Towards a Methodology for Requirements Analysis of Data Warehouse Systems

Fábio Rilston Silva Paim, Ana Elizabete Carvalho, Jaelson Brelaz de Castro
Universidade Federal de Pernambuco, Centro de Informática, Recife, Brazil
{frsp, aesc, jbc}@cin.ufpe.br

Abstract

Few researches have been carried out to provide means of requirements acquisition and assessment in the novel field of data warehousing. This work proposes a methodology for requirements specification in the context of data warehouse systems, as a way to accomplish the peculiar aspects inherent to such systems. Moreover, a case study and its practical contributions in modeling a strategic information system for the Brazilian Government are presented.

Key words: requirements engineering, methodology, data warehouse, twin peaks.

1. Introduction

Requirements Engineering has focused on the threefold goal of capturing, analyzing and managing software requirements information, as remarkably noted in [21], [19], [14]. These overall goals have commonly been revisited every time the dynamics of computer science grants scientific community with a new technology. Recently forged on this field, *Data Warehouse* technology has flourished as a powerful way to extract, integrate and analyze data from heterogeneous sources. The reason behind this success lies on its great contribution: making predictions about the (near) future, a feature ever searched for business companies.

However, the multidimensionality [1] inherent to data warehouse applications presents a new challenge to software requirements analysis. Information in such systems is a matter of data exploitation and integration under a subject-oriented paradigm, which requires strong knowledge from both high-level users and software development team. More than ever, success here depends on building the most precise user requirements specification, as a basis for assembling an efficient architecture, capable of supporting the strategic organization level with all information its decision makers might require. In this sense, a methodological approach is required to guarantee (*i*) the correct comprehension of user requirements, (*ii*) a fine-tuned design phase and (*iii*) the construction of a dimensional schema that enables decision makers to perform all necessary analysis. Furthermore, such methodology, allied to efficient requirements management tools, works as an instrument to trace the changes in user requirements along the project, and to allow developers to make the proper, on-time maintenance on the application data model.

In this work we propose a methodology for requirements analysis of data warehouse systems. Our approach is twofold: (*i*) defining a phase-oriented methodology to serve as a guide throughout the data warehouse specification process; (*ii*) generating a set of artifacts to collect each aspect of the users demand. Moreover, we describe a practical application of this approach carried out at SERPRO, a Data Processing Agency of the Brazilian Federal Government, and its main contributions to the underlying project.

This work is organized as follows. In section 2 we present the basic concepts pertaining the data warehouse technology. In section 3 we establish a parallel between related works and

our present approach. Section 4 discusses the specificities of a decision-support system's development process. In section 5 we present our methodology, followed by the case study reported in section 6. Section 7 summarizes our concluding remarks.

2. Data Warehousing

The term *Data Warehouse* was first coined by [11] to describe "*a collection of consistent, subject-oriented, integrated, time-variant, non-volatile data in support of management's decisions* ". More than just a collection of data, *data warehousing* is a defined process pertaining 3 phases: (*a*) **extracting** data from distributed operational sources (mostly legacy systems); (*b*) **organizing and integrating** data consistently into the warehouse; and (*c*) **accessing** integrated data in an efficient and flexible fashion. The main contribution of a data warehouse relies on its capability of transforming data into strategic information, accessible to decision-makers in the highest levels of an organization. Such capability is supported by the use of OLAP (*OnLine Analytical Processing*) technology [4], which provides final users with configurable views of data from different angles and on different aggregation levels.

Fast and flexible OLAP analysis can only be achieved if data are arranged in a multidimensional form [1] where information is classified according to facts and dimensions. *Facts* are numeric or factual data that represents a specific business activity that we wish to analyze. *Dimensions* are single perspectives on the data that determines the *granularity* (the data detail level) to be adopted for fact representation. Fact units and their values are referred to as *measures*. In addition, each dimension is described by a set of descriptive *attributes*, which qualify the data content. For instance, in a company whose commercial activity is devoted to vehicle sales, the **sales amount** of imported vehicles is a *fact*, which can be analyzed through the perspective of the stores that performed sales along the year. The chain of **stores**, the car **category**, and the **time** (date) of a sale correspond to *dimensions*. Moreover, these dimensions can be described in terms of *attributes* such as **name**, **city** and **state**, for the store dimension; **name** and **type**, for the car category dimension; and finally **day**, **month**, and **year**, for the time dimension. Still, attributes in a dimension can be arranged as a *hierarchical* chain, allowing measures to be assembled (classified or aggregated) along the hierarchy. Recalling our latter example, *name* and *day* represent the lowest granularity level of each dimensional hierarchy.

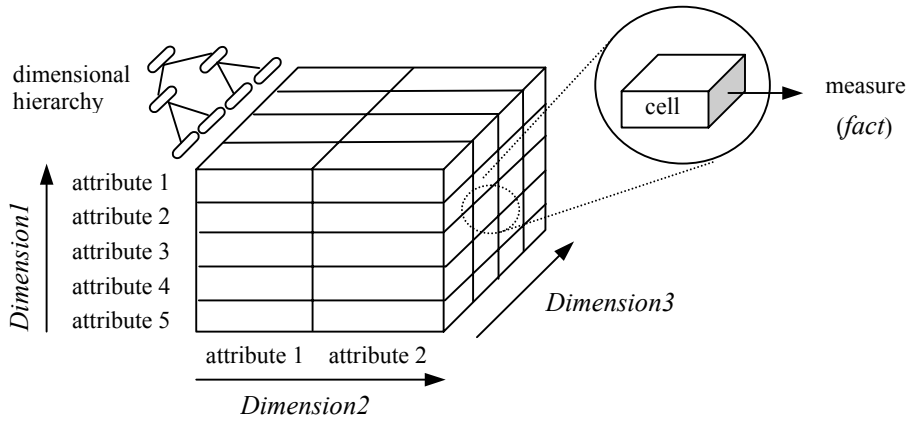

Figure 1. Data cube multidimensional metaphor ([3]).

The *data cube* metaphor is frequently used to clarify the structural multidimensionality inherent to data warehouses [3], each cell representing an intersection point between *n* dimensions, which holds a measure on which analysis is to be performed (***Figure 1***). Furthermore, in order to facilitate the management of the data warehouse assembling process,

developers commonly make use of the so-called "*divide to conquer*" strategy to identify and deploy meaningful subsets of data, in form of small data warehouses, containing information related to a certain business activity. These well-defined blocks of strategic information are referred to as *Data Marts*, and represent a starting point in an incremental cycle that aims to deliver the enterprise-wide data warehouse by integrating its independent blocks, one at a time.

3. Related Works

Few works in the literature attempt to connect Requirements Engineering to Data Warehouse Systems. Such works conceive requirements analysis as one of a sequence of phases that compound an entire development lifecycle ([13], [8]). The requirements phase is introduced as a prior step to conceptual design. Informal techniques for requirements elicitation, documentation, analysis and validation (not necessarily all together) play a more significant or incidental role, depending on the particular approach. Besides the heterogeneous concern regarding each technique, none of these works offer a consistent, fully-fledged methodology specifically aimed at requirements analysis and management. We discuss here some works that explore the requirements phase in a more detailed level.

In [9] the operational Entity-Relationship Schema of transactional applications delivers basic information to determine eligible multidimensional requirements. Business Domain experts select strategically relevant operational attributes that are classified as dimensions and/or measures. The resulting requirements are presented in a tabular list of attributes along with their multidimensional purpose. Supplementary information (*integrity constraints, additional derived requirements*) can be added informally in a textual appendix. Although it innovates by proposing a useful artifact for requirements elicitation (the "tabular listing"), this approach lacks addressing a reliable requirements acquisition process.

[6] makes use of an Object-Oriented Software Engineering [12] approach to depict the enterprise goals and objectives, and develops an Object-Oriented Conceptual Framework for data warehouses development life cycle. Within this framework, business requirements are described by means of use-cases, which are directed to identify *actors* (stakeholders); express behavior between actors and business *subprocesses*; and thus specify *objects* in support of the required dimensional architecture. Despite its complexity and technical-driven sophistication, the latter approach shows little commitment to a realistic, high-leveled requirements analysis methodology, whereas it proves to be strictly suited to object-oriented environments. We argue that our approach is neither dependent of the software language nor it is of the database logical/physical organization.

For the sake of completeness, one cannot skip looking into the strict field of requirements engineering to note important techniques like Viewpoint [23], Goal-oriented [22], or Non-Functional Requirements [24] analysis. They extend the core requirements engineering phases and go beyond the *what-how* analytical paradigm to bound requirements to the reasoning (*why*) that motivates their specification, as well as to the alternative goals that affect their allocation into the software-to-be [25]. On the one hand, it seems obvious that the same line of thought deserves certain attention when developing decision-support systems. On the other hand, as later discussed in this work, the specificities that make this domain area unique suggest that applying these techniques to model multidimensional requirements requires specific reevaluation and tailoring, which are out of the scope of this paper.

4. Data Warehouse Development: Engineering between Peaks

The development of data warehouse systems is rather different than the development of conventional operational systems [3]. Designing and implementing such an environment is a

highly complex engineering task that calls for a methodological support [10], capable of weaving together architectural and specification requirements. An alternative to reduce the gap between such apparently distinct requirements is to adopt an iterative process to produce progressively (and simultaneously) more detailed requirements and design specification, like the Twin Peaks Development Process proposed in [15]. In fact, requirements analysis in data warehouse systems cannot be performed without taking into account (multidimensional) architecture constraints that exert static as well as dynamic influence in the system scope, and compels developers to focus equally on either of these two "peaks" at any one time. Our experience supports that the three management principles advocated in the Twin Peaks Model are proven effective to such modern applications. With regard to the **IKIWISI** (*I'll know it when I see it*) principle, data warehouse requirements are often clarified when stakeholders assess the system-to-be by means of prototypes and early schema definitions. Although componentized units of work (**COTS**) are not a tendency in data warehouse development, reuse of early data warehouse requirements can be increasingly valuable to the evolution of a *Data Mart* solution. Furthermore, an approach that pinpoints the distinction between data warehouse *core* requirements and strict *Data Mart* ones improves the multidimensional model integration, and consequently strengthens such architectures against the **rapid changes** of system requirements to which they are naturally exposed. Our approach does it so by emphasizing all the aforementioned aspects.

Nevertheless, in order to better correlate architecture and problem specification, the following set of concepts needs to be accommodated while performing requirements engineering in data warehouses:

i) **Represent *facts* and their properties** – Facts are central to data warehouses. Analyzing user requirements implies identification of facts through perceiving the measures behind user demands;

ii) **Distinguish and connect *dimensions* to facts –** dimensions offer the key to understand fact measures by allowing the user to view data through a specific (mostly strategic) point of view. Sometimes the analysis of a single user statement gives rise to a number of candidate dimensions. For instance, the sentence "*Monthly sales of individual stock items in each store*" clearly specifies the *time*, *stock* and *store* dimensions, as well as the *sales amount* fact. Realizing dimensions and facts (and their correct association) within user requirements is a leading issue in data warehouse specification, as well as an error-prone aspect;

iii) ***Summarizability* assurance** – a functionally strong requirement of data warehouse specification is to guarantee the correctness of aggregation results, also known as *summarizability*. Problems arise when attributes in a *product* dimension like "video system" or "water usage" are not valid for all products. Instead, they depend on a specific context (*washers are products but do not have a video system, whereas video articles have a video system, but no water usage property*). Drawbacks can be avoided by clearly expressing the constraint requirements regarding aggregation of data, as well as conforming *Data Mart* facts and dimensions to the enterprise data warehouse model;

iv) **Represent integration with data sources** – in a data warehouse environment, data is collected from several different sources, inside and outside the enterprise environment. This activity involves not only importing data from all sort of bases, but also uncovering informal business requirements. Understanding requirements and procedures concerning integration with data sources is essential for designing a quality decision-support system, as well as it guarantees that application results do not face inconsistency;

v) **Fast track of user requirements changes** – an issue of concern in conventional systems development, keeping track of changes in system requirements assumes an exponential importance to data warehouse applications. Even sometimes so-called "insignificant

changes" can put to test the overall validity of derived data, weakening the decision-support potential of the application;

vi) **High-quality documentation** – Unlike conventional systems, whose documentation sometimes needs to be totally raised up "from the scratch", developing a data warehouse always involves developers using pre-existing operational information to define data integration and querying procedures. Apart from rare exceptions, such legacy documentation lacks the necessary quality, turning the extraction of technical requirements from such a material a cumbersome activity. Hence, a high-level documentation designed to overcome this situation, providing a general interface for requirements exploitation of information sources, is extremely required.

5. Methodology

The methodology hereby presented is structured in a set of phases (***Figure 2***). Each phase follows the abstraction level of the application in depth, as the project requirements are gathered to form a requirements *baseline*. Surrounding this process stands a backbone activity named *Requirements Management Control*, which performs permanent quality assessment of requirements changes in background.

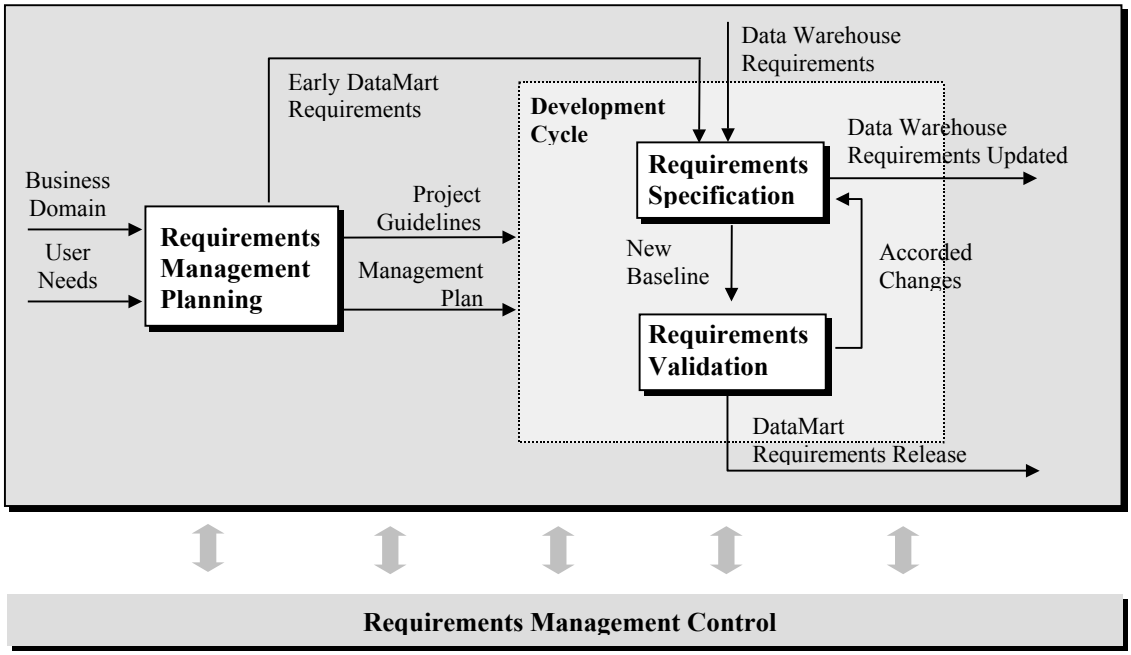

Figure 2. Framework

Coupled with the above architecture, serving as recording instruments of the facts and cutting-edge points for the system development, stands a set of document *templates* proposing a pre-defined structure for requirements documentation, suitable for registering all aspects of stakeholders' needs, and conceiving both the dimensional schema, and the data warehouse specification. In the sequel we provide descriptions of each methodology phase.

5.1 Requirements Management Planning

Before eliciting requirements, rules for an effective requirements management process must be defined. Such a policy encompasses general guidelines that will guide the appropriate application of the methodology. These guidelines concern the acquisition, documentation and control of selected requirements, and can be defined in terms of business rules, procedures and processes commonly agreed to clarify the following aspects:

(1) **Project Objectives.** Due to general flexibility provided by the dimensional model and OLAP tools, users might be tempted to see every possible desire as achievable. On the other hand, developers tend to see the data warehouse construction through an eminently technical point of view, designing functionalities that drift apart from user's real needs. A balance can be achieved by clearly stating the project objectives among all participants before it gets started.

(2) **Dimensional Requirements Focus.** What is the granularity scope in each *Data Mart*? What legal constraints restrict multidimensional analysis of data? In which ways would users like to have data summarized along dimensions? The answers to these and many other important questions will rule the dimensional modeling, and therefore must be carefully depicted as general project objectives.

(3) **Source Integration Premises.** One must define clear rules for data exchanging between systems, centered in a standard integration layout. *Periodicity*, *data loading priorities* and *responsibilities*, among other issues, will constitute the core of integration guidelines.

(4) **Project Schedule and Management.** Stakeholders and the Development Team are responsible for setting up statements that will undoubtedly point out the project constraints with regard to deployment time, management evaluation, and project boundaries. These statements will furthermore be integrated into the Requirements Management Plan.

The earlier issues do not imply an exhaustive list, to which there might certainly be other important statements to be added up. Yet, the resulting Requirements Management Plan describes (a) how system requirements are structured within the data warehouse project, (b) which artifacts hold the requirements specification, (c) what conditions have been established to manage the development cycle, and (d) the set of characteristics needed to assure traceability over requirements.

5.2 Requirements Specification

The success of the requirements engineering process depends on the ability to proceed from informal, fuzzy individual statements of requirements to a formal specification that is understood and agreed by all stakeholders [14]. In data warehouse development, this process underpins a cyclic approach of acquisition, representation and evaluation of requirements to gradually yield a project specification. Thus, an iterative process seems to be more appropriate to support such working flow. The initially elicited (raw) *Data Mart* requirements traverse a sequence of iterations in a spiral model (***Figure 3***), along which requirements are analyzed, negotiated within process participants, registered and conformed to a broader data warehouse specification. The product of each iteration is 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 that truly reflects end user's perceptions about the target system. As more

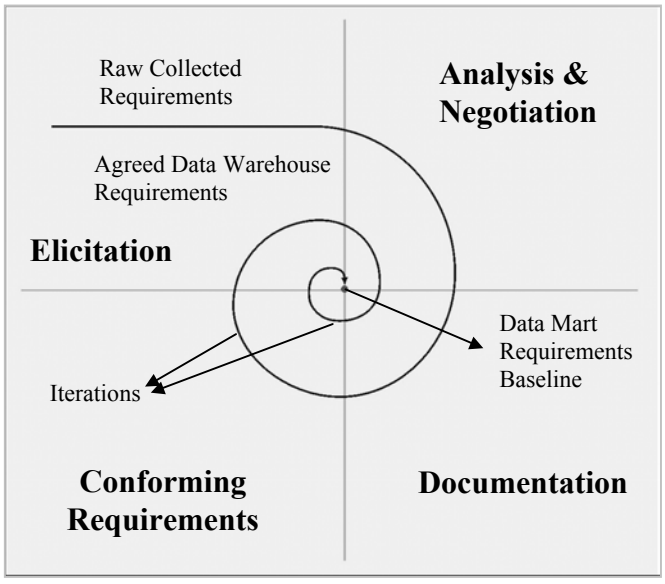

Figure 3. Requirements Specification Process

increasingly refined information is fed back from a previous iteration, the following iteration tends to be faster and more easy-solving, its product being closer to an agreed requirements baseline.

Modeling data warehouse systems requires an extra concern with the reuse of earlier agreed requirements. Because building the whole enterprise data warehouse entails a long term process of developing and integrating individual *Data Marts*, the only way to achieve such integration is to specify common factual and dimensional requirements in a way that they mean the same thing across all *Data Marts*. Therefore, recently specified requirements must conform to the overall data warehouse requirements set to avoid redundancy and guarantee adherence to the bigger model. Similar rationale can be performed to procedures and business rules within each *Data Mart* construction stage. The proper requirements feedback and conformance at this point is provided by the **Conforming Requirements** phase.

The following sections describe the (sub)processes that compound the Requirements Specification activity.

5.3 Requirements Elicitation

This phase aims to implement a process of multidimensional requirements discovering by communicating with stakeholders. As for conventional systems, the data warehouse elicitation phase requires application domain, and organizational expertise from both users and systems analysts. We found the following techniques the most useful for eliciting data warehouse requirements:

- **Interviews.** Ask questions to stakeholders regarding the strategical analysis that are to be performed, seeking to understand their real needs while taking notes of the obtained answers. This proceeding is specially indicated to soothe users' natural inability to describe in concrete terms their strategic needs. [7] illustrates innumerous examples of interview sessions suitable for extracting information from stakeholder representatives, while [13] present good tips on how to plan for the interview;
- **Prototyping.** Used within the process as an experimental system, prototypes show stakeholders how system facilities will aid in decision-making. The prototype simulates system behavior, and offers stakeholders a valuable opportunity to consolidate their ideas about system requirements, especially those associated with (architectural) multidimensional aspects.
- **Scenarios.** Working out a set of interaction scenarios aids developers in clarifying and detailing system requirements, in a use case fashion [12]. Considering that the main transactions compounding a data warehouse application tend to be identical from project to project (namely *extracting*, *integrating* and *accessing* data), its use case model will focus on specifying the business rules, procedures and querying functionalities that make one project unlike the other. Additionally, use cases enable the reuse of behavior shared among different *Data Mart* scenarios.

To all the above techniques, domain experts must work in strict accordance with requirements engineering, so that not only the necessary but also the correct information is collected.

5.4 Requirements Analysis and Negotiation

Recently discovered requirements, together with those from earlier negotiation phases are input to a thorough analysis that aims to check requirements against omissions, conflicts, overlaps and inconsistencies. Generated documents must be revised for ensuring that

specification follows quality standards and major multidimensional constraints, as well as it holds adequate balance between architectural and conceptual issues.

One important technique applied to support this phase is the *Requirements Checklist*. By checklist we imply a list of questions driven to assess each requirement through reading the requirements documents. The checklist can be implemented as a spreadsheet where the rows are labeled with the requirements identifiers. Our proposed checklist for data warehouse requirements is presented below:

Item	Description
Automatic Aggregation	Do all dimensional levels lead to a complete automatic aggregation approach, in terms of the multidimensional model elaborated?
Facts and Dimensions representation	Are all stakeholders' analytical needs represented in terms of a multidimensional schema?
Facts and Dimensions Connection	Is the entire set of dimensional levels properly associated in all levels to the basic set of facts being analyzed?
Integration Completeness	Are all integration requirements and procedures defined as to correctly incorporate external information into the system?
Documentation Quality	Do all defined documents serve as tools to accomplish all user needs under established quality standards?
Unnecessary Requirements	Do requirements correspond to a user need or serve only as a cosmetic addition to the system?
Requirements Ambiguity	Is there ambiguity within requirements, i.e., could any requirement be read in different ways by different people? What are the possible interpretations of the requirement?
Requirements Testability	Are the requirements testable, that is, are they stated in such a way that tests can be derived, to show that the system meets user requirements?
Requirements Conformity	Can we truly "drill" across fact tables by navigating through conformed dimensions without incurring in data loss or inconsistence?

5.5 Requirements Documentation

This phase is at the core of our methodology. The purpose here is to provide a complete, detailed documentation of the elicited system requirements, in such a way that they become understandable to all stakeholders. In fact, requirements documentation is produced throughout the data warehouse development, not being restricted to the documentation phase (see ***Figure 4***). In our approach we have designed a set of templates to accommodate all data warehouse functional and non-functional requirements. The templates are meta-documents that describe their own purpose and appropriate usage, like the example shown in the *Appendix*. Following, we present a brief description of each artifact:

- **Requirements Management Plan.** Documents management aspects essential to project regulation, as discussed in Section 5.1 (*see Appendix*).
- **Project Glossary.** Collects and organizes all terminology and concepts specific to the problem domains envisioned in the overall data warehouse project, thus enhancing the common understanding of basic terms among all involved parties.
- **Data Warehouse Vision.** Describes enterprise data warehouse requirements, including descriptions of the motivation and problem issues, conformed dimension and facts,

project aims and guidelines, stakeholders' profile, and other general issues concerning the data warehouse.

- **Data Mart Vision.** Collects high-level Data Mart user needs, as well as the features and actors that support such needs, under a multidimensional viewpoint. Moreover, the document tackles other important aspects related to Data Mart implementation, such as internal dimensions and procedures to be attained at each construction stage.
- **Data Mart Use Cases.** Detail the procedures required to implement all functionalities designed for each Data Mart construction phase, representing the possible sequences that might happen until the final result is achieved.
- **Data Warehouse NFR.** Complements the Use Case documents, describing all non-functional requirements not covered by the use case model, as well as design constraints and other restrictive factors.
- **Data Warehouse Rules.** The aim of this document is to register all business rules that constrain the enterprise data warehouse (including its constituent Data Marts) conception. Additionally, according to the volume of rules, distinct subject-oriented documents can be produced, each one for a set of Data Marts' rules.
- **Revision Report.** A simple report to hold the actions agreed after a requirements validation session (*see section 5.7*).
- **Traceability Matrices.** Help managing requirements changes, as stated in section 5.8.

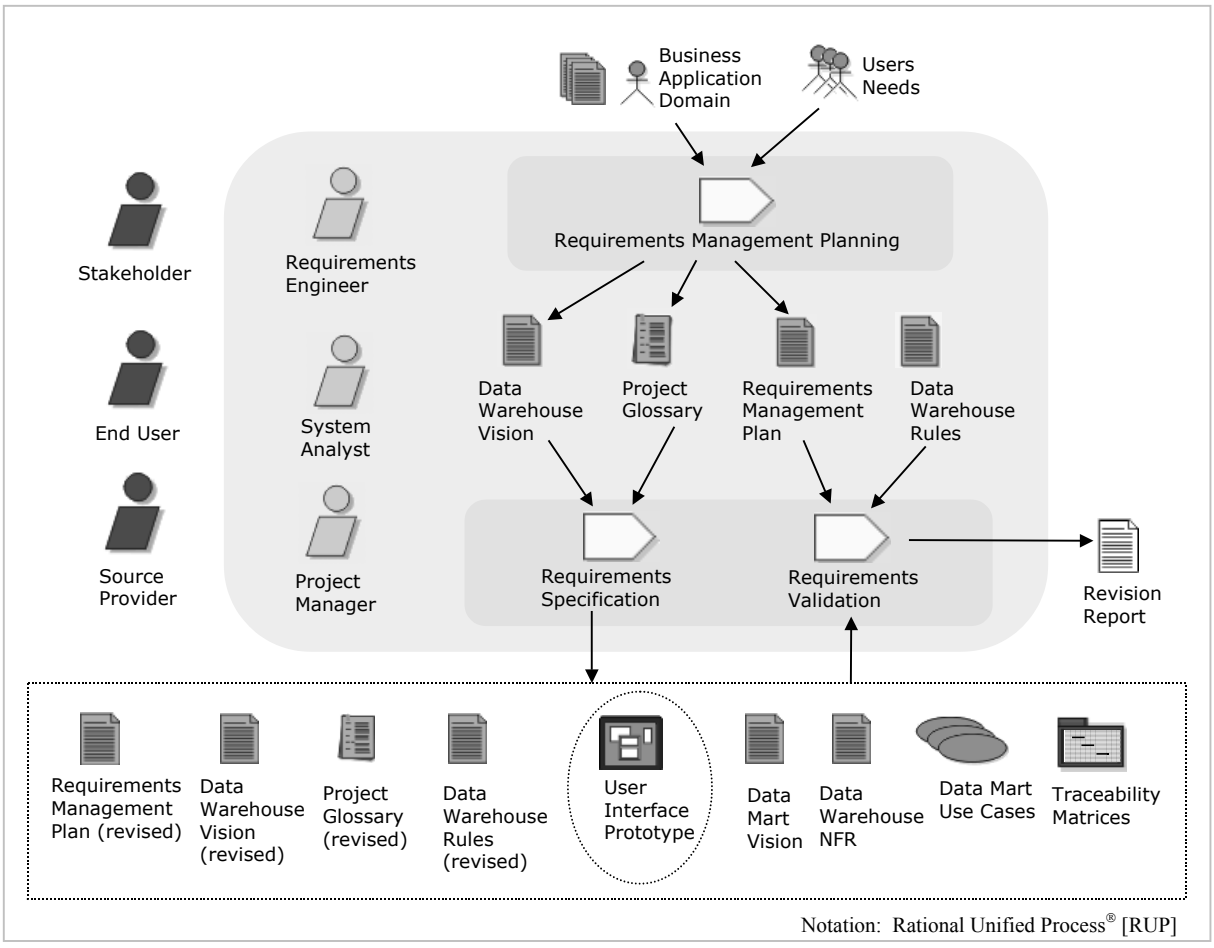

Figure 4. The Documentation Process.

5.6 Conforming Requirements

Data warehouse projects can only be successful if its definition pieces are sound in two interleaved dimensions: the subject-driven *Data Mart* vision and the global enterprise data warehouse framework. Any attempt to define isolated pieces of the data warehouse that, afterwards, cannot usefully be tied together, will cause the project to fail. According to [13], if one hopes to build a data warehouse that is robust and resilient in the face of continuously evolving requirements, one must adhere to a *Data Mart* definition on which common dimensions and facts are conformed among all *Data Marts*. A dimension is said to be "conformed" when it means the same thing to every fact table it is attached to. Similarly, a fact conforms to the overall enterprise model if the same terminology is used across Data Marts to represent its content.

In our approach, this concept is extended to an utmost level of abstraction where all common system requirements are conformed. A requirement is conformed if it is identically the same in each *Data Mart* vision of the enterprise data warehouse. More than just multidimensional aspects, conformed requirements respond for every function, characteristic or constraint to the system development that holds the same reasoning all over the project, and therefore must be represented in a unique form. In other terms, conforming requirements is one of the multiple faces of requirements reuse. Conformed requirements bring the following benefits to the data warehouse specification:

- **a.** Avoid redundancy and ambiguity between requirements that oversees the entire data warehouse;
- **b.** Allow common dimensional aspects such as dimension tables to be applied to multiple facts in the same database space;
- **c.** In conjunction with a scenario-based approach, promote reusability of agreed knowledge in the project, thus enhancing quality;
- **d.** Improve consistency of user interfaces and data content whenever the conformed model is used;
- **e.** Enable drill-across[1] operations between Data Marts;
- **f.** Guarantee the required integration among Data Marts, thus enabling the enterprise multidimensional architecture to work as a whole;
- **g.** Make data warehouse evolution a much easier task;
- **h.** Facilitates adherence to design and organizational standards;

It is a major responsibility of the data warehouse design team to establish, publish, maintain, and enforce requirements conformance. Following every documentation phase, all specification documents must be analyzed to carve off those requirements that represent enterprise data warehouse specificities. Analysts must attempt to recognize requirements overlapping and similarities, and proceed to adjustments in the specification to conform these requirements, even promoting requirements to a higher-level documentation artifact if need be.

5.7 Requirements Validation

Considering a data warehouse application, it is likely that, after completing all the previously defined phases, some misunderstandings and/or misconceptions regarding the analytical features to be provided might still remain. Sometimes, neither the user group nor the development team is confident enough of what the product being delivered is capable of

[1] OLAP operation related to retrieving facts from diverse data warehouse visions linked by common dimensions.

doing. *Review* sessions, together with prototyping, prove to be an effective strategy to detect and remove defects in the target application, before they become part of the delivered *Data Mart* package. During the review meeting, the *Data Mart* final release is presented to all involved parties, and described in terms of its full multidimensionality.

When problems are located, the validation team must immediately attach a list of *actions* in response to each problem, and agree with the actions to be enrolled. The development process returns to the specification phase where the actions are applied to conform the requirements documents to the right specification. We argue that it is extremely desirable to include in this phase external domain experts who have not been involved in the process of requirements specification. These external reviewers bring a fresh perspective into the project environment, as they are not bound to preconceived notions about the solution.

5.8 Requirements Management Control

Requirements cannot be managed effectively without *traceability*. In data warehouse systems development, traceability and change management must be carried out in both *requirements* and architectural spheres. The former investigation field will be concerned with managing changes to agreed requirements, and its impact to other requirements in the same or external document. The later will complementarily extend this investigation to the database architecture, in order to clear up what impact the underlying change will have in the multidimensional schema. Supporting tools exist for both investigation activities ([17], [5], [2], [20]). The usage of such instruments becomes mandatory in data warehouse applications for their development process involves handling large amounts of requirements and database attributes.

When tracing requirements, *Traceability Matrices* are the largest used component in special-purpose tools to show requirements dependencies. Rows and columns of the matrix represent system requirements. By reading across a row, all requirements on which a specific requirement depends are shown. A group of different matrices must be defined to support a complete requirements analysis (ex. user needs *versus* features; features *versus* facts; facts *versus* dimensional attributes; business rules *versus* use case steps; and many more). Discussion groups, requirements version control, web publishing services, and report generation facilities are among a large range of additional techniques made available by these tools. On the other hand, changes in the requirements model affect straightforwardly the database model in both *Data Mart* and Data Warehouse views, which requires the overall impacts to be traced. CASE tools can be very supportive in accomplishing this task, as they offer automated search capabilities to trace the correspondent dimensional requirement in the database, and discover how many (and at what extent) database components are affected by the change. These features reduce the effort of evaluating and performing the necessary maintenance to database table fields, thus preserving the safety of the solution.

6. Case Study

In this section we introduce SAFE[2], a decision-support system developed by SERPRO, under the premises of our methodology. SAFE collects and stores client's 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 holds real world facts vital to the analysis scenario envisioned for the subject. Attached to each fact table, a collection of dimension tables compounds star-schema models, which are

[2] SAFE is an acronym to "*Sistema de Análises Fiscais Estratégicas*" (Internal Revenues Strategic Information System). Due to strong secrecy conditions imposed by our contracted client, we will focus our description on the system's general aspects.

interconnected through their common dimensions in a complex data warehouse structure, well known in the domain literature as *constellation* [16]. The system is built on a three-tiered architecture. The application server level bridges *Java*-based interface instances and the multidimensional repository in between, enabling end users to operate the data warehouse from whichever point across client's intranet, on a "24 X 7" basis.

In developing SAFE, a software engineering team was composed of two requirements engineers and a group of eight developers (including the project leader). The chosen strategy rested on capturing critical requirements, i.e., those tightly connected to the system bottlenecks: (a) integration process definition, (b) multidimensionality mapping, and (c) user needs change control. Then, our approach was to intensively use the processes and artifacts specified in the methodology to fight back bottleneck points, with special care to requirements specification and management. On average, two iterations were followed for each *Data Mart* development in one-year trial. Following we summarize the lessons acquired during the trial at each methodological phase:

- ***Requirements Management Planning.*** The Plan helped engineers in capturing the system requirements attributes, from which classification and (mainly) prioritization became possible. The phase wrapped clients into the novel experience of discussing the data warehouse inner aspects, objectives and construction guidelines in equality with the technical team. Such an experience turned out to be a critical step towards avoiding misconceptions about the product that, in other projects, had been postponed to deal with in later phases, inevitably incurring in schedule and cost overhead. At the end, not only the clients' confidence in the project rose high, but also the so-called gap between client (from now on, *stakeholder*) and developer was extremely reduced. The phase also revealed the need for a glossary of terms, considering the wide range of business domains tackled by the data warehouse.
- ***Requirements Specification.*** Interviews conducted during this phase enabled a high-level information exchange among stakeholders, to whom evolving prototyped versions added a broader understanding about the data warehouse capabilities. Likewise, a detailed use case definition stage allowed engineers to keep focus on real user needs, and hence implement functionalities to attain at such needs. All gathered requirements were first cataloged onto instances of the document templates, and later loaded into a RequisitePro® requirements database, from which they could be easier managed by the project leader. Comparing to earlier experiences in developing data warehouse solutions, engineering team experts proved the efficiency of this new approach in correctly representing and conforming multidimensional elements, integration requirements, and summarizability constraints. Two types of documents were highlighted in the process: the *Vision Documents*, providing sections to neatly register the interleaved association among external sources, facts and dimensions; and the *Traceability Matrices* automated by RequisitePro® to facilitate management activities that, without these instruments, would be considerably harder for the project leader to achieve. A distinct product of applying our methodology was the generation of a conformed structure of dimensions (*periodicity, organizational level, tax-payer, tax status, among others*) and facts (*collected amounts, tax payment quantifiers, internal revenues amounts, and others*) to serve as a backbone for new *Data Mart* implementations. Similarly, quality assurance procedures were generalized and applied to the entire data warehouse development. Iteratively documenting requirements also generated an efficient basis for future consulting and validation. Developers realized that, by doing so, they established means for contractual agreement with the clients so that no specification would undergo implementation without client's approval.
- ***Requirements Validation.*** We note here that clients added a new perspective to the documents validation, promoting the scheduled review sessions to open *Workshops*, where

requirements documents were discussed among all involved parties, including external observers, as suggested in our approach. We also note that observers contributed with unbiased questions that pointed out misconceived (or even forgotten) conceptual and architectural aspects, which reinforced the importance of such elements in this phase. Furthermore, information was validated among all parts so that what had been specified corresponded to what was meant to be implemented. The requirements documents were then updated and action reports produced to be managed by the project leader.

- ***Requirements Management Control*** – The core stages of our methodology generated a total amount of 4.306 requirements of all sorts. As previously argued in this work, such amount would result unmanageable without full-automated support. RequisitePro® facilities came to rescue in this task, allowing the project leader to evaluate the impacts of requirements changes to the project, and thus better negotiate the change with the client. On the architectural side, impacts were analyzed using the Oracle's Designer 2000®, the project's official CASE tool, and the results joined up to enrich discussions with the client.

Applying the methodology operated a deep change on the client-developer paradigm. Clients felt like being part of the construction process for the first time, which helped in the development of an agreed solution. Moreover, the use of requirements management facilities increased clients' perception of requirements changing consequences, and the impact of continuously changing needs to the project schedule. On the other hand, developers were now aware of requirements reusability during the development of a data warehouse, looking for preserving conformance to the enterprise broader model. In spite of the benefits proved, the trial also revealed some weaknesses in the methodology:

- The large amount of requirements caused traceability matrices to become hard to manage or visualize, as the associations between requirements grew fast and sparse. Higher-leveled requirements attributes were introduced to more coarsely relate requirements in each matrix, but that did not keep the project leader from executing partial analysis, in a number of steps, to completely estimate the impact of requirements changes.
- More than analyzing the impact of changes, one felt the need of means for controlling the client's requests for changing workflow. Processes or instruments to use in support of this activity are not defined in our present approach.
- The methodology does not treat maintenance projects.
- Considering that SERPRO holds an ongoing process to adhere to SEI's CMM-Level2, some consideration about the links between the proposed Requirements Engineering approach and other CMM Key Process Areas are missing.

Such weak points will be object of specific studies in upcoming versions. For the moment, however, a final analysis indicates the use of the methodology to aid in the development of decision-support systems, as demonstrated by the achievement of project schedule, small incidence of errors, multidimensional characteristics coverage, and a sound client's satisfaction with the quality obtained in the delivered product.

7. Conclusions

In this paper we have proposed a methodology for requirements analysis of data warehouse systems, as an important step towards delivering a quality decision-support solution. When developing a data warehouse, we must keep track of its inherent multidimensionality, a complex aspect that makes the development of such systems rather different than in conventional ones. In addition, data warehouse applications belong to a

modern class of systems to which architectural and specification issues have equal status. The key to address the later aspects is a reliable requirements engineering process, which in this case claims for a methodological support based on iterative, yet domain-driven steps. Our approach provides such support with innovative undertaking on documentation, management and reuse of requirements.

As demonstrated in the case study, the proposed methodology represents a powerful tool in analyzing and managing data warehouse requirements. It provides means for capturing users needs and domain knowledge over the analytical problem, while drives development towards creating an integrated multidimensional solution. Additionally, we presented the overall benefits in using such methodology as a guide throughout the data warehouse development, proving that it helps stakeholders to expand and consolidate their knowledge along with the own evolution of the project.

Among our future works, we intend to update and release new methodology versions based on the lessons acquired in present and forthcoming trials. We have just started researches towards defining a framework to largely support reuse of requirements and other components of pre-existent data warehouse solutions, using domain engineering techniques.

8. References

[1] 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.

[2] Oracle 9*i*Discoverer. http://otn.oracle.com/products/discoverer/content.html

[3] Boehnlein, M., Ende, A. U. "Deriving Initial Data Warehouses Structures from the Conceptual Data Models of the Underlying Operational Information Systems". *Proc. of Workshop on Data Warehousing and OLAP* (DOLAP), Kansas City, MO, USA, 1999.

[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] Telelogic DOORS®/ERS. http://www.telelogic.com/products/doorsers/index.cfm

[6] Firestone, J.M. "Object-Oriented Data Warehousing". Technical Report, Executive Information Systems White Paper No.5, 1997.

[7] Giovinazzo, W. *Object-Oriented Data Warehouse Design*, Prentice Hall, 1st edition, February, 2000.

[8] Golfarelli, M., Rizzi, S. "Designing the Data Warehouse : Key Steps and Crucial Issues". Journal of Computer Science and Information Management, Vol. 2, No. 3, 1999.

[9] Hüsseman, B., Lechtenbörger, J., Vossen, G. "Conceptual Data Warehouse Design". *Proc. of Intl. Workshop on Design and Management of Data Warehouses* (DMDW2000), Stockholm, Sweden, June, 2000.

[10] Hahn, K., Sapia, C., Blaschka, M. "Automatically Generating OLAP Schema from Conceptual Graphical Models". *Proc. of Workshop on Data Warehousing and OLAP* (DOLAP), Washington DC, USA, 2000.

[11] Inmon, W. H. *Building the Data Warehouse*, John Wiley & Sons, 2nd edition, 1996.

[12] Jacobson, I. *Object-Oriented Software Engineering*, Addison-Wesley, 1992.

[13] Kimball, R., Reeves, L., Ross, M., Thornthwaite, W. *The Data Warehouse Lifecycle Toolkit*, New York, John Wiley & Sons, 1998.

[14] Loucopoulos, P., Karakostas, V. *System Requirements Engineering*, McGraw-Hill, London, 1995.

[15] 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.

[16] Raden, N. “Star Scheme 101”. http://members.aol.com/nraden.str.htm
[17] Rational RequisitePro. http://www.rational.com/products/reqpro/docs/datasheet.html
[18] Rational Unified Process: Artifacts Notation. www.rational.com/products/rup/index.jsp
[19] Sommerville, I., Kotonya, G. *Requirements Engineering: Processes and Techniques*, Addison-Wiley, 1997.
[20] Popkin’s System Architect. http://www.popkin.com/
[21] Zahran, S. *Software Process Improvement*, Addison-Wiley, 1998.
[22] Mylopoulos, J., Chung, L., Liao, S., Wang, H., Yu, E. “Exploring Alternatives During Requirements Analysis”, IEEE Software, January/February, 2001, pp 2-6.
[23] Finkelstein, A., Kramer, J., Nuseibeh, B., Goedicke, M. “Viewpoints: A Framework for Integrating Multiple Perspectives in Systems Development”, International Journal of Software Engineering and Knowledge Engineering, 2(10): 31-58, 1992.
[24] Chung, L., Nixon, B., Yu, E., Mylopoulos, J. *Non-Functional Requirements in Software Engineering*, Kluwer Publishing, 2000.
[25] van Lamsweerde, A. “Requirements Engineering in the year 00: A Research Perspective”. Invited Paper to ICSE’2000, *Proc. of the International Conference on Software Engineering*, Limerick, June, 2000.

9. Appendix – *Example of Requirement Document Template*

Revision History

Date	Version	Description	Author	Revised by

Requirements Management Plan

1. **Roles and Responsibilities**
   <identifies the roles and respective responsibilities of those involved in managing the data warehouse development process.>

2. **Procedures and Processes**
   <describes the processes and procedures to be applied when managing requirement changes, validating data transformation, and deploying user interface facilities. These regulations must define behavior to be followed by the roles (users and developers) involved in each stage of the application development. The list of processes/procedures should be appropriately organized into sections regarding each stage name, in order to improve readability. The most common stage labels are Data Extraction, Data Transformation, Database Loading, and User Analysis Generation, but modern stages of concern nowadays can also be included such as Data Updating and Securing. This section should also include details on how the Requirements Management Plan itself is to be updated, in a "Project Management" section.>

3. **Artifacts**
   < enumerates and briefly describes the project artifacts that hold the requirements definition.>

4. **Requirements Identification**
   <describes how requirements items are to be classified, marked, and numbered.>

   4.1. **Types**
   <specifies all functional and non-functional requirements types associated with the project. Common types are Dimension, Fact, Actor, User Need, Feature, Use Case Name, Constraint, and many more. To facilitate classification, a specific mnemonic must be created to identify each requirement type (ex.: DIM – Dimension).>

   4.2. **Attributes**
   <lists the requirements attributes to be used in evaluating, accompanying, prioritizing and managing the underlined requirements. Useful attributes are Risk, Effort, Priority, Status, Benefit and Aggregation Level. Each attribute must be described in terms of a classification range, from a lower importance level to a higher one (ex.: Priority can be classified into High, Medium and Low). A brief description must follow each level.>

   4.3. **Identification Rule**
   <specifies the numbering rule to be considered for identifying requirements.>

5. **Traceability Criteria**
   *<describes any additional rules and guidelines with regard to requirements traceability. Applicable constraints such as "*every approved dimensions must be linked to at least one fact table*" must be described in this section.>*

6. **Milestones**
   <enumerates internal and costumer milestones related to the requirements management effort.>