Enhancing Data Warehouse Design with the NFR Framework

Fábio Rilston Silva Paim, Jaelson F. B. Castro

Universidade Federal de Pernambuco, Centro de Informática
Av. Professor Luis Freire s/n, Cidade Universitária,
CEP 50740-540, Recife, PE, Brazil
{frsp,jbc@serpro.gov.br}

Abstract. In recent years, *Data Warehouse* has emerged as a powerful technology for integrating heterogeneous data into a multidimensional repository on behalf of decision-support analysis. The complex extraction, transformation and loading process involved, as well as the aggregational-intensive queries are governed by a multitude of quality factors such as integrity, accessibility, performance, and other domain-specific non-functional requirements (NFRs). This clearly advocates the use of an NFR approach in support of building a high-quality data warehouse specification. In this work we extend the NFR Framework [3] to define catalogues of major data warehouse NFR types and related operational methods, for latter reuse during the specification stage. We illustrate the contributions of our approach in a case study on a large data warehouse project.

1. Introduction

In recent years Data Warehouse (DW) has emerged as a powerful technology for integrating sparsely distributed operational data into a comprehensive analytical fashion on behalf of an old enterprise's dream: to predict and thus make decisions upon their (near) future. The design of such systems is rather different from the design of the conventional operational systems that supply data to the warehouse. The former not only involves information requirements of decision makers, but also the structure and allocated requirements of the latter. Software engineers are required to deal with the complex process of extracting, transforming, and aggregating data while managing to deploy a solution that precisely, timely integrates with a number of heterogeneous source-provider systems; presents analytical results in an accurate, reliable form; offers flexibility at the front-end where ad-hoc queries are to be launched; and do all this with the support of a complete, non-redundant dimensional model. Thus, both operational and strategical visions have to be wrapped up in a multidimensional package to meet corporative analytical requirements that pervade pure decision-support functionality as well as strong quality constraints like *integrity*, *accessibility*, *performance* and domain-specific non-functional requirements such as *multidimensionality* [2]. This clearly advocates the use of Requirements Engineering techniques to build a precise data warehouse specification.

To pursue this goal, we innovate by proposing a methodological approach for requirements analysis of data warehouse systems in [1]. Our approach provides an iteractive, phase-oriented method to guide requirements engineers throughout the data warehouse specification process. The approach, however, is rather general with regard to exploring non-functional requirements and the alternative paths developers would have to probe into in order to understand both positive and negative influences of a certain quality requirement to the data warehouse design process.

The NFR Framework (Chung et al. [3]) fills in this gap by enabling developers to produce tailored solutions that embrace the quality characteristics of a particular domain, including priorities, related operational methods and reasoning about the influence of a non-functional choice to the system design. To deal with the large number of possible development alternatives, developers can consult the Framework's design catalogues, which organize past experience, standard techniques, knowledge about particular non-functional requirements as well as their tradeoffs and interdependences.

In this work, we adopt the NFR Framework to complement the requirements specification phase of our methodology. We start by defining a hierarchical tree of the major data warehouse non-functional requirements. Most significant non-functional requirements are further decomposed into catalogues of operational methods that, in conjunction with the Framework structure, enable engineers to select from the combination of qualitative and implementation factors that best meet users' decision-support needs.

This work is organized as follows. In section 2 we briefly present our approach for requirements analysis of data warehouse systems. In section 3 we describe the NFR Framework. The main set of data warehouse non-functional requirements, and their related Operationalization Catalogues, are discussed in Sections 4 and 5 respectively. Section 6 illustrates the contributions of the extended NFR Framework to the design of a large-scale governmental data warehouse. Section 7 summarizes our conclusions.

2. Analysing Data Warehouse Requirements

Data Warehouse systems offer efficient access to integrated and historical data from heterogeneous information sources to help managers in planning and decision-making. The data within the warehouse is extracted from the sources, consolidated, aggregated and accumulated in multidimensional data structures to support strategic analysis, powered by a technology named *OnLine Analytical Processing* (OLAP) [4]. To achieve good OLAP performance, the multidimensional model classifies data into *facts*, numeric data that quantifies a specific business activity that one wishes to analyze (e.g. *quantity of products sold*); and *dimensions*, a hierarchical classification chain of qualitative values through which data can be consolidated (e.g. *the quantity of products sold can be summarized hierarchically by* clerk*,* department *and* store *along a* "Store" *dimension*). To facilitate the assembling process, developers commonly make use of the so-called "divide to conquer" strategy to identify and deploy meaningful subsets of data, containing information related to a certain business activity, named *Data Marts*. These well-defined blocks represent a starting point in an incremental cycle that aims to deliver the enterprise-wide data warehouse by integrating each independent piece, one at a time.

Designing and implementing such an environment is a highly complex engineering task that calls for methodological support. In [1], we rely on the Twin Peaks principle [5] to propose an iterative process of requirements modeling that progressively (and simultaneously) yields an increasingly more detailed requirements specification. Initially elicited Data Mart requirements traverse a sequence of iterations that analyze, negotiate within involved groups, register and conform requirements to a broader data warehouse specification. The product of each iteration can be either a set of more refined requirements to serve as the entry point for a subsequent iteration; or a final version (baseline) of the Data Mart specification. The Specification Process integrates a phase-oriented methodology (Fig.1) that guides developers throughout the data warehouse requirements analysis, whereas template artifacts collect each aspect of the users demand.

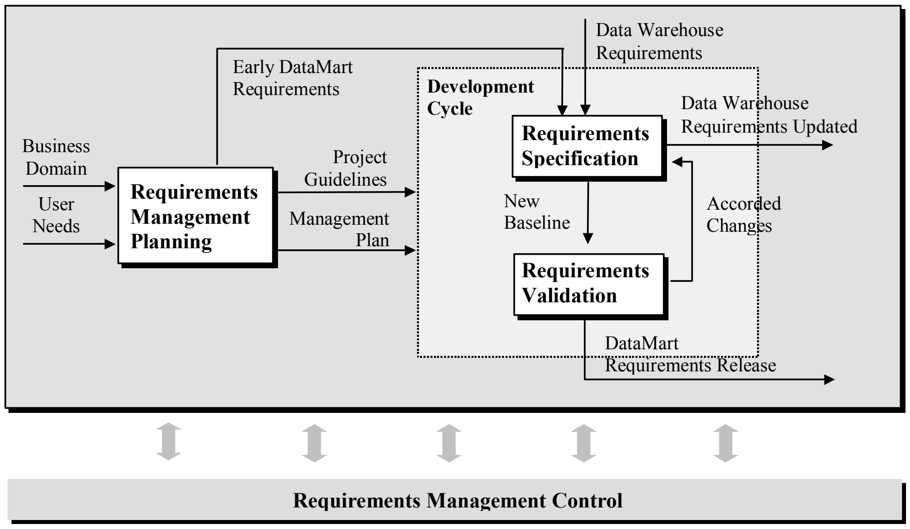
Data Warehouse Requirements
Early DataMart Requirements
Development Cycle
Data Warehouse Requirements Updated
Requirements Specification
Business Domain
User Needs
Project Guidelines
Management Plan
Requirements Management Planning
New Baseline
Accorded Changes
Requirements Validation
DataMart Requirements Release
Requirements Management Control

Fig. 1. Phase-Oriented Framework for Data Warehouse Requirements Analysis

Surrounding this process stands a backbone activity (*Requirements Management Control*) that performs permanent quality assessment of requirements change impacts. For more detailed information about the methodology, the reader can refer to [1].

3. Overview of the NFR Framework

The NFR Framework [3,6,7,8] is a goal-driven, process-oriented approach to dealing with Non-Functional Requirements (NFRs), hence being complementary to the traditional product-oriented approach whose emphasis lies in product evaluation, usually involving metrics. During the software development process, the framework allows treating NFRs as potentially conflicting or synergistic goals to achieve, while considering development alternatives which could meet the stated NFRs, examining design tradeoffs, relating design decisions to NFRs, justifying the decisions in relation to the needs of the intended application domain, and assisting defect detection [8].

A cornerstone concept in the framework is the notion of ***softgoals***. A softgoal represent a goal that has no clear-cut definition and/or criteria as to whether it is satisfied or not. Softgoals are related through relationships that represent the influence or interdependency of one softgoal on another [3]. A softgoal is said to be "satisfied" when there is sufficient positive evidence and little negative evidence against it. These interdependencies and influences can be investigated in terms of the incremental and interactive construction, elaboration, analysis and revision of *Softgoal Interdependency Graphs* (SIG – Fig.2), which comply the following basic steps[1]:

1. Develop the NFR goals and their decomposition into an AND/OR softgoal hierarchy.
2. Define knowledge catalogues to systematically organize development techniques (operationalization methods) regarding a specific NFR.
3. Identify correlations among softgoals.
4. Develop goal criticalities.
5. Analyze design tradeoffs and rationale.
6. Identify an operationalizing scenario that best satisfies quality system requirements.
7. Relate the decision made to functional requirements in the target system.

The light clouds in a SIG indicate a NFR softgoal, i.e., the NFR itself. The NFR softgoals have the following nomenclature: *Type [Topic1, Topic2,...]*, where *Type* is a non-functional aspect (e.g. *security*) and *Topic* is a subject of the target system to which the softgoal is associated (e.g. *accounts*). Topics can further be decomposed into attributes, indicated by a "." following the topic description (e.g. *accounts.balance*). The dark clouds indicate a design softgoal, i.e., a design technique, operation, constraint or other architectural component. The lines between dark and light clouds indicate the degree to which the design softgoal corresponding to the dark cloud satisfices the NFR represented by a light cloud. Satisficing can occur in four intensities: strongly positively

[1] A more recent use of The NFR Framework has been to investigate alternative architecture configurations for a target system, to which it has been proved very effective [10,36]. Nonetheless, to what concerns data warehouse systems, we are now interested in more general sets of operationalizing methods to assist data warehouse designers in settling down the basis for a future architecture. Our basic-step description reflects this goal.

satisficing (++), positively satisficing (+), negatively satisficing (-), strongly negatively satisficing (--). Determination of the degree of satisficing takes place during the step 3 above.

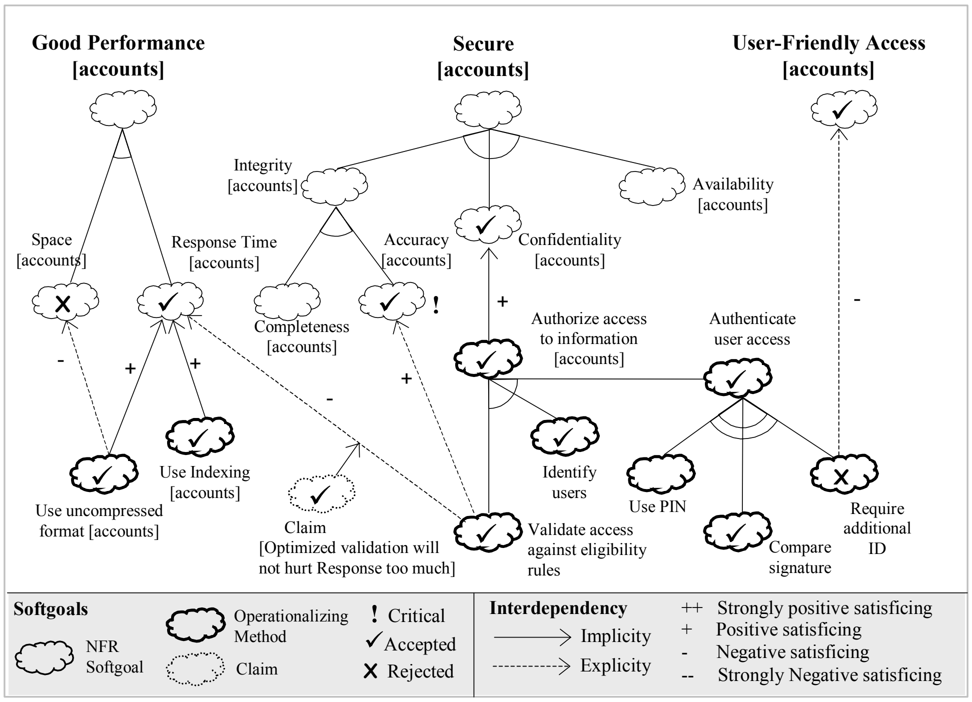

Fig. 2. Softgoal Interdependency Graph (adapted from [3])

Goal criticalities are indicated by a "**!**" sign next to the softgoals – a single "**!**" denotes that the softgoal is critical while "**!!**" indicates that the softgoal is severely critical. Critical considerations take place during the step 4 above. In addition, a single arc means an AND of all the softgoals originating from a softgoal, while a double arc means an OR of all the sub-softgoals. Finally, in step 6, an interactive labeling procedure is used to label the leaf NFR softgoals based on how much they are satisficed and their criticalities, and propagate the labels up the SIG at the end. The latest version of the NFR Framework notation uses numeric values (metrics) inside a NFR cloud to register the degree to which the generated architecture satisfices the various softgoals. In this work, however, for simplicity reasons, we will use a "✓" and an "**x**" signs instead of metrics to indicate selection (acceptance) or rejection (denying) of a given softgoal. Dashed clouds represent claim softgoals, i.e., the rationale applied to select/reject a given softgoal based on domain characteristics such as priorities and workload [10].

4. Data Warehouse NFRs

Handling quality requirements during data warehouse design involves allowing for distinct needs and domain visions of each stakeholder role. Jarke et al. put forward that decision makers are usually interested in the quality of stored data, their timeliness and easy of querying through OLAP tools [11]. Designers also need to guarantee the quality of the multidimensional schema that underpins decision-support analysis, and rely heavily on the quality of operational systems documentation and data to achieve this goal. On the other hand, database administrators are concerned with the degree of accessibility to operational source data and their consolidated version already stored in the warehouse, as well as the performance and timeliness of the loading process.

The NFR Framework organizes such quality requirements in NFR Type Catalogues, as a hierarchy of types, the more general ones shown above more specialized ones. NFR type catalogues can be customized to reflect domain-specific characteristics. Chung et al. [3] thoroughly details catalogues for Accuracy, Security and Performance. We use a Knowledge-Based approach [10] to adopt and extend this prime NFR set to provide a broad catalogue of most important data warehouse NFRs (Fig.3), based on the work of [3,8,11,12,13,14,15,16,17]. The goal here is to make further use of these NFR types and operationalization catalogues during the requirements specification phase of our methodology.

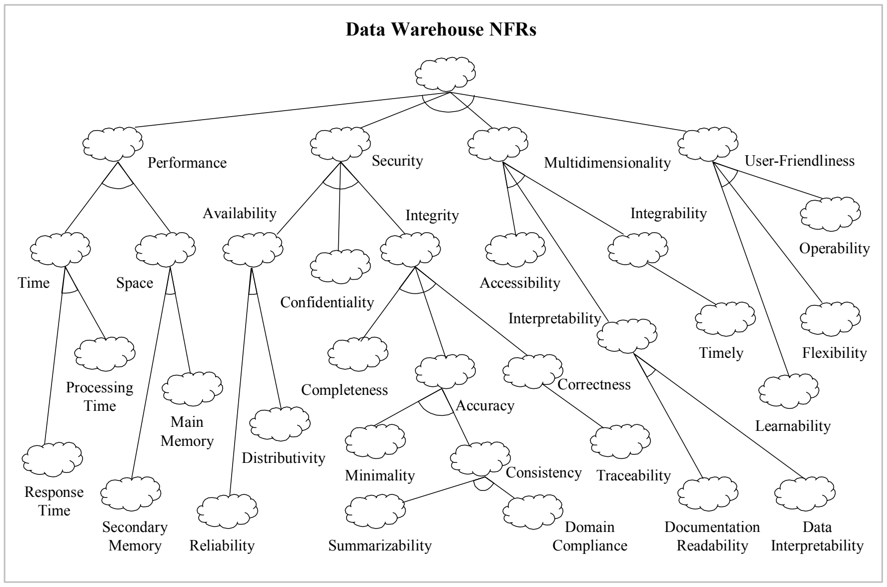

Fig. 3. Hierarchical tree representing the Data Warehouse NFR Type Catalogue

We note, however, that our proposed catalogues do not intend to enclose an exhaustive list of data warehouse NFR types and methods, but to focus on the most important ones to data warehouse design. Table 1 describes the main set of data warehouse NFRs and their related quality requirements. In the next sections we discuss in a more fine-grained fashion these main NFRs, while developing their operationalization catalogues. Due to space restrictions, we will attain our discussion to the three most relevant NFRs: *performance*, *multidimensionality*, and *integrity*.

Table 1. Data Warehouse Non-Functional Requirements

1. Performance		
The data warehouse architecture degree of efficiency at responding to a processing request.		
1.1	**Time**	The data warehouse architecture good time performance.
1.1.1	**Response Time**	Decreased or low time response to analytical querying.
1.1.2	**Processing Time**	Decreased or low time response to backstage processing.
1.2	**Space**	Efficient usage of processing memory.
1.2.1	**Main Memory**	Degree of optimization of main memory usage.
1.2.2	**Secondary Memory**	Degree of optimization of secondary memory usage.
2. Security		
The degree of information protection and resilient behavior of the data warehouse and its data.		
2.1	**Integrity**	Degree of precision and validity of multidimensional data.
2.1.1	**Accuracy**	Precision of stored data and summarized results.
2.1.1.1	**Consistency**	Logical coherence of data warehouse data.
2.1.1.1.1	**Domain-Compliance**	Data adequacy to domain standards.
2.1.1.1.2	**Summarizability**	Ability to correctly summarize data along aggregational paths.
2.1.1.2	**Minimality**	Degree up to which undesired redundancy is avoided during the source integration process.
2.1.2	**Correctness**	Extent to which the data warehouse specification maps source information to satisfy user needs.
2.1.2.1	**Traceability**	Capacity of relating stakeholders requirements to the data warehouse schema.
2.1.3	**Completeness**	Degree to which all data warehouse crucial knowledge is properly implemented on its multidimensional model and stored data.
2.2	**Availability**	Extent to which the source or data warehouse system is promptly available to all stakeholders.
2.2.1	**Reliability**	Percentage of time the source or data warehouse system is available for use considering aspects of maturity, fault tolerance and recoverability.
2.2.2	**Distributivity**	Data warehouse capacity of reaching all decision makers.
2.3	**Confidentiality**	Data warehouse capacity of guarding against unauthorized disclosure.

3. Multidimensionality		
Ability to represent decision-support requirements as and provide access to dimensional and factual data.		
3.1	**Conformance**	Ability to represent common data warehouse aspects in identically the same way across the entire data warehouse specification.
3.2	**Integrability**	Capacity of adequately and efficiently integrate operational information.
3.2.1	**Timely**	Degree to which the data updating frequency meets business users.
3.3	**Accessibility**	Possibility to access data for querying.
3.4	**Interpretability**	Extent to which data can be interpreted to efficiently model the data warehouse.
3.4.1	**Documentation Readability**	Degree to which documentation in operational sources is understandable.
3.4.2	**Data Interpretability**	Degree of data description soundness.
4. User-Friendliness		
Degree to which the data warehouse software is ease to use.		
4.1	**Operability**	The ease of operation of a data warehouse.
4.2	**Flexibility**	Extent to which the data warehouse software facilitates ad hoc querying.
4.3	**Learnability**	The physical and or intellectual skill required to learn the system.

5. NFR Operationalization Catalogues

Operationalizing Methods refine NFR softgoals into operationalizing softgoals, or the latter into more specific operationalizing softgoals [10]. These operationalizations depict possible design or implementation scenarios and once in the SIG diagram enable designers to measure the implications and contributions of these scenarios to the (or part of the) whole data warehouse design process.

5.1 Operationalizing Performance

Performance is a vital quality factor for systems [3], and this is no less true for data warehouse systems. To build a successful DW, it is crucial that its central component - the database - be a high-performance product that will meet organization's current and future needs. Inmon and Hackathorn [18] there even affirm to be a very real, although indirect, relationship between the end user’s query speed and his/her productivity as a decision-maker. This is just one side of the problem. On the other side stands an enormous amount of data to be extracted, transformed and loaded onto the repository. As data warehouse applications usually deal with aggregation of millions of data, both backstage and query performance are essential aspects to be considered.

There are several techniques to tackle this issue (Fig.4). One can generally organize them into two large groups: techniques for optimizing **time** and **space** performance. The first group of techniques seeks to offer best response and processing times. Optimal response time can be achieved by pre-aggregating harvested data in database views, and breaking the traditional data modeling rules by denormalizing the database. To overcome data retrieving overload, data warehouses make large use of joining and indexing techniques. Besides accessing, data must be efficiently populated into the repository. Good processing performance often requires shared disk architectures [20] and, whenever the warehouse exceeds billions of data, parallelism of processors. The second group of techniques deals with optimal space usage, at both main and secondary memory levels. Blocksize adjustment and Caching [12] eases main memory usage, whereas data duplication techniques like Data Mirroring and RAID-5 [21] enhance file disk (secondary memory) performance. Yet, Data Partitioning is another solution to improve both time and space performance, where multiple tables are created to store minute levels of data.

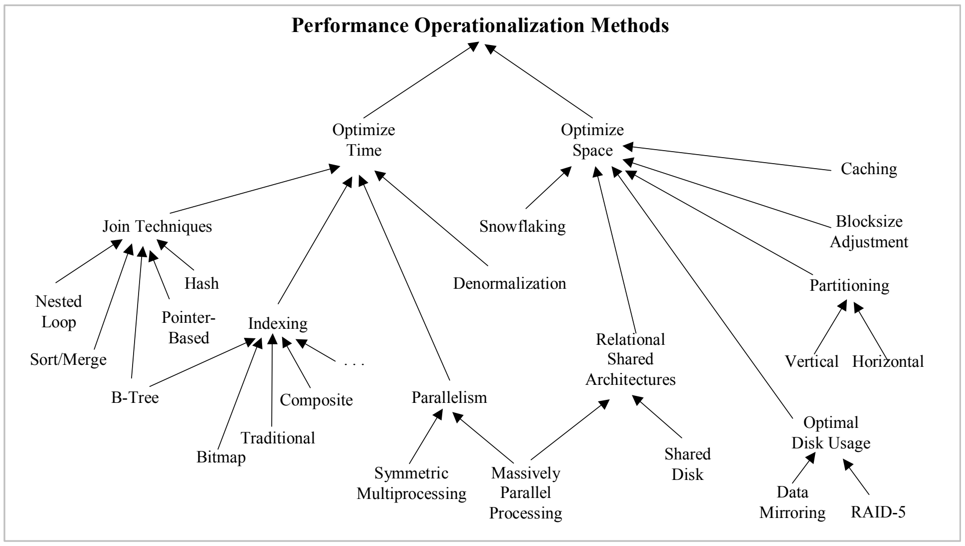

Fig. 4. Performance Operationalization Catalogue

5.2 Operationalizing Multidimensionality

Multidimensionality [2], an aspect unique to data warehouse systems, is normally stated as a technique of modeling information as facts (what we want to analyze) and dimensions (what we shall use to analyze). To the quality extent, it requires more than a collection of facts and dimensions. Building a multidimensional architecture entails accessing external and internal information under strict time and quality control constraints; integrating raw information to derive suitable strategic information; and conforming common dimensional requirements to get reusability. We broaden then the multidimensionality concept to embrace all this intertwined quality factors, while compose a related methods catalogue as demonstrated in Fig.5.

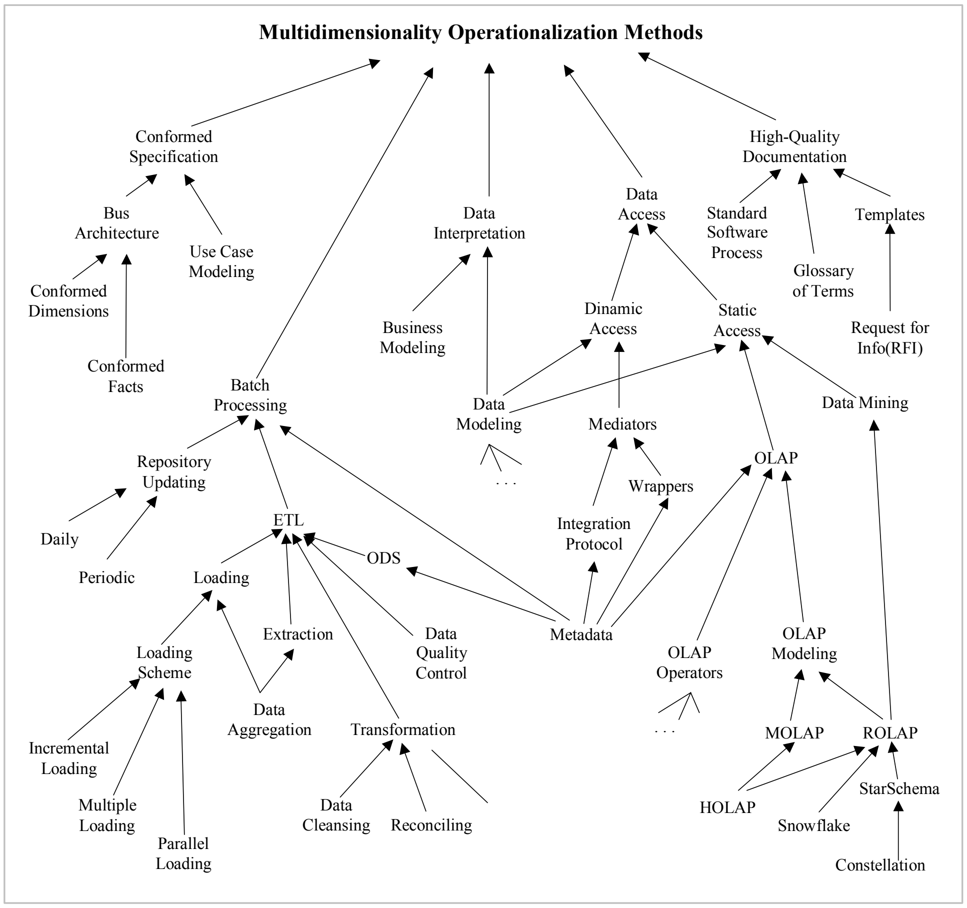

Fig. 5. Multidimensionality Operationalization Catalogue

The first step towards multidimensionality is to integrate operational data on a timely basis. A great number of techniques have been proposed in the literature to deal with ETL (Extraction/Transformation/Loading) activities such as data cleansing, summarization, and reconciling ([23]). Data administrators must also decide for the loading periodicity that offers best tradeoff between users requirements and environment restrictions. An essential condition when planning for ETL is to assure good quality external source documentation, which is strongly facilitated by adopting software documentation and development standards. Data description soundness is almost mandatory to guarantee cost-effective data loading and querying. Data Modeling facilities like the use of domains, primary and foreign keys, aliases, as well as business modeling techniques contribute to increase data interpretability.

Data Accessibility is at the core of multidimensional databases. Data warehouses aim at providing fast and up-to-date access to business information. Unfortunately, these two aspects are somehow conflicting. In order to gain fast retrieving on query processing, OLAP operations work on relational (ROLAP) or multidimensional (MOLAP) data architectures that store summarized data. This approach, however, lacks freshness of information, once data content might have been changed after being materialized into the repository. To attack this problem, modern DW architectures make use of Mediators [24] to permit direct access to operational data.Yet, real-time integration schemes like mediators tend to decrease the querying response-time, apart from being a relatively new technique on the commercial scene, which is dominated by the static access approach and its multidimensional modeling techniques: *star* (denormalized) and *snowflake* (normalized) schema [27]. A majority of data warehouses is built upon the star schema (where dimension tables are stick to a central fact table, forming a star shape) due to its incredible speed performance. Still, a single dimension table can serve more than one fact table (e.g. *Time*). It is, thus, fundamental for such dimension to possess a unique, multi-service structure. Analogously, there might be facts common to more than one fact table (e.g. *sales total*). One can accomplish a certain degree of multidimensional conformance by utilizing systematic approaches as in [1] and planning for bus architecture construction as in [12]. Metadata [25] holds a major supportive role across all previously mentioned techniques, collecting all necessary information to assist in the ETL phase and content analysis.

5.3 Operationalizing Integrity

The completeness, correctness and accuracy of the data extracted and fed into the warehouse will be a function of the reliability and authenticity of the warehouse to its users. Thus, in data warehouse environments, there needs to be a means to ensure the integrity of data, first by having procedures to control the quality of the data movement from operational systems, and second by having controls to assure consistent data consolidation and presentation. Data integration techniques (see Sect. 5.2) like data cleansing positively satisfice integrity requirements. Notwithstanding, final adjustments are still needed to ensure accuracy of data in terms of minimality and consistency. Integrity constraints are basic to deploy minimal (non-redundant) data and data aggregation assurance (summarizability). Data consistency is also determined by the degree of data compliance to business domains, which is empowered by large use of metadata, integration standards and data quality control procedures. These last two assets are very important means of improving data correctness and completeness, and so is a good requirements analysis process. Adding to the issue of correctness, it is extremely required to track user requirements accommodation to the multidimensional schema with means like traceability matrices [26]. Fig.6 summarizes the main integrity-assurance techniques.

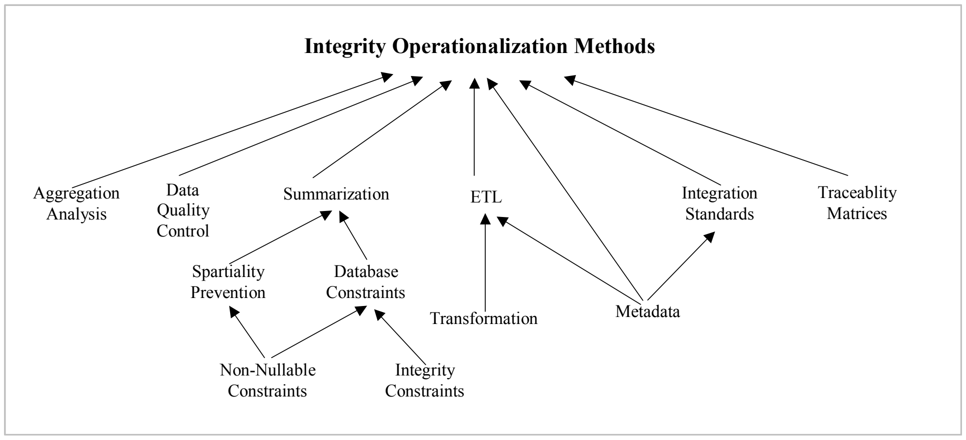

Fig. 6. Integrity Operationalization Catalogue

6. Using the Data Warehouse-Extended NFR Framework

We now illustrate the application of the Data Warehouse-Extended NFR Framework (hereafter DW-ENF) to the development of a large data warehouse system named SAFE[2]. SAFE was developed by SERPRO, a Federal Data Processing Agency of the Brazilian Government, to provide the Internal Revenues Department with a detailed strategical view over the federal taxes collecting process. SAFE stores tax information in a subject-driven perspective to perform complex OLAP queries. Subjects are defined as client's core business areas, and modeled by means of single Data Mart solutions. A central fact table in each Data Mart holds real world facts vital to the analysis scenario envisioned for each subject.

SAFE is an ongoing project conducted under the premises of our proposed methodology [1]. To enhance the power of our method (and SAFE's final quality as well),

[2] SAFE is an acronym to "*Sistema de Análises Fiscais Estratégicas*" (Internal Revenues Strategic Information System).

we made use of the Extended NFR Framework during the Requirements Specification phase to examine optimal design alternatives. Due to space restrictions, we chose three design critical requirements to work on in this brief case study:

R1 – Query response time must not exceed 1 minute;
R2 – The system should present the most up-to-date information as possible;
R3 – The system must allow the user to drill across data from different subjects;

One can realize from the DW-ENF that these quality criteria relate directly to the performance and multidimensionality potential expected from SAFE. The question to be answered at this point is which design alternative best fits the three conditions. We will have thus to investigate the querying *response time*, the *timeliness* of data loading and the *accessibility* implementation methods. Using the DW-ENF catalogues, we construct the SIG shown in Fig.7.

From R1 we conclude that *time performance* is critical to SAFE. In fact, a robust machine with terabytes of storage capacity has been acquired to deal with the space issue, so that space is no longer a major issue to the project. This leads us to mark the **Time** softgoal with a "**!**". Still, **response time** prevails against processing time, for the frequency of data updating overcomes the time to process data in significance, which can be taken from R2 and also from the mentioned architectural facilities. To represent this, we mark the Timely softgoal with a "**!**" and the Response Time softgoal with a "**!!**". Time investigation will focus now on the response time factor.

Now let us take a closer look over the performance problem. The SIG shows that, in order to optimize time response, the designer can choose from any of the following methods: *join* and *indexing* techniques, *parallelism* of processors, and/or data *denormalization*. However, the tool used to implement the OLAP environment automatically controls and optimize data joining. Yet, SAFE's warehouse is expected to store up to a hundred millions of factual and dimensional registers, which eliminates the need for investments in expensive parallel processing. From the prior arguments, the data warehouse designer can reject respectively the Join Techniques and the Parallelism softgoals. We continue with the analysis of indexing techniques. SAFE Interface enables users to perform regular as well as complex analytical queries. Complex queries frequently perform joins between multiple tables, which is empathized by the need of drill across operations (R3). Regular queries, instead, return a low amount of data from a few fact tables. The two kinds of queries require distinct indexing approaches, indicated by the **Indexing[queries]** softgoal decomposition into **Indexing[RegularQueries]** and **Indexing[ComplexQueries]**. Regular queries are well supported by B-Tree Indexes, while complex ones require more sophisticated indexing schemes like Bitmap Indexes. The designer chooses B-Tree and Bitmap Indexing, then, to satisfice SAFE's query performance requirements.

On the other hand, requirement R3 strongly suggests the adoption of a special form of star schema namely Constellation [27], where single star schemas are hierarchically linked from their fact tables. The links between fact tables will provide the ability required by the client to “drill across” different subject information, which is indicated by the strong interdependency between the Constellation and the Drill Across design softgoals. Denormalization, in turn, becomes an eligible softgoal, as it strongly supports star schemas.

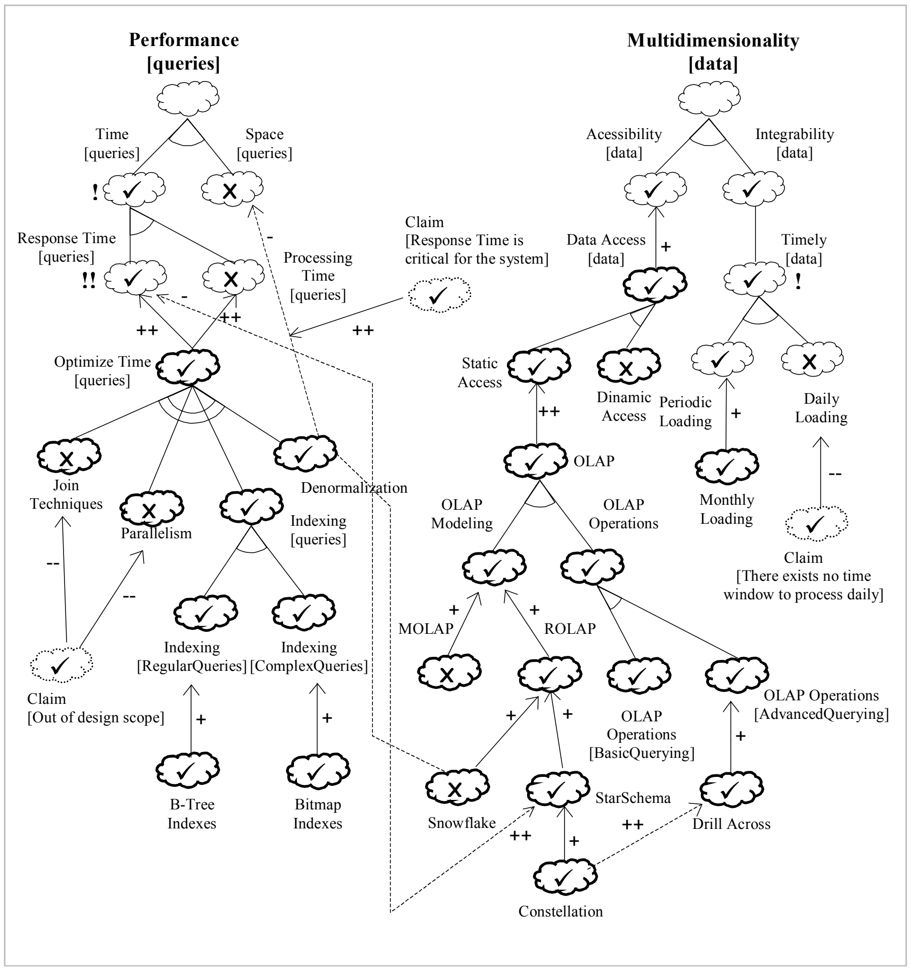

Fig. 7. Example of DW-ENF application in the SAFE system design

Although it may hurt space performance, such is not a concern in this project. Besides, the other logical choice, Snowflaking, has been denied for it hurts the dominant *response time* softgoal. Ultimately, the designer is confronted with the need for very timely loading of data. Requirement R2 suggests that a daily processing approach be chosen. This approach, however, has to be denied due to the absence of sufficient "time window" to extract data from all sources (*this activity takes place inside SERPRO's mainframes*). In fact, the shortest period of time when all data could be loaded into the warehouse is a month. Thus, a monthly loading approach is accepted, and the client is informed of the update restriction.

We finish our brief investigation with the concluding design alternative for the SAFE system, regarding requirements R1, R2 and R3:

The data warehouse architecture will be built around a constellation schema, with interlinked fact tables to enable drill across operations. Data denormalization, together with B-Tree and Bitmap indexes will be used to improve query speed.

7. Related Works

Quality characteristics of software have been an important theme in software engineering for a long time [29]. In recent years, Requirements Engineering has covered more in-depth the crucial role played by quality (or non-functional) requirements in software specification [31,32,33]. The first efforts addressed the what-how range of the requirements problem, leaving an implicit, quantitative notion of requirements behind data and operation sets. Further works attempted to capture the reasons why requirements were needed in a given software specification, and whether they satisfied higher-level objectives (goals) that naturally arise in any requirements engineering process [30].

In the early nineties, two complementary frameworks were proposed for integrating goals and requirements modeling. The first framework [31] utilized a quantitative approach to investigate the degree of satisfaction between goals and refined subgoals, introducing operationalizing links to relate goals to requirements on operations and objects. The NFR framework [6] appeared subsequently to propose a qualitative approach where lower-level requirements are expected to satisfy more abstract "soft" goals. The NFR Framework has been used ever since in a variety of ways and domains to treat non-functional requirements [8,9,10,28,33]. Our work is, however, the first example of application of the NFR Framework in the data warehouse domain. In fact, few researches have addressed the quality assurance theme in such domain. The long term DWQ Project [19] heads most of the works in the area, looking forward to developing a semantic foundation that allows data warehouse designers to link the choice of architectural models with quality-of-service factors. As part of this project, the works of [11,16] developed a comprehensive study about the influence of quality requirements in the decision-support systems design, from the point of view of involved stakeholders. Kimball et al. [12] discuss techniques and procedures to assure data warehouse high-quality data, while [13,36] analyze thoroughly particular multidimensional quality factors. Our approach goes beyond these works to propose a framework that assists designers in a higher-level of abstraction to deploy user-fitting solutions.

8. Conclusions

In this paper we have addressed the problem of dealing with non-functional requirements in the development of data warehouse systems. We extended the NFR Framework to define a set of data warehouse-specific NFR types and operationalization catalogues, which we referd to as DW-ENF. The catalogues can be further reused during the Requirements Specification phase of an innovative data warehouse requirements analysis methodology to investigate design alternatives that *satisfice* overall users' quality constraints. We illustrated the potentiality of our DW-ENF approach with an excerpt from the development process of a large data warehouse system for the Brazilian Government.

Although simple in nature, the study case gives an idea of how designers can benefit from using the DW-ENF approach to build quality into decision-support systems, and bridge the gap between functional and non-functional aspects of the application. Complex systems like SAFE, however, demand more intricate tradeoff analysis that cannot be easily accomplished without tool support. We intend to use the NFR-Assistant tool [35] to explore in more deeply ways the structure and contributions of the DW-ENF framework to the development of data warehouse systems in our ongoing project with SERPRO. Further works include also extending the framework to investigate weaknesses and strengths of alternative architecture configurations of a given data warehouse solution to support design evolution.

References

1. Paim, F. R., Carvalho, A. E., Castro, J. B. "Towards a Methodology for Requirements Analysis of Data Warehouse Systems". In Proc. of the XVI Simpósio Brasileiro de Engenharia de Software (SBES2002), Gramado, Rio Grande do Sul, Brazil, 2002.
2. Abelló, A., Samos, J., Saltor, F. "Benefits of an Object Oriented Multidimensional Data Model". Lecture Notes in Computer Science, a. 1944, pg. 141 ff, Proc. of Objects and Database 2000 (ECOOP Workshop), France, 2000.
3. Chung, L., Nixon, B., Yu, E., Mylopoulos, J. Non-Functional Requirements in Software Engineering. Kluwer Publishing, 2000.
4. Codd, E. F., Codd, S. B., Salley, C. T. "Providing OLAP (Online Analytical Processing) to User Analyst: an IT Mandate". White paper at http://www.arborsoft.com/OLAP.html, Arbor Software, 1993.
5. Nuseibeh, B. "Weaving The Software Development Process Between Requirements and Architecture". Proceedings of ICSE2001 International Workshop: From Software Requirements to Architectures (STRAW-01), Toronto, Canada, 2001.

6. Mylopoulos, J., Chung, L., Nixon, B. "Representing and Using Non-Functional Requirements: A Process-Oriented Approach". IEEE Transactions on Software Engineering, Vol. 18, No. 6, June 1992, pp. 483-497.
7. Mylopoulos, J., Chung, L., Liao, S. S. Y., Wang, H., Yu, E. "Exploring Alternatives during Requirements Analysis". IEEE Software, Jan/Feb, 2001, pp. 2-6.
8. Chung, L., Nixon, B. "Dealing with Non-Functional Requirements: Three Experimental Studies of a Process-Oriented Approach". In Proceedings of the IEEE 17th International Conference on Software Engineering (ICSE), Seattle, April 24-28, 1995, pp. 25-37.
9. Chung, L., Subramanian, N. "Process-Oriented Metrics for Software Architecture Adaptability". In Proceedings of the 12th Intl. Symposium on Software Reliability Engineering (ISRE), Hong Kong, China, 2001.
10. Subramanian, N., Chung, L. "Software Architecture Adaptability: An NFR Approach". In Proceedings of the Intl. Workshop on Principles of Software Evolution (IWPSE`01), September 10-11, Vienna, Austria. IEEE Computer Society Press, 2001.
11. Jarke, M., Jeusfeld, M., Quix, C., Vassiliadis, P. "Architecture and Quality in Data Warehouses: An Extended Repository Approach". Information Systems, Vol. 24, No. 3, 1999, pp. 229-253.
12. Kimball, R., Reeves, L., Ross, M., Thornthwaite, W. The Data Warehouse Lifecycle Toolkit. New York, John Wiley & Sons, 1998.
13. Lenz, H.-J., Shoshani, A. "Summarizability in OLAP and Statistical Data Bases". In Proc. of the 9th Intl. Conference on Scientific and Statistical Database Management (SSDBM), IEEE Computer Society, 1997.
14. Pressman, R. S. Software Engineering: A Practitioner 's Approach. McGraw-Hill Book Company (New York), fifth edition, 2000.
15. Date, C. J. An Introduction to Database Systems. Addison-Wesley Publishing Company, seventh edition, 1999.
16. Vassiliadis, P., Bouzeghoub, M., Quix, C. "Towards Quality-oriented Data Warehouse Usage and Evolution". In Proceedings of the 11th Conference on Advanced Information Systems Engineering (CAiSE '99), Heidelberg, Germany, 1999.
17. Vassiliadis, P. "Data Warehouse Modeling and Quality Issues". PhD thesis, Department of Electrical and Computer Engineering, National Technical University of Athens, Greece, 2000.
18. Inmon, W. H., Hackathorn, R. D. Using the Data Warehouse, John Wiley & Sons, 1994.
19. Foundations of Data Warehouse Quality – DWQ Project. http://www.dbnet.ece.ntua.gr/~dwq/.
20. Spiliopoulou, M., Hatzopoulos, M., Contronis, Y. "Parallel Optimization of Large Join Queries with Set Operators and Aggregates in a Parallel Environment Supporting Pipeline". IEEE Transactions on Knowledge and Data Engineering, 8(3): 429--445, June, 1996.
21. Asami, S. "Reducing the cost of system administration of a disk storage system built from commodity components". Technical Report CSD-00-1100, University of California, Berkeley, 2000.
22. Marco, D. Building and Managing the Meta Data Repository: A Full Lifecycle Guide. John Wiley & Sons, first edition, 2000.
23. English, L. Improving Data Warehouse and Business Information Quality: Methods for Reducing Costs and Increasing Profits. John Wiley & Sons, first edition, 1999.
24. Wiederhold, G. "Mediators in the Architecture of Future Information Systems". IEEE Computer, 25(3): 38-49, 1992.
25. Do, H. H., Rahm, E. "On Metadata Interoperability in Data Warehouses". Technischer Report 1-2000, Institut für Informatik, Universität Leipzig, 2000.
26. Sommerville, I., Kotonya, G. Requirements Engineering: Processes and Techniques. Addison-Wiley, 1997.

27. Moody, D. L., Kortink, M. A. “From enterprise models to dimensional models: A methodology for data warehouse and data mart design”. In Proceedings of 2nd Intl. Workshop on Design and Management of Data Warehouses (DMDW), Stockholm, Sweden, 2000.
28. Chung, L., Nixon, B. A., Yu, Eric. “Using Quality Requirements to Systematically Develop Quality Software”. In Proceedings of the 4th International Conference on Software Quality, McLean, VA, USA, October 3-5, 1994.
29. Boehm, B. W., Brown, J. R., Kaspar, H., Lipow, M., MacLeod, G. J., Merritt, M. J. Characteristics of Software Quality. Amsterdam: North-Holland, 1978.
30. van Lamsweerde, A. "Requirements Engineering in the year 00: A Research Perspective". In Proceedings of the International Conference on Software Engineering (ICSE’2000), Limerick, June, 2000 (invited paper).
31. Dardenne, A., van Lamsweerde, A., Fickas, S. “Goal-Directed Requirements Acquisition”. Science of Computer Programming, Special Issue on 6th Intl. Workshop on Software Specification and Design, Como, Italy, 1991.
32. Fickas, S. Automating the Software Specification Process. Technical Report 87-05, University of Oregon Computer Science Department, December, 1987.
33. Jarke, M., Bubenko, J., Rolland, C., Sutcliffe, A., Vassiliou, Y. “Theories Underlying Requirements Engineering: An Overview of NATURE at Genesis”. Intl. Symposium on Requirements Engineering, San Diego, CA, January, 4-6, 1993.
34. Chung, L., Subramanian, N. "Architecture-based Semantic Evolution for Embedded Systems: A Study of Remotely Controlled Systems". To appear in Journal of Software Maintenance and Evolution.
35. Tran, Q., Chung, L. "NFR-Assistant: Tool Support for Achieving Quality". In Proc. of IEEE Symposium on Application-Specific Systems and Software Engineering and Technology (ASSET'99), Richardson, Texas, March 24 - 27, 1999.
36. Theodoratos, D., Bouzeghoub, M. “Data Currency Quality Factors in Data Warehouse Design”. In Proc. of Intl. Workshop on Design and Management of Data Warehouses (DMDW'99), Heidelberg, Germany, June 1999, pp. 15 / 1-16.