usage Quality. Since databases and -in our case- data warehouses are built in order to be queried, the most basic process of the warehouse is the usage and querying of its data. Figure 3.2 shows the hierarchy of quality dimensions related to data usage.

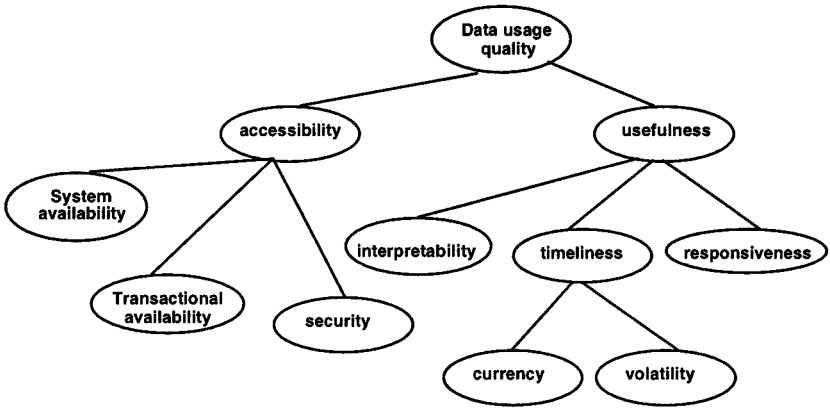

Figure 3.2 Data usage quality dimensions

The *accessibility* dimension is related to the possibility of accessing the data for querying. The *security* dimension describes the authorization policy and the privileges each user has for the querying of the data. *System availability* describes the percentage of time the source or data warehouse system is available (i.e. the system is up and no backups take place, etc.). The *transactional availability* dimension, as already mentioned, describes the percentage of time the information in the warehouse or the source is available due to the absence of update processes which write-lock the data.

The *usefulness* dimension describes the temporal characteristics (*timeliness*) of the data as well as the *responsiveness* of the system. The *responsiveness* is concerned with the interaction of a process with the user (e.g. a query tool which is self reporting on the time a query might take to be answered). The *currency* dimension describes when the information was entered in the sources or/and the data warehouse. The *volatility* dimension describes the time period for which the information is valid in the real world. The *interpretability* dimension, as already mentioned, describes the extent to which the data warehouse is modeled efficiently in the information repository. The better the explanation is, the easier the queries can be posed.

3.2 From Architecture to Quality

We now turn to the formal handling and repository-based management of DW quality goals such as the ones described in the previous section.

A first formalization could be based on a qualitative analysis of relationships between the quality factors themselves, e.g. positive or negative goal-subgoal relationships or goal-means relationships. The stakeholders could then enter their subjective evaluation of individual goals as well as possible weightings of goals and be supported in identifying good trade-offs. The entered as well as computed evaluations could be used as quality measurements in the architecture model of figure 2.3, thus enabling a very simple integration of architecture and quality model.

Such an approach is widely used in industrial engineering under the label of Quality Function Deployment, using a special kind of matrix representation called the House of Quality [Akao90]. Formal reasoning in such a structure has been investigated in works about the handling of non-functional requirements in software engineering, e.g. [MCN92]. Visual tools have shown a potential for negotiation support under multiple quality criteria [GJJ97].

However, while this simple approach certainly has a useful role in 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. In the DWQ project, such methods are systematically adopted or newly developed for all quality factors found important in the literature or our own empirical work. To give an impression of the richness of techniques to be considered, we use a single quality factor -- responsiveness in the sense of good query performance -- for which the DWQ project has studied three different approaches, one each from the conceptual, logical, and physical perspective.

We start with the logical perspective [TS97]. Here, the quality indicator associated with responsiveness is taken to be a weighted average of query and update "costs" for a given query mix and given information sources. A combinatorial optimization technique is then proposed that selects a collection of materialized views as to minimize the total costs. This can be considered a very simple case of the above Quality Function Deployment approach, but with the advantage of automated design of a solution.

If we include the physical perspective, the definition of query and update "costs" becomes an issue in itself: what do we mean by costs -- response time, throughput, or a combination of both (e.g. minimize query response time and maximize update throughput)? what actually produces these costs -- is database access or the network traffic the bottleneck? A comprehensive queuing model [NJ97] enables the prediction of such detailed metrics from which the designer can choose the right ones as quality measurements for his design process. In addition, completely new design options come into play : instead of materializing more views to improve query response time (at the cost of disturbing the OLTP systems longer at update time), the designer could buy a faster client PC or DBMS, or use an ISDN link rather than using slow modems.

Yet other options come into play, if a rich logic is available for the conceptual perspective. The description logic DWQ uses for formalizing the conceptual perspective [CDL97], allows to state that, e.g., information about all instances of one concept in the enterprise model is maintained in a particular information source, i.e. the source is complete with respect to the domain. This enables the DW designer to drop the materialization of all views on other sources, thus reducing the update effort semantically without any loss in completeness of the answers.

It is clear that there can be no decidable formal framework that even comes close to covering all of these aspects in a uniform language. When designing the meta database extensions for quality management, we therefore had to look for another solution that still maintains the overall picture offered by the shallow quality management techniques discussed at the beginning of this section but is at the same time open for the embedding of specialized techniques.

Our solution to this problem builds on the widely used Goal-Question-Metric (GQM) approach to software quality management [OB92]. The idea of GQM is that quality goals can usually not be assessed directly, but their meaning is circumscribed by questions that need to be answered when evaluating the quality. Such questions again can usually not be answered directly but rely on metrics applied to either the product or process in question; techniques such as statistical process control charts are then applied to derive the answer of a question from the measurements.

Our repository solution uses a similar approach to bridge the gap between quality goal hierarchies on the one hand, and very detailed metrics and reasoning techniques on the other. The bridge is defined through the idea of quality measurements as materialized views over the data warehouse which we already introduced in figure 2.3, and through generic queries over these quality measurements. This implementation strategy provides more technical support than usual GQM implementations. It is enabled through the powerful parameterized query class mechanism offered by the ConceptBase system.

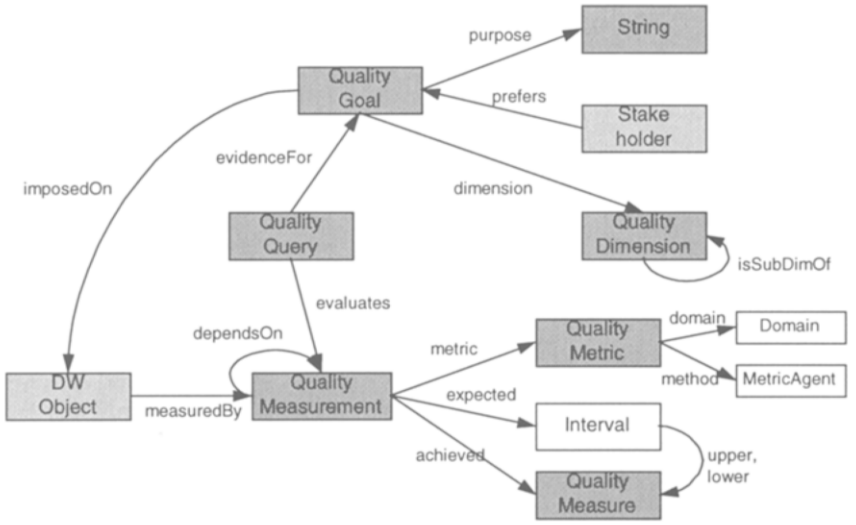

Figure 3.3: A notation for Data Warehouse Quality

The purpose of a quality goal is usually to improve some quality values of the DW or to achieve a certain quality value. Quality goals are associated with types of queries defined over quality measurements. These queries will support the evaluation of a specific quality goal when parameterized with a given (part of a) DW meta database. Such a query usually compares the analysis goal to a certain expected interval in order to assess the level of quality achieved. Furthermore, goals are established by stakeholders, who may have several subjective quality preferences. As a consequence, the quality measurement must contain information about both expected and actual values. Both could be entered into the meta database manually, or computed inductively by a given metric through a specific reasoning mechanism. For example, for a given physical design and some basic measurements of component and network speeds, the queuing model in [NJ97] computes the quality values for response time and throughput, and it could indicate if network or database access is the bottleneck in the given setting. This could then be combined with conceptual or logical quality measurements at the level of optimizing the underlying quality goal.

The interplay of goals, queries, and metrics with the basic concepts of the architecture model is shown in the Telos meta model of figure 3.3. While the development and integration of numerous specific metrics is the goal of ongoing work in the DWQ project, our current implementation just covers the upper levels of the picture, such that only manual entry of quality measurements is supported. A number of quality queries have been implemented, focusing on some that turned out to be relevant when validating the architecture against three case studies: creating a model of Software AG's SourcePoint DW loading environment, modeling data quality problems hindering the application of data mining techniques in Swiss Life, and conceptually re-constructing some design decisions underlying the administrative data warehouses of the City of Cologne, Germany [DWQ97a, DWQ97b].

Quality queries access information recorded in quality measurements. A quality measurement stores the following information about data warehouse components:

1. an interval of expected quality measures
2. the current quality measure
3. the metric used to compute a measure
4. dependencies to other quality measurements

The dependencies between quality measurements can be used to trace quality measurements outside the expected interval to their causes. The following two queries exemplify how quality measurements classify data warehouse components and how the backtracing of quality problems can be done by queries to the meta database:

```
QueryClass BadQualityMeasurement isA QualityMeasurement
 with constraint
   c: $ not (this.expected contains this.current) $
end

GenericQueryClass CauseOfBadQuality isA DW_Object
 with parameter
   badObject : DW_Object
 constraint
   c: $ exists q1,q2/QualityMeasurement
        (badObject measuredBy q1) and
        (q1 in BadQualityMeasurement) and
        (q1 dependsOn q2) and
       (q2 in BadQualityMeasurement) and
       ((this measuredBy q2) or
       (exists o/DW_Object (o measuredBy q2) and
       (this in CauseOfBadQuality[o/badObject]))) $
end
```

4 Related Work

Our approach extends and merges results from data warehouse research and data/software quality research.

Starting with the data warehouse literature, the well-known projects have focused almost exclusively on what we call the logical and physical perspectives of DW architecture. While the majority of early projects have focused on source integration aspects, the recent effort has shifted towards the efficient computation and re-computation of multi-dimensional views. The business perspective is considered at best indirectly in these projects. The *Information Manifold* (IM) developed at AT&T is the only one that employs a rich domain model for information gathering from disparate sources such as databases, SGML documents, unstructured files [LSK95, KLSS95, LRO96] in a manner similar to our approach.

TSIMMIS (*The Stanford-IBM Manager of Multiple Information Sources*) is a project with the goal of providing tools for the integrated access to multiple and diverse information sources and repositories [CGMH+94, Ull97]. Each information source is equipped with a *wrapper* that encapsulates the source, converting the underlying data objects to a common data model - called *Object Exchange Model* (OEM). On top of wrappers, *mediators* [Wie92] can be conceptually seen as views of data found in one or more sources which are suitably integrated and processed.

Similarly, but with slightly different implementation strategies, the *Squirrel* Project [HZ96, ZHK96] provides a framework for data integration based on the notion of *integration mediator*. Integration mediators are active modules that support incrementally maintained integrated views over multiple databases. Moreover, data quality is considered by defining formal properties of *consistency* and *freshness* for integrated views.

The WHIPS (*WareHouse Information Prototype at Stanford*) system [HGMW+95, WGL+96] has the goal of developing algorithms for the collection, integration and maintenance of information from heterogeneous and autonomous sources. The WHIPS architecture consists of a set of independent modules implemented as CORBA objects. The central component of the system is the *integrator*, to which all other modules report.

Turning to data quality analysis, Wang et al. [WSF95] present a framework based on the ISO 9000 standard. They review a significant part of the literature on data quality, yet only the research and development aspects of data quality seem to be relevant to the cause of data warehouse quality design. In [WRK95], an attribute-based model is presented that can be used to incorporate quality aspects of data products. The basis of

this approach is the assumption that the quality design of an information system can be incorporated in the overall design of the system. The model proposes the extension of the relational model as well as the annotation of the results of a query with the appropriate quality indicators. Further work on data quality can be found in [BT89], [BWPT93], [Jans88], [LU90], [Hall78], [Krie79], and [AA87].

Variants of the Goal-Question-Metric (GQM) approach are widely adopted in software quality management [OB92]. A structured overview of the issues and strategies, embedded in a repository framework, can be found in [JP92]. Several goal hierarchies of quality factors have been proposed, including the GE Model [MRW78] and [Boeh89]. ISO 9126 [ISO91] suggests six basic factors which are further refined to an overall 21 quality factors. In [HR96] a comparative presentation of these three models is offered and the SATC software quality model is proposed, along with metrics for all their software quality dimensions.

5 Discussion and Conclusions

The goal of our work is to enrich meta data management in data warehouses such that it can serve as a meaningful basis for systematic quality analysis and quality-driven design. To reach this goal, we had to overcome two limitations of current data warehouse research.

Firstly, the basic architecture in which data warehouses are typically described turned out to be too weak to allow a meaningful quality assessment : as quality is usually detected only by its absence, quality-oriented meta data management requires that we address the full sequence of steps from the capture of enterprise reality in operational departments to the interpretation of DW information by the client analyst. This in turn implied the introduction of an explicit enterprise model as a central feature in the architecture. To forestall possible criticism that full enterprise modeling has proven a risky and expensive effort, we point out that our approach to enterprise model formation (including the formal language used in [CDL97]) is fully incremental such that it is perfectly feasible to construct the enterprise model step by step, e.g. as a side effect of source integration or of other business process analysis efforts.

The second major problem is the enormous richness in quality factors, each associated with its own wealth of measurement and design techniques. Our quest for an open quality management environment that can accommodate existing or new such techniques led us to an adaptation and repository integration of the Goal-Question Metric approach where parameterized queries and materialized quality views serve as the missing link between specialized techniques and the general quality framework.

The power of the repository modeling language determines the boundary between precise but narrow metrics and comprehensive but shallow global repository. The deductive object base formalism of the Telos language provides a fairly sophisticated level of global quality analysis in our prototype implementation but is still fully adaptable and general; once the quality framework has sufficiently stabilized, a procedurally object-oriented approach could do even more, by encoding some metrics directly as methods, of course at the expense of flexibility. Conversely, a simple relational meta database could take up some of the present models with less semantics than offered in the ConceptBase system, but with the same flexibility.

As of now, both the framework and its implementation can only be considered partially validated. One strain of current work therefore continues the validation against several major case studies, in order to set priorities among the quality criteria to be explicated in specific metrics and analysis techniques. A second overlapping strain concerns the development of these techniques themselves, and their linkage into the overall framework through suitable quality measurements and extensions to global design and optimization techniques. Especially when progressing from the definition of metrics and prediction techniques to actual design methods, it is expected that these will not be representable as closed algorithms but must take the form of interactive work processes defined over the DW architecture.

As an example, feedback from at least two case studies suggests that, in practice, the widely studied strategy of incremental view maintenance in the logical sense is far less often problematic than the time management at the physical and conceptual level, associated with the question when to refresh DW views such that data are sufficiently fresh for analysis, but neither analysts nor OLTP applications are unduly disturbed in their work due to locks on their data. Our research therefore now focuses on extending the conceptual level by suitable (simple) temporal representation and reasoning mechanisms for representing freshness requirements, complemented by an array of design and implementation methods to accomplish these requirements and the definition of processes at the global level to use these methods in a goal-oriented manner to fulfill the requirements.

While such extensions will certainly refine and in parts revise the approach reported here, the experiences gained so far indicate that it is a promising way towards more systematic and computer-supported quality management in data warehouse design and operation.

Acknowledgements. The authors would like to thanks their project partners in DWQ, especially Maurizio Lenzerini, Mokrane Bouzeghoub and Enrico Franconi, for fruitful discussions of the architecture and quality model.

6 References

[AA87] N. Agmon, N.Ahituv, Assessing data reliability in an information system, *J. Management Information Systems* 4, 2 (1987)

[Akao90] Akao, Y., ed., *Quality Function Deployment*, Productivity Press, Cambridge MA. , 1990

[Arbo96] Arbor Software Corporation. *Arbor Essbase.* http://www.arborsoft.com/essbase.html, 1996.

[Boeh89] Boehm, B., *Software Risk Management,* IEEE Computer Society Press, CA, 1989.

[BT89] D.P. Ballou, K.G. Tayi, Methodology for allocating resources for data quality enhancement, *Comm. ACM*, 32, 3 (1989)

[BWPT93] D.P. Ballou, R.Y. Wang, H.L. Pazer, K.G. Tayi, Modeling Data Manufacturing Systems To Determine Data Product Quality, (No. TDQM-93-09) *Cambridge Mass.: Total Data Quality Management Research Program*, MIT Sloan School of Management, 1993

[CDL97] D. Calvanese, G. De Giacomo, M. Lenzerini. Conjunctive query containment in Description Logics with n-ary relations. *International Workshop on Description Logics*, Paris, 1997.

[CGMH+94] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *Proc. of IPSI Conference*, Tokyo (Japan), 1994.

[DWQ97a] DWQ, *Deliverable D1.1, Data Warehouse Quality Requirements and Framework*, NTUA, RWTH, INRIA, DFKI, Uniroma, IRST, DWQ TR DWQ – NTUA - 1001, 1997

[DWQ97b] DWQ, *Deliverable D2.1, Data Warehouse Architecture and Quality Model*, RWTH, NTUA, Uniroma, INRIA, DWQ TR DWQ – RWTH - 002, 1997

[GJJ97] M. Gebhardt, M. Jarke, S. Jacobs, CoDecide -- a toolkit for negotiation support interfaces to multi-dimensional data. *Proc. ACM-SIGMOD Conf. Management of Data*, Tucson, Az, 1997.

[Hall78] Halloran et al., Systems development quality control, *MIS Quarterly*, vol. 2, no.4, 1978

[HGMW+95] J. Hammer, H. Garcia-Molina, J. Widom, W. Labio, Y. Zhuge. The Stanford Data Warehousing Project. *Data Eng., Special Issue Materialized Views on Data Warehousing,* 18(2), 41-48. 1995.

[HR96] L. Hyatt, L. Rosenberg, A Software Quality Model and Metrics for Identifying Project Risks and Assessing Software Quality, *8th Annual Software Technology Conference*, Utah, April, 1996.

[HZ96] R. Hull, G. Zhou. A Framework for supporting data integration using the materialized and virtual approaches. *Proc. ACM SIGMOD Intl. Conf. Management of Data*, 481 - 492, Montreal 1996.

[Info97] Informix, Inc.: *The INFORMIX-MetaCube Product Suite.* http://www.informix.com/informix/products/new_plo/metabro/metabro2.htm, 1997.

[ISO91] ISO/IEC 9126, *Information technology -Software product evaluation- Quality charactyeristics and guidelines for their use, International Organization for Standardization*, http://www.iso.ch

[Jans88] M. Janson, Data quality: The Achilles heel of end-user computing, *Omega J. Management Science*, 16, 5 (1988)

[JGJ+95] M. Jarke, R. Gallersdörfer, M.A. Jeusfeld, M. Staudt, S. Eherer: ConceptBase - a deductive objectbase for meta data management. In *Journal of Intelligent Information Systems,* 4, 2, 167-192, 1995.

[JP92] M.Jarke, K.Pohl. Information systems quality and quality information systems. In Kendall/Lyytinen/DeGross (eds.): *Proc. IFIP 8.2 Working Conf. The Impact of Computer-Supported Technologies on Information Systems Development* (Minneapolis 1992), North-Holland 1992, pp. 345-375.

[JV97] M. Jarke, Y. Vassiliou. Foundations of data warehouse quality -- a review of the DWQ project. *Proc. 2nd Intl. Conf. Information Quality (IQ-97),* Cambridge, Mass. 1997.

[KLSS95] T. Kirk, A.Y. Levy, Y. Sagiv, and D. Srivastava. The Information Manifold. *Proc. AAAI 1995 Spring Symp. on Information Gathering from Heterogeneous, Distributed Environments*, pp. 85-91, 1995.

[Krie79] C. Kriebel, Evaluating the quality of information system, *Design and Implementation of Computer Based Information Systems*, N. Szyperski/ E.Grochla ,eds. Sijthoff and Noordhoff, 1979

[LRO96] A.Y. Levy, A. Rajaraman, and J. J. Ordille. Query answering algorithms for information agents. *Proc. 13th Nat. Conf. on Artificial Intelligence (AAAI-96)*, pages 40-47, 1996.

[LSK95] A.Y. Levy, D. Srivastava, and T. Kirk. Data model and query evaluation in global information systems. *Journal of Intelligent Information Systems*, 5:121-143, 1995.

[LU90] G.E. Liepins and V.R.R. Uppuluri, Accuracy and Relevance and the Quality of Data, *A.S. Loebl*, ed., vol. 112, Marcel Dekker, 1990

[MBJK90] J. Mylopoulos, A. Borgida, M. Jarke, M. Koubarakis: Telos – a language for representing knowledge about information systems.. In *ACM Trans. Information Systems*, 8, 4, 1990, pp. 325-362.

[MCN92] J. Mylopoulos, L. Chung, B. Nixon. Representing and using non-functional requirements -- a process-oriented approach. *IEEE Trans. Software Eng. 18*, 6 (1992).

[MRW78] J.A. McCall, P.K. Richards, G.F. Walters, Factors in software quality, *Technical Report, Rome Air Development Center*, 1978

[MStr97] MicroStrategy, Inc. *MicroStrategy's 4.0 Product Line*. http://www.strategy.com/launch/ 4_0_arc1.htm, 1997.

[NJ97] M. Nicola, M. Jarke. Integrating Replication and Communication in Performance Models of Distributed Databases. Technical Report, RWTH Aachen, AIB 97-10, 1997.

[OB92] M. Oivo, V. Basili: Representing software engineering models: the TAME goal-oriented approach. *IEEE Trans. Software Eng.* 18, 10 (1992).

[SAG96] Software AG: *SourcePoint White Paper*. Software AG, Uhlandstr 12, 64297 Darmstadt, Germany, 1996.

[SKR97] M. Staudt, J.U. Kietz, U. Reimer. ADLER: An Environment for Mining Insurance Data. *Proc. 4th Workshop KRDB-97*, Athens, 1997.

[TS97] D. Theodoratos, T. Sellis. Data Warehouse Configuration. *Proc. 23th VLDB Conference*, Athens, 1997.

[Ull97] J.D. Ullman. Information integration using logical views. In *Proc. 6th Int. Conf. on Database Theory (ICDT-97)*, Lecture Notes in Computer Science, pages 19-40. Springer-Verlag, 1997

[WGL+96] J. L. Wiener, H. Gupta, W. J. Labio, Y. Zhuge, H. Garcia-Molina, J. Widom. A System Prototype for Warehouse View Maintenance. *Proceedings ACM Workshop on Materialised Views: Techniques and Applications* , Montreal, Canada, June 7, 1996, 26-33.

[Wie92] G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, pp. 38-49, March 1992.

[WRK95] R.Y. Wang, M.P. Reddy, H.B. Kon, Towards quality data: an attribute-based approach, *Decision Support Systems,* 13(1995)

[WSF95] R.Y. Wang, V.C. Storey, C.P. Firth, A framework for analysis of data quality research, *IEEE Trans. Knowledge and Data Eng.* 7, 4 (1995)

[ZHK96] G. Zhou, R. Hull, R. King. Generating Data Integration Mediators that Use Materialization. *Journal of Intelligent Information Systems*, 6(2), 199-221, 1996.