Designing Data Marts for Data Warehouses

ANGELA BONIFATI
Politecnico di Milano
FABIANO CATTANEO
Cefriel
STEFANO CERI
Politecnico di Milano
ALFONSO FUGGETTA
Politecnico di Milano and Cefriel
and
STEFANO PARABOSCHI
Politecnico di Milano

Data warehouses are databases devoted to analytical processing. They are used to support decision-making activities in most modern business settings, when complex data sets have to be studied and analyzed. The technology for analytical processing assumes that data are presented in the form of simple data marts, consisting of a well-identified collection of facts and data analysis dimensions (star schema). Despite the wide diffusion of data warehouse technology and concepts, we still miss methods that help and guide the designer in identifying and extracting such data marts out of an enterprisewide information system, covering the upstream, requirement-driven stages of the design process. Many existing methods and tools support the activities related to the efficient implementation of data marts on top of specialized technology (such as the ROLAP or MOLAP data servers). This paper presents a method to support the identification and design of data marts. The method is based on three basic steps. A first top-down step makes it possible to elicit and consolidate user requirements and expectations. This is accomplished by exploiting a goal-oriented process based on the Goal/Question/Metric paradigm developed at the University of Maryland. Ideal data marts are derived from user requirements. The second bottom-up step extracts candidate data marts

The editorial processing for this paper was managed by Axel van Lamsweerde.
Authors' addresses: A. Bonifati, Politecnico di Milano, Piazza Leonardo da Vinci 32, I-20133 Milano, Italy; email: bonifati@elet.polimi.it; F. Cattaneo, Cefriel, Via Fucini 2, I-20133 Milano, Italy; email: cattaneo@cefriel.it; S. Ceri, Politecnico di Milano, Piazza Leonardo da Vinci 32, I-20133 Milano, Italy; email: ceri@elet.polimi.it; A. Fuggetta, Politecnico di Milano and Cefriel, Piazza Leonardo da Vinci 32, I-20133 Milano, Italy; email: fuggetta@elet.polimi.it; S. Paraboschi, Politecnico di Milano, Piazza Leonardo da Vinci 32, I-20133 Milano, Italy; email: paraboschi@elet.polimi.it.

from the conceptual schema of the information system. The final step compares ideal and candidate data marts to derive a collection of data marts that are supported by the underlying information system and maximally satisfy user requirements. The method presented in this paper has been validated in a real industrial case study concerning the business processes of a large telecommunications company.

Categories and Subject Descriptors: D.2 [**Software**]: Software Engineering; D.2.1 [**Software Engineering**]: Requirements/Specifications—*Methodologies* (e.g., object-oriented, structured); D.2.2 [**Software Engineering**]: Design Tools and Techniques

General Terms: Design, Management

Additional Key Words and Phrases: Data warehouse, data mart, design method, conceptual modeling, software quality management

1. INTRODUCTION

Data warehouses are becoming an important asset of any modern business enterprise, with classical applications in business planning and strategic analysis. For example, sales departments use data warehouses to study the buying profiles of their customers and decide their commercial and distribution strategies accordingly. The data warehouse component is a database built for analytical processing whose primary objective is to maintain and analyze historical data.

In order to standardize data analysis and enable simplified usage patterns, data warehouses are normally organized as problem-driven, small units, called "data marts"; each data mart is dedicated to the study of a specific problem. The data organization of a data mart, called a star schema, is very simple: the data being analyzed, or facts, constitute the star's center; around the center, other data describe the dimensions along which data analysis can be performed. In the archetypical case, facts are the sales of an organization, and dimensions enable the analysis by product, customer, point of sale, time of sale, and so on. In simple warehouses, data marts may extract their content directly from operational databases; in complex situations, the data warehouse architecture may be multilevel, and the data mart content may be loaded from intermediate repositories, often denoted as "operational data stores."

This paper is concerned with the design of data marts starting from a conceptual description of the enterprisewide information system, or at least of the schema fragment which is relevant for the data mart design. In our experience, many large organizations do have such a conceptual description, because even before the development of data warehouses, an integrated schema is developed to enhance data interoperability at the operational level. If an operational data store is available, we consider its conceptual description.

Despite the existence of several methods and tools addressing the problems related to the physical implementation of the data mart, or to the statistical manipulation of its data, little attention has instead been paid to the design of the data mart schema, probably assuming that the conceptual

simplicity of such a schema would make this phase a trivial problem. Actually, the definition of the schema is one of the most critical steps in the overall data warehouse development process, and there are several questions that must be correctly answered in order to adequately capture user requirements and build an effective warehouse. A few of these questions are: What are the data to be collected to address the business needs? How many dimensions should be present in order to facilitate their analysis? What is the level of granularity of the data to be stored in the data mart? In practice, these questions are approached nowadays by relying on the experience of the designer.

This paper presents a method for identifying and building data marts. The basic principle underlying the proposed approach is that the design of data marts should be driven by the business needs that each data mart is expected to address. As a consequence, the data mart design process must be based on a deep understanding of the users' expectations.

The method proposed in this paper consists of three basic steps: top-down analysis, bottom-up analysis, and integration. This method takes advantage of the two complementary perspectives of data mart design: top-down analysis emphasizes the user requirements, while bottom-up analysis brings to the surface the semantics of the existing operational databases. The final step integrates these two viewpoints, and thus generates a feasible solution (i.e., supported by the existing data sources) that best reflects the user's goal.

—In the first step, user requirements are collected through interviews. The purpose of interviews is to gather information from business analysts and/or managers about the company goals and needs. This is accomplished by expressing user expectations through the Goal/Question/Metrics paradigm [Basili et al. 1994]. The goals obtained are aggregated and refined, until a small number of goals subsume all the collected information. Eventually, each aggregated goal is specified by means of an *abstraction sheet*, which expresses the goal characteristics in great detail. From abstraction sheets, it is possible to extract the specifications of ideal star schemas, i.e., fragments of the conceptual schema of the data warehouse that reflect the user requirements in a way independent of the availability of the underlying data. This phase is focused on the analysis of functional requirements, which correspond to entities in the conceptual modeling. Other nonfunctional requirements, such as cost, performance, reliability, maintainability, and robustness of the software system, for example, are not considered, since they are mostly relevant to measure the quality of the system, but not to guide the abstract task of conceptual design.

—The second step of the proposed method is devoted to examining the conceptual model of the operational databases, finding out candidate star schemas for the data warehouse. The approach is based on an exhaustive graph analysis technique that exploits an entity-relationship representation of the conceptual schema of the operational databases: from this

analysis, the candidate star schemas are structured as an aggregation of data around "additive" data items. Therefore, this step constitutes a sort of bottom-up approach to the definition of all the possible data marts. Although this step can generate a large number of candidates, the analysis can be executed in a fully automatic manner.

—The third step matches each ideal star schema determined by the first step with all the candidate star schemas produced by the second step, and ranks them according to a set of metrics. Then, it is the designer's responsibility to choose the candidates that best fit the ideal schema.

The proposed design method has been conceived and then validated through real industrial experiments conducted by the authors during the past couple of years. In particular, in this paper we will consider the processes related to *service assurance*, *service delivery*, and *network creation* of a large telecommunications company. The main needs of the process owners are to study and analyze cost, duration, effort, performance, and quality perceived by customers.

The paper is organized in eight sections. Section 2 relates the proposed approach to the state of the art in the field. Section 3 provides a brief presentation of the case study that will be used in the paper to illustrate and discuss the different steps in the method. Section 4 discusses the top-down phase by which an ideal star schema is derived from users' requirements. Section 5 introduces the analysis techniques used to extract candidate star schemas from the operational database schema. Section 6 presents the integration of the two techniques to derive the best-fit data marts. Section 7 gives a summary description of the proposed method. Finally, Section 8 draws some conclusions and outlines future research work.

2. STATE OF THE ART IN DATA WAREHOUSE DESIGN

2.1 Basic Concepts in Data Warehouse Technology

Operational databases offer an integrated view of the application, so that each entity of the real world is mapped to exactly one concept of the schema. Therefore, a complex reality is associated to complex schemas, which aim at capturing the complexity of the application domain. Schemas are often represented at an abstract level through an entity-relationship data model.

In contrast, data warehouses often master the complexity by offering a vision where data are split in a number of simple schemas, called *data marts*, each one specialized for a particular analysis activity. Each data mart, in turn, represents data by means of a *star schema*, consisting of a large *fact table* as center and a set of smaller *dimension tables* placed in a radial pattern around the fact. The fact table contains numerical or additive measurements used to compute quantitative answers about the business. The dimension tables give the complete descriptions of the dimensions of the business.

Each fact table has a primary key that is composed of the keys of all the dimensions; this establishes a one-to-many mapping from each dimension tuple to the (many) fact tuples which share the same value as the dimension's key. Furthermore, in a star schema the fact table is at least in Boyce-Codd normal form, while the dimension tables are not normalized. Therefore, data inside a dimension may be functionally dependent on other data, and thus redundant. By applying the classical normalization process, each dimension may be turned into a hierarchy of tables, with a one-to-many mapping for each functional decomposition. This yields a more complex schema, called a *snowflake schema*, where each normalized dimension corresponds to a hierarchy of tables. This approach makes it possible to reduce data replication, but the schema is slightly more complicated than the pure star schema.

Data marts can be simple because each of them is focused on a single aspect over the operational data [Kimball 1996]. Suitable mechanisms (called *replication managers*) populate the data marts from the operational databases. Therefore, there can be multiple, possibly overlapping, data marts over the same operational data; this situation may be efficiently managed by hierarchical warehouse architectures, in which data are initially loaded into an operational data store and subsequently loaded into data marts. The arbitrarily complex semantic relationships among data are managed at the operational level, while they are reduced to much simpler abstractions at the data warehouse level. The *data warehouse schema design problem* consists then in determining a suitable collection of star schemas (or, with slightly higher complexity, of snowflake schemas) that enable the analysis activity and can be efficiently supported by the schema of the operational database.

To assess the feasibility of implementing a data warehouse schema on top of a specific operational database, it is necessary to study the underlying database and check whether the data required to create the data warehouse are indeed available. This means that there must be some description of the schema of the operational database, possibly expressed through an entity-relationship diagram; this is the only prerequisite to the applicability of our method. When the operational database is implemented across several independent legacy systems, determining an integrated entity-relationship schema of its contents may indeed be difficult. Traditional techniques of *view integration* and of *reverse engineering* can be applied to this purpose (see, for example, Batini and Lenzerini [1984] and Francalanci and Fuggetta [1997]); these are outside of the scope of this paper.

2.2 Related Work

An overview of current data warehousing and OLAP technology and related research issues on multidimensional databases can be found in several articles and books [Chaudhuri and Dayal 1997; Inmon 1996; Jarke et al. 1999; Kimball 1996; Widom 1995]. A comprehensive on-line bibliography on

this subject has been published by Mendelzon (available via http://www.cs.toronto.edu/~mendel/dwbib.html).

An influential book, particularly from the practitioners' viewpoint, is Kimball [1996], which discusses the major issues arising in the design and implementation of data warehouses. The book presents a case-based approach to data mart design and suggests that each data mart should be independently designed. The design process emphasizes the crucial importance of interviewing the end users: the interviews introduce the data warehouse team to the company's core business and deepen the data warehouse designer's understanding of users' expectations. Furthermore, the book discusses the suitability of mixing end-user interviews with database administrator interviews, in order to collect information about the feasibility and costs related to the gathering of data which are needed to meet each of these expectations. Therefore, the book sets up a general framework that is consistent with our method and that we found applicable in concrete experiences.

A significant amount of research in the database community has been dedicated to the physical and logical design of data warehouses. Some of the proposed approaches are related to the integration of heterogeneous sources and the selection of materialized views. Other approaches deal with extending the multidimensional data model [Calvanese et al. 1998a; 1998b; Ezeife 1997; Garcia-Molina et al. 1998; Gupta and Mumick 1999; Harinarayan et al. 1996; Lehner 1998; Theodoratos and Sellis 1998; 1999; Wiener et al. 1997]. However, the issue of conceptual schema design is considered only in few recent works. A framework for multidimensional database design has been proposed by Agrawal et al. [1997], outlining a data model based on one or more hypercubes and a set of basic operations designed to operate on it. A Multi-Dimensional Data Model has been developed by Gyssens and Lakshmanan [1997], which aims at identifying a conceptual model and query language to support OLAP functionalities, i.e., querying, restructuring, classification and summarization. Theodoratos and Sellis [1999] proposed a theoretical framework for the logical and physical design problem. The approach is structured as a state search problem. In particular, the data warehouse is defined by selecting a set of views that fit the space available to the data warehouse, and minimize the total query evaluation and view maintenance cost.

An overall approach to modeling, design, implementation, and optimization of data warehouses has been developed as part of the "Foundations on Data Warehouse Quality" (DWQ) research project and condensed in a book [Jarke et al. 1999]. In particular, the design method is based on the development of a Conceptual Data Warehouse Data Model (CDWDM), which describes the relevant aggregated entities and dimensions of the domain [Franconi and Sattler 1999]. The CDWDM is able to capture the database schemas expressed in the Extended Entity Relationship Data Model and is based on Description Logics, offering reasoning services, e.g., implicit taxonomic links between entities. The final aim of that approach is to provide an extended modeling formalism. Our approach does not target

the formalization problem and is tailored to give methodological guidance specifically focused on data mart identification; hence we see the two approaches as complementary.

In general, we found very few contributions in the literature specifically concerned with data mart design. The approaches presented in Cabibbo and Torlone [1998] and Golfarelli et al. [1998] are perhaps the closest to ours, in particular as far as the bottom-up phase of our method is concerned.

—Cabibbo and Torlone [1998] illustrate a method for developing data marts out of conceptual operational schemas. Data marts are based on graphs entailing F-tables, i.e., tables that serve as abstract implementation of both relational databases and multidimensional arrays. The proposed design method builds an *MD* scheme starting from an underlying operational database, where an *MD* scheme consists of a finite set of dimensions, a finite set of F-tables, and a finite set of level descriptions of the dimensions. *MD* schemes are obtained through four steps: identification of facts and dimensions, restructuring of the entity-relationship schema, derivation of a dimensional graph, and finally, translation into the *MD* model, singling out the F-tables, i.e., combinations of dimension levels with the chosen fact node.

—Similarly, Golfarelli et al. [1998] propose a methodology for the transformation of an entity-relationship schema in a tree-structured *fact schema*. This model represents the fact as the tree's root and the dimension attributes as the tree's descendants. A *query pattern* corresponds to a set of markers placed on the tree nodes, and in each hierarchy, the markers indicate at which level fact instances must be aggregated.

Both above contributions are concerned with the investigation of data warehouse conceptual design starting from the underlying operational schemas (bottom-up phase). Still, they do not integrate this design phase with a goal-driven analysis of user requirements (top-down phase). In addition, they identify a single "solution," while our approach provides the designer with a set of ranked solutions.

There is a specific approach that exploits the GQM method to support data warehouse evaluation [Vassiliadi et al. 1999]. This approach exploits GQM to identify the factors and metrics that make it possible to evaluate the quality of a data warehouse, once it has been developed. Our approach uses GQM for making explicit user requirements in the upstream stage of the data warehouse design activity. Hence, the focus and final objective of the two contributions are different, even if both of them are based on the same methodological approach (i.e., GQM).

As a final observation, it is important to clarify the relationship between our approach (based on GQM) and other goal-oriented techniques developed in the past. Certainly, goal orientation has demonstrated to be particularly effective in eliciting and structuring requirements [Anton and Potts 1998]. Consequently, several goal-oriented languages and methods

have been developed and applied in practice. The goal orientation of these approaches is justified and motivated by the nature of the information that needs to be described, i.e., the expectations that users have on the functionality of the software system to be developed. For instance, KAOS [Darimont et al. 1997] makes it possible to describe "goals" and their relationships with actions, events, and constraints that characterize the application domain being observed from a functional viewpoint. GQM is used to describe different kinds of goals, i.e., those related to the evaluation of the quality aspects of the entity being observed. GQM is strongly oriented to simplifying the process by which generic goals concerning an assessment and evaluation activity are translated and articulated in specific quality factors and evaluation criteria. Therefore, even if these approaches are all based on the same concept (i.e., goal orientation), they are structured and used for different purposes.

With respect to the method proposed in this paper, a possible replacement of GQM as a goal representation technique is the approach proposed by Mylopoulos et al. [1992]. The authors propose a formal technique to structure nonfunctional requirements as a hierarchy of goals. Goals are related through different types of links such as AND/OR, goal inclusion, and positive/negative influences between goals. In our work, this approach could have been used as an alternative to GQM, since it could be used to organize the quality and variation factors of a GQM abstraction sheet. We argue, however, that for our purposes GQM makes it simpler to elicit quantitative business aspects and to convert them into data mart schema, as will be illustrated in Section 4.

3. A MOTIVATING CASE STUDY

In this paper we will use an example derived from a real industrial experiment carried out by the authors. The main needs of the company were to study and analyze cost, duration, effort, performance, and quality of significant company processes as perceived by customers. In particular, the company was interested in analyzing *service assurance* ("is the reliability and maintenance of the network coherent with the company goals?"), *service delivery* ("are customers' requests correctly managed?"), and *network creation* ("is the deployment of new services and infrastructures accomplished according to the company's strategic objectives?").

The company has two main divisions, each one devoted to a specific class of customers (i.e., business and residential). Each division is geographically structured on multiple levels (region, subregion, small city, or part of a major city). At the lowest level, each unit is composed of a maintenance work center, responsible for the maintenance process and deployment activities of the network resources located in its territory. The company comprises around 200 work centers. Typically, a work center employs around 50 technicians. Overall, this is a rich and complex environment, characterized by a great amount of data and a complex operational database schema. The data warehouse built for the case study eventually

includes a fact table and some dimension tables containing several millions of tuples.[1]

Overall, the complete schema of the operational database used for managing work centers amounts to several hundred entities and relationships, and a few thousand attributes. However, it was possible to single out "auxiliary tables," supporting applications and not really related to storing persistent data. In this way, we produced a more manageable conceptual schema, consisting of about 100 entities and relationships.

The data warehouse design project was carried out by a joint team composed of the authors (acting as consultants and methodologists) and a group of people from the Information Systems Department of the company. The project was conducted as part of a companywide initiative aiming to create new decision support systems for different management levels in the organization. Eventually, the results of our activity have been turned into a new support system that is currently used throughout the company.

4. THE TOP-DOWN PHASE

The top-down phase of the data warehouse design method presented in this paper is based on the GQM paradigm [Basili et al. 1994]. GQM has been developed at the University of Maryland as a systematic approach for the development and exploitation of a software measurement program, i.e., a set of metrics and related procedures that provide a quantitative evaluation of a specific phenomenon concerning software development [Fuggetta et al. 1998]. The basic principle underlying GQM is that there is no universal measurement program that can fit any situation or measurement goal. Rather, the metrics and the related procedures must be defined by carefully considering the specific characteristics of the measurement problem being addressed.

As discussed in the introduction, there are several similarities between designing a data warehouse and creating a measurement program. For this reason, we have borrowed and adapted several GQM ideas and concepts to create our data warehouse design method. The rest of the section illustrates GQM main concepts and how they have been used in our approach.

4.1 A Quick Overview of GQM

GQM is composed of a process model, and a set of forms and guidelines. The process model identifies the different activities that have to be carried out to create and operate a measurement program. The process starts by identifying the context (i.e., the characteristics of the software factory)

---

[2]For the sake of precision, in this case we realized that the financial data are stored in a database that is under the control of a different division of the company. Therefore, the integration of this information, even if certainly feasible from a technical viewpoint, was considered too complicated at that stage of the project for organizational reasons.

[1]The conceptual schema of the operational database was available as part of the company's assets prior to the begin of the study and was made available to us as a collection of ER subschemas created and managed through Cayenne Software Inc.'s GroundWorks.

where the measurement program has to be established. The next step is the identification of the *goals* of the measurement activity. This is accomplished by filling up two forms for each goal being identified.

(1) The first form is used to formulate the goal in abstract terms (*goal definition*). In particular, each goal is described through the following information:
   —**Object of study:** the part of reality that is being observed and studied.
   —**Purpose:** the motivations for studying the object.
   —**Quality focus:** the object characteristics that are considered in the study.
   —**Viewpoint:** the person or group of people interested in studying the object.
   —**Environment:** the application context where the study is carried out.

(2) The second form (called *abstraction sheet*) is then used to enrich the goal definition with more detailed information. In particular, the quality focus is further refined by identifying the specific characteristics to be observed and the related attributes. The additional information is provided to identify the variation factors that might affect the identified quality characteristics. Finally, the abstraction sheet lists the baseline hypotheses of the organization about the quality factors being observed (i.e., the "values" of the metrics that people have in their mind) and the estimated effect that variation factors might have on the baseline hypotheses.

Once goals have been reasonably formulated, they are compared and assessed. This may lead to collapse different goals that overlap, to eliminate goals that are not considered important or relevant, and to sort out the remaining goals according to some priority or criticality factor. All the activities from the initial formulation of goals to the final consolidation of the relevant ones are accomplished by interacting with the different people involved in the process being observed.

The portion of the GQM process that has been described so far constitutes the core of the top-down approach presented in this paper. GQM includes other phases for defining, collecting, and analyzing measures that can be used for validating goals. In the remainder of this section we will illustrate a revised GQM process and how we have applied it to the case study introduced in Section 3.

4.2 Goal Identification

The starting point in the adaptation of GQM to our problem is the consideration that data warehouse design is a goal-driven process. For this reason we use the goal definition forms of GQM to explicitly state the different goals that people have in their mind about the data warehouse to be developed.

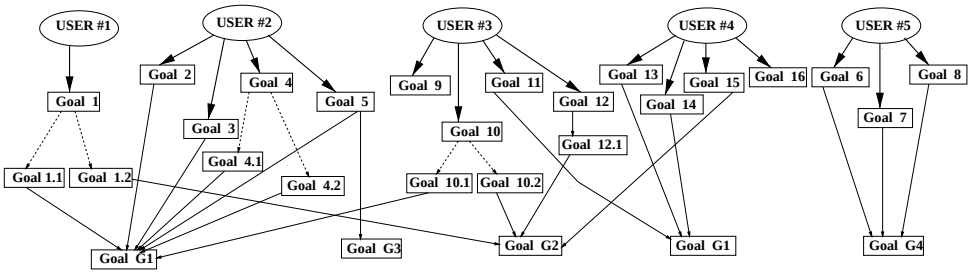


Fig. 1. Relations among interviewees and aggregated goals.

In our case study, we interviewed about 20 people organized in five groups operating at three different levels of the organization (work centers, regional departments, and main headquarters). Basically, each group was composed of the manager in each department of the organization and their support staff (e.g., the manager of a work center located in Milan and two staff people). Each meeting lasted about two hours and was organized in three steps. Initially, the participants were asked to present their mission in the organization and the relationship with the other company's entities. Then we moved to discuss problems and issues encountered in their daily activity. Finally, through brainstorming we interactively produced an initial draft of the GQM goals. Eventually, we identified 16 different goals (see Figure 1). The analysis of the goals and an iteration of some of the interviews led us to further subdivide some of the goals, identifying two or more precise subgoals whenever a goal was indeed ambiguous and further decomposable. Notice that Figure 1 is meant to describe the complexity of the relationships among users, goals, and subgoals in the considered case study. The meaning of the arrows is illustrated by the legend: a solid arrow connects users with the goals they have generated. A dashed arrow connects a goal with its subgoals. Finally, a plain arrow connects subgoals with a subsuming goal (see Section 4.3). Notice also that the box representing Goal G1 has been duplicated for readability reasons.

A precise identification and formulation of goals is fundamental to the purpose of data warehouse design. To appreciate this point, consider the definitions of Goal 1.1 and Goal 1.2:

Goal 1.1: maintenance costs. Analyze the service assurance process performed by the company's maintenance centers, to characterize its cost, duration, and resource usage from the viewpoint of different levels of management (i.e., the head of the maintenance center, the head of the regional branch where the maintenance center is located, and the head of the top-level direction).

Goal 1.2: quality and revenues of the access network. Analyze the quality of service and induced revenues of the access network (i.e., the physical telephone infrastructure) supported within the area managed by given maintenance centers, from the viewpoint of different levels of management (i.e., the head of the maintenance center, the head of the

regional branch where the maintenance center is located, and the head of the top-level direction).

These two goals apply to the same context, i.e., a single maintenance center and the portion of the access network under its direct control. The viewpoints are somehow similar. The real different factors are the object of study and the quality focus. In the first goal, the focus is on studying cost, duration, and effort related to the *service assurance process*. In the second goal, the focus is on studying the portion of the *access network* under the control of a maintenance center, to understand the relationship between quality of service and revenues for that geographic region.

As a general comment, we stress the importance of iterating on goals' definition until they become clear in all their aspects. Notice that the above two goals were originated by the same person. At the beginning of the interview, he expressed just a generic requirement about understanding costs and revenues. Only by formalizing the goals we realized that in reality there were different expectations and criticalities to be addressed.

4.3 Goal Integration

The subsequent step is the analysis of similarities and implications between goals, in order to reduce the number of identified goals to a manageable number. This result is achieved by suppressing goals that are subsumed by other goals and by merging related goals. Let us consider some further examples. Goal 12.1 was identified by interviewing a potential user of the data warehouse.

Goal 12.1: preventive maintenance. Analyze the access network to characterize the failure rate of network devices from the viewpoint of the manager of the regional branch and within the context of a single maintenance center.

This is a subgoal of Goal 1.2, since it aims at characterizing one aspect of the quality of service (failure rate) from the viewpoint of one of the managers indicated in Goal 1.2 (the manager of the regional branch). Therefore, we can suppress Goal 12.1, and note that it is subsumed by Goal 1.2.

As an additional example of goal formulation and restructuring, let us consider now a goal derived from interviewing a third potential user of the data warehouse:

Goal 9: perceived quality. Analyze the service assurance process to characterize the perceived quality from the viewpoint of the manager of a maintenance center and within the context of the same center.

This goal can be joined with Goal 1.1. Actually they are both related to properties of the maintenance service process. Goal 9 focuses on the perceived quality, i.e., the quality of the service as perceived by the end user, while Goal 1.1 considers other attributes of the maintenance process such as duration and cost of an intervention in field. Therefore, Goal 1.1

can encompass Goal 9 once we add perceived quality as a property of the service process.

Application of goal integration, in our case, identified four goals, labeled in Figure 1 as goals G1, G2, G3, and G4, that summarize and subsume all other goals:

Goal G1: costs and quality for maintenance centers. Analyze cost and quality factors, under direct control of maintenance centers, relative to the processes of *service assurance*, *service delivery,* and *network creation* from the viewpoints of the managers of maintenance centers, of regional branches, and of the top-level direction of the company.

Goal G2: incomes for maintenance centers. Analyze the income, under direct control of maintenance centers, relative to the processes of *service delivery* from the viewpoints of the managers of maintenance centers, of regional branches, and of the top-level direction of the company.

Goal G3: telephone traffic's income and quality of services for maintenance centers. Analyze the access network, under direct control of maintenance centers, for the purpose of characterizing the telephone traffic's income and the quality of the provided service, from the viewpoints of the managers of maintenance centers, of regional branches, and of the top-level direction of the company.

Goal G4: costs, use, and income for the top-level direction. Analyze the access network, under direct control of maintenance centers, for the purpose of characterizing the costs, use, and incomes, from the viewpoint of the top-level direction of the company.

Due to space reason, in the remainder of the paper we will focus on goal G1 only.

4.4 Goal Analysis

The simple goal definition scheme used in the previous section is often insufficient to properly describe a goal. In many situations, the comparison of two goals can be accomplished only if they are described at a finer level of detail. Most importantly, we need to analyze goals in a more detailed way to elicit the information needed to produce the ideal star schemas. For this reason, we collect additional information on the nature of goals and use it to fill up *GQM abstraction sheets*. This additional information is once again elicited by carefully considering users' answers and observations during the interviews. As discussed in the remainder of this section, abstraction sheets turned out to be essential in the process of translating a perceived goal into the facts and dimensions of an ideal star schema.

Table I shows the abstraction sheet for Goal G1. The upper-left part of the sheet contains a detailed description of the quality focus, i.e., the characteristics of the object of study that we want to refine. For instance, in Table I the quality focus concerns the different relevant properties and

Table I. Abstraction Sheet for Goal G1

Object of Study	Purpose	Quality Focus	Viewpoint	Environment
Processes of service, assurance, service, delivery, and network creation	To characterize	Cost, duration, resource use, performance, quality for customer	Maintenance center manager, regional branch manager, top-level direction manager	Activities developed by the maintenance center

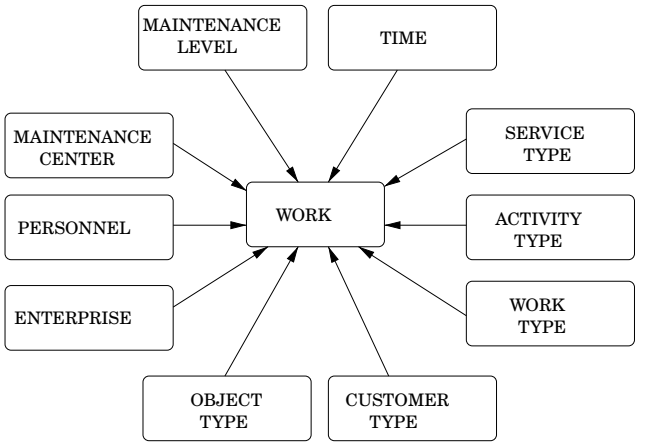
Quality Focus	Variation Factor
How can quality focus be detailed?	What factors can influence quality focus?
QF1. Cost (all the processes) QF1.1. Activity Cost QF1.1.1. Work Cost QF2. Duration (all the processes) QF2.1. Activity Duration QF2.1.1. Work Duration QF3. Resource Use (all the processes) QF3.1. Activity Resource Use QF3.1.1. Work Resource Use QF4. Resource Productivity (all the processes) QF4.1. Number of works per time unit QF5. Performance (all the processes) QF5.1. Average time of work QF5.2. Average time of requests' stock QF5.3 Number of works per time unit QF6. Quality for the customer (service assurance) QF6.1. Response time QF6.2. Resolution time QF6.3. Number of complaints QF6.4. Number of current works QF6.5. Number of open indications QF6.6. Number of works within N days	VF1. Type of service (service assurance, service delivery, network creation) VF2. Type of Activity VF3. Type of work (new plant, cessation, removal, ISDN, etc.) VF4. Type of customer VF4.1. Expenditure class (total on phone bill) VF4.2. Type of expenditure (subdivision device, traffic) VF4.3. Marketing sector VF5. Type of working object VF6. Personnel executing the work (social or industrial) VF7. Enterprise that executes the work VF8. Work Center (Maintenance Center) VF8.1. Division VF8.1.1. Employee VF9. Maintenance level (i.e., number of maintenance work hours in the past) VF10. Time of observation

Baseline Hypotheses	Impact on Baseline Hypotheses
What are the values assigned to the quality focus of interest?	How do baseline hypotheses vary quality focus?
BH1. Average cost of service assurance BH1.1. Average cost of activities of type X is Y BH1.1.1. Average cost of works for activities of type W is T BH1.2. Average cost of activities of type U for clients V is Z BH1.3.	IH1. Average cost of activities for TOP customers is more than the average cost for business customers. IH2. Cost of activities of type A in Maintenance center B is more than cost in Maintenance center C. IH3. Average cost for business customer is less than cost for residential customer. IH4. Cost depends on the enterprise that executes the work.

characteristics of the service processes (e.g., cost, duration, resource used, performance, perceived quality,...). The upper-right part of the sheet contains the variations factors, i.e., those factors that are supposed to influence the values of the attributes. For instance, in the example it was observed that the trends of attributes such as cost and performance of the maintenance process might depend on factors such as the type of the customer requesting the service (e.g., business or residential). Another dependency might be identified between costs and work centers (i.e., different work centers have different productivity). The two lower parts of the abstraction sheet contain the baseline hypotheses and the impact of variation factors. Baseline hypotheses represent typical beliefs about the values of fact attributes (actual values for variables are not given in Table I, as they do not contribute insights to a reader). For instance, in Table I, a baseline hypothesis states the value estimated for the average cost of a service. The impact of variation factors expresses users' beliefs about the influence that variation factors have on the baseline hypotheses. For instance, often the cost of a maintenance activity for residential customers is higher than for business customers, since the location of the customer might be located in the country side or in other areas that are difficult to reach.

4.5 Deriving Ideal Schema Fragments

Abstraction sheets discussed in the previous section are extremely useful to create an effective characterization of goals. Indeed, filling up abstraction sheets is a fairly simple process, since it is based on refining and discussing the information provided by data warehouse users. At the same time, once created on the basis of user's information, an abstraction sheet can be revisited and reinterpreted to derive ideal star schemas. To do so, the different zones of the abstraction sheet are interpreted as descriptors of the facts and dimensions that "ideally" should be part of the data mart; these generate the "ideal" star schema. In particular, the items listed in the `quality focus` section are considered as quality measures, i.e., facts in the ideal star schema. Then, the `variation factors` section defines the variables that may cause changes to quality measures, i.e., dimension analysis in the ideal star schema. Once singled out and named, each individual quality measure contributes an attribute to the ideal facts, and each variation factor contributes a dimension. Furthermore, the `baseline hypotheses` represent the expected queries that the data warehouse is supposed to manage. Then, these are verified "a posteriori" against the ideal star schema, in order to check whether they can be expressed with the chosen fact and dimensions. The `impact on baseline hypotheses` corresponds to the expected query result that the data warehouse is supposed to produce once it has been created. They can be verified "a posteriori" on the running system implementing the data warehouse. The process of extraction of information from abstraction sheets is accomplished manually, but the basic rules are straightforward and could be supported by a tool.



GQM Factual Attributes
Cost
Duration
Resource use
Average time of work
Average time of requests' stock
Number of works per time unit
Response time
Resolution time
Number of complaints
Number of current works
Number of open indications
Number of works within N days

Fig. 2. Ideal star schema for Goal G1.

Example. The abstraction sheet of Table I for goal G1 can be interpreted to generate the ideal star schema of Figure 2. The quality factors (e.g., *Work Cost*, *Work Duration*, *Work Resource Use*) detail the attributes of the ideal fact, *Work*, while the variation factors (e.g., *Work Type*, *Service Type*, *Activity Type*) correspond to the ideal dimensions. The careful analysis of the abstraction sheet leads to extracting 12 fact attributes and 10 dimensions, as shown in Figure 2. In addition, the baseline hypotheses section provides reasonable queries on the ideal star schema, while the impact section represents reasonable expected results of the data analysis.

Eventually, the top-down phase discussed in this chapter is able to identify a set of ideal star schemas. Notice that these candidate star schemas might turn out to be unfeasible to implement, due to the lack of information in the operational database or to the high cost of collecting them. This is exactly the rationale for complementing the top-down phase discussed so far with a bottom-up, data-driven phase (discussed in the next chapter), where candidate star schemas are derived from the structure of the available operational databases. The results of these two complementary phases are then matched in the final step of the proposed method (the integration phase).

5. BOTTOM-UP ANALYSIS

Bottom-up analysis aims at identifying all the candidate star schemas, i.e., star schemas that can be realistically implemented on top of the available operational databases. Starting from an entity-relationship (ER) model, this step performs an exhaustive analysis in order to discover the entities that are candidates to become facts, and then produces a large number of directed graphs having these entities as center nodes. Each graph corresponds to a potential star schema, characterized by the fact entity and with all the possible dimensions around it.

Potential star schemas are produced by an algorithm exploring the ER model. The algorithm considers an ER schema as a graph; it initially transforms each n-ary ($n > 2$) or many-to-many relationship into one-to-many relationships, by adding suitable nodes and edges. Then, the algorithm assigns to each entity a numerical label equal to the number of additive attributes (attributes with a numeric or temporal interval type) it has: potential fact entities are the ones having a nonnull label. Next, each potential fact entity is taken as the center node of a graph, produced by considering all possible dimensions reachable from the center through one-to-many relationships or one-to-one relationships. The algorithm automatically produces all the possible snowflake schemas; it achieves a good efficiency reusing in the computation the results of previous steps. Finally, snowflake schemas are reduced to star schemas, performing a step that can be interpreted as a denormalization of relations.

We first describe how an entity-relationship schema (ER) is mapped to a Connectivity Graph (C-Graph) which synthesizes the information of the schema relevant for the analytical processings performed in a data warehouse. Next, we show the algorithm that extracts snowflake schemas from the C-Graph. Finally, we show how each snowflake schema is transformed into a star schema. Each step is described by means of examples drawn from the considered case study.

5.1 Mapping an Entity-Relationship Schema into a Connectivity Graph

Let ER be an entity-relationship schema.

Definition 1. All the entities are *potential dimensional entities* (or *dimensions*).

Definition 2. A *potential fact entity* is an entity $e_j \in R$ such that e_j has at least one additive attribute a (i.e., an attribute with a numeric or temporal interval type).

Definition 3. The *C-Graph* is a directed graph $\{V, E\}$ constructed from the ER schema as follows.

—Each entity $e_i \in ER$ corresponds to a node $v_i \in V$. Fact nodes are represented with boxes (called *fact nodes*), all other entities with circles. Each fact node $\langle e_i, label_{e_i} \rangle$ is labeled with the number of additive attributes appearing in the entity.

—Each one-to-many relationship from entity e_i to entity e_j is mapped to a *directed* arc $\langle e_i, e_j \rangle \in E$.

—Each one-to-one relationship between entities e_i and e_j is mapped to a *bidirectional* arc $\langle e_i, e_j \rangle \in E$.

—Each many-to-many relationship from entity e_i to entity e_j is converted in two directed arcs originating from a new node e_k and directed to the nodes e_i and e_j.

—Each n-ary relationship between entities $e_1, \ldots, e_n$ introduces a new node e_k and n directed arcs originating from e_k and directed to $e_1, \ldots, e_n$.

—Each generalization hierarchy between an entity e_i and its specialization entity e_j is mapped into a *directed dashed* arc $\langle e_i, e_j \rangle \in E$.

The arcs are labeled with the name of the corresponding relationship in the *ER* schema. To simplify the C-Graph, we omit to represent the labels, except when there is ambiguity because two or more arcs of the same kind connect the same two entities.

Example. By applying this construction to the conceptual schema of our case study, which is not represented here for space reasons, we obtain the C-Graph of Figure 3, representing the activities of each maintenance center, their decomposition into basic steps, the elementary network components being repaired, the materials required, and the employees executing the repair.

Activity is a typical fact entity in the schema, as it contains five additive attributes, i.e., the number of persons involved, the duration of the activity expressed in minutes, the number of minutes required to complete the activity (zero for completed activities), the minimum starting time of the activity, and the maximum one. *Activity* is also a dimension entity. In fact, an entity with additive attributes can be a fact entity in a candidate star schema and a dimension entity in another. *Network Object*, instead, is only a dimension entity, because it does not have any additive attribute.

Activity and *Network Object* were connected in the ER schema by a one-to-many relationship, and so there is a directed arc in the corresponding C-Graph. *Territorial Working Unit* was a specialization of *Maintenance Center* in the ER schema, and therefore these two nodes are connected by a directed dashed arc in the C-Graph. *Maintenance Center* and *Geographic Area* were linked through a one-to-one relationship, so a double-edged continuous arc links the corresponding C-Graph's nodes. Between *Activity* and *Means Type*, instead, there was a many-to-many relationship: this is turned into a fictitious node (*Estimated Means*) linked through two directed arcs to the *Activity* and *Means Type* nodes.

5.2 Exploring the C-Graph

Each potential fact entity is a center of a snowflake schema. We present an algorithm that finds all the candidate snowflake schemas, starting from the potential fact entities and exploring all possible paths that reach dimensional entities. The algorithm receives as input the C-Graph and produces as output one subgraph (called *snowflake graph*) for each potential fact entity; each snowflake graph contains the reachable nodes and arcs from that node.

Algorithm 1 (Snowflake Graph Algorithm). Given an arbitrary C-Graph $\langle V, E \rangle$, the algorithm returns all the possible snowflake graphs. The set of

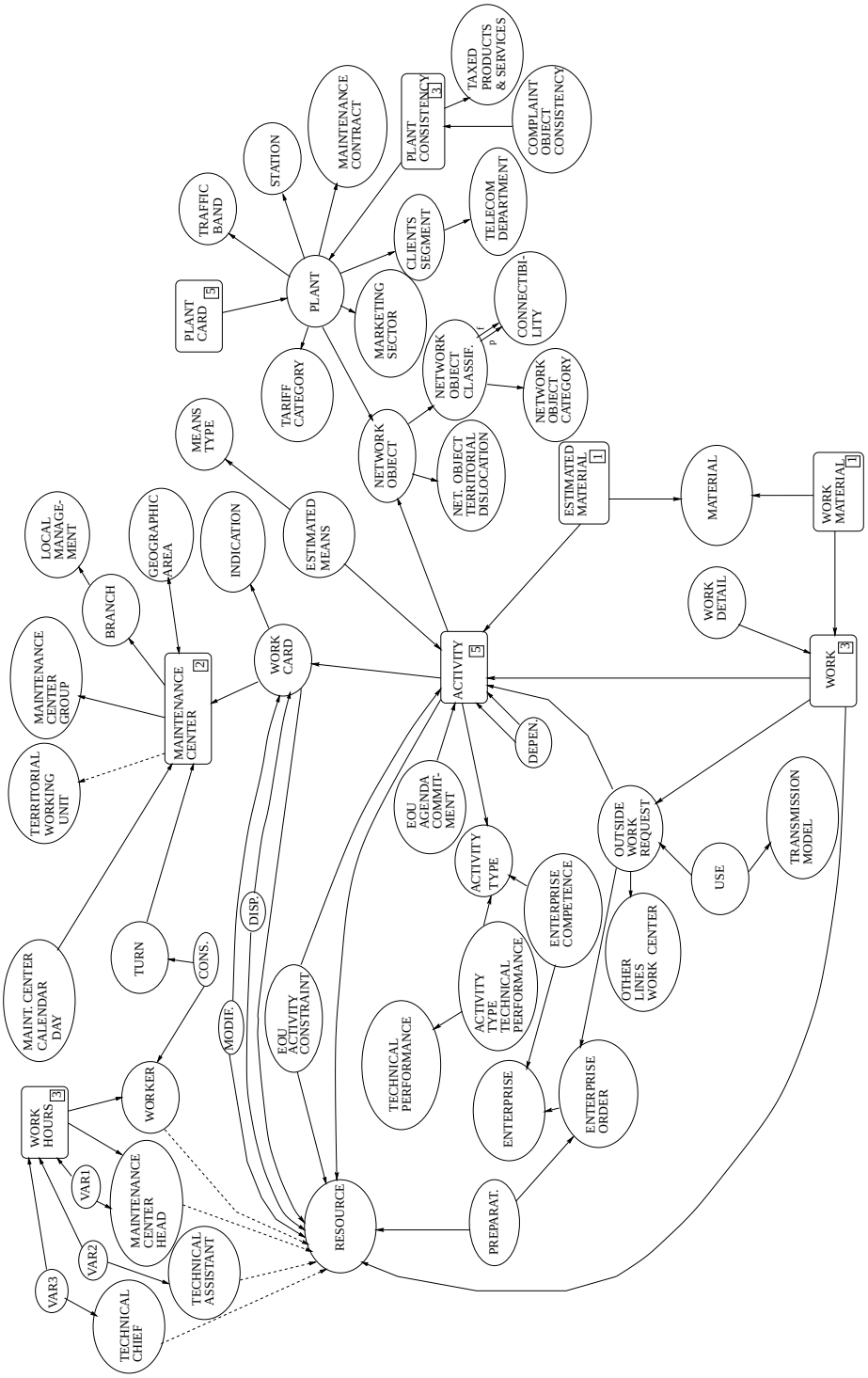


Fig. 3. C-Graph of the case study.

snowflake graphs is contained in an array of graphs SF: the graph in position i of the array $SF[i] = \langle V_i, E_i \rangle$ represents the snowflake graph obtained starting the analysis from node n_i. If a subgraph is entirely included in another, and the first has already been computed, the method avoids the recomputation of the subgraph.

```
begin
    for each n_i ∈ C-Graph
        SF[i]:= SF[i] ∪ n_i;
        if not n_i.visited then
            nodesToAnalyze:= {n_i};
            Visit(n_i, nodesToAnalyze, SF);
        end for;
    end;


procedure Visit(n_i: node; nodesToAnalyze: setOfNodes; SF: arrayOfGraphs)
begin
    n_i.visited:=true;
    for each a_j ∈ n_i.outgoingArcs
        n_k:= a_j.endpoint;
        if not n_k.visited then
            for each n_l ∈ nodesToAnalyze
                SF[l]:= SF[l] ∪ a_j;
                SF[l]:= SF[l] ∪ n_k;
            end for;
            nodesToAnalyze:= nodesToAnalyze ∪ n_k;
            Visit(n_k, nodesToAnalyze, SF);
        else
            for each n_l ∈ nodesToAnalyze
                SF[l]:= SF[l] ∪ SF[k];
            end for;
        end if;
    end for;
end;
```

The core of the algorithm is represented by the recursive procedure *Visit*. The procedure receives as input parameters the set of nodes currently analyzed and extends the search in the direction of outgoing arcs. Procedure *Visit* actually implements a depth-first exhaustive exploration of the graph. The procedure starts setting the value of a *visited* flag, associated to the visited node n_i, to *true*; this step guarantees that the procedure is executed only once for each node. The procedure then considers all the arcs exiting from node n_i; for each node n_k reached by the arc, if n_k has already been visited, the snowflake graph associated to it is added to all the snowflake graphs associated to the nodes currently being visited (those in *nodesToAnalyze*); otherwise the search continues in a depth-first way, adding n_k to *nodesToAnalyze*. The procedure is called by an external program, which is simply responsible of invoking the procedure on all the nodes.

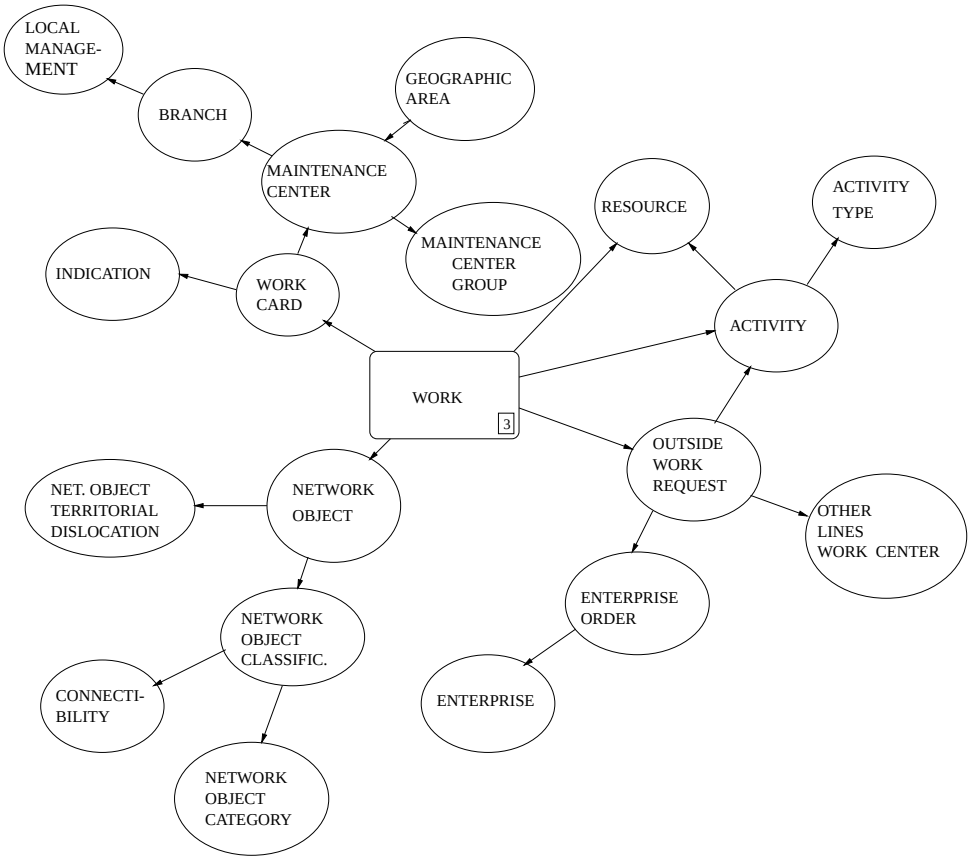


Fig. 4. Example of snowflake graph.

Complexity Analysis. The complexity of the algorithm is linear with the size of the output, because at the innermost level there are operations that create the output structure, and no redundant assignments are done. Since the algorithm is linear in the size of the output, the complexity of the algorithm is optimal. The size of the output is quadratic on the number of nodes and arcs of the graph: if n is the number of nodes and m the number of arcs, the result will contain n snowflake graphs with at most $n + m$ components (nodes and arcs) in each of them. Since schemas have normally a limited number of potential fact entities and have a simple structure, the actual complexity of the algorithm is typically acceptable.

Example (Continued). From the ER schema of the case study we obtain eight snowflake graphs, whose center nodes are *Activity*, *Work*, *Work Material*, *Estimated Material*, *Plant Consistency*, *Plant Card*, *Maintenance Center,* and *Work Hours*. The one centered on *Work* is shown in Figure 4: this snowflake graph (and the others not shown, due to space limitations) is quickly obtained from the C-Graph of Figure 3, starting from the potential fact *Work* and yielding, through the algorithm, all the possible reachable dimensions.

Table II. Properties of Candidate Star Join Graphs Extracted from the C-Graph of Figure 3

Star Join Graph	Number of Dimensions	Number of Fact-Additive Attributes
Work Hours	9	3
Activity	14	5
Work	19	3
Maintenance Center	4	2
Estimated Material	16	1
Work Material	21	1
Plant Card	13	5
Plant Consistency	14	3

5.3 Star Join Graph

Snowflake graphs are converted into simpler star join graphs. The rationale of this conversion step is, that although snowflake graphs are a rich and nonredundant representation of a data mart, the matching of ideal and candidate data marts is eased if the dimension hierarchies are abstracted away. In fact, this representation lets us focus on a direct one-to-one matching among the concepts identified by the top-down and bottom-up approaches.

Definition (Star Join Graph). Let $\langle V_i, E_i \rangle$ be an arbitrary snowflake graph. A star join graph $\langle N, P \rangle$ is derived from $\langle V_i, E_i \rangle$ with the following rules:

—$e_i \in V_i$ corresponds to $f \in N$ (fact);

—all other nodes $e_k \in V_i$ correspond to $d_k \in N$ (dimensions);

—$p_i = \langle e_k, e_h \rangle$; $p_i \in P$; if $\langle e_k, e_h \rangle$ is a directed arc or a directed path between e_k and e_h, then e_k is a fact entity and e_h is a dimension entity.

Example (Continued). Eight snowflake graphs are extracted from the C-Graph of Figure 3, as shown in the previous example. Next, Definition 4 is applied, and the snowflake graphs are converted into star join graphs illustrated in Figure 5 and Figure 6 flattening the dimension hierarchies. For each star, we list the fact attributes. *Work Hours* and *Maintenance Center* have a small number of dimensions, while *Work* or *Estimated Material* are very rich in dimensions. *Activity* and *Plant Card* have several additive attributes, while *Estimated Material* has only one. The number of dimensions and of aggregate attributes is summarized in Table II.

6. INTEGRATION

The top-down phase identifies the structure of the warehouse as it emerges from user requirements. The bottom-up phase returns all the possible multidimensional schemas that can be extracted from the existing information system. The crucial step is to integrate the two solutions, specifying how the ideal requirements can be mapped to the real system. The integration may also give the opportunity to consider new analysis aspects

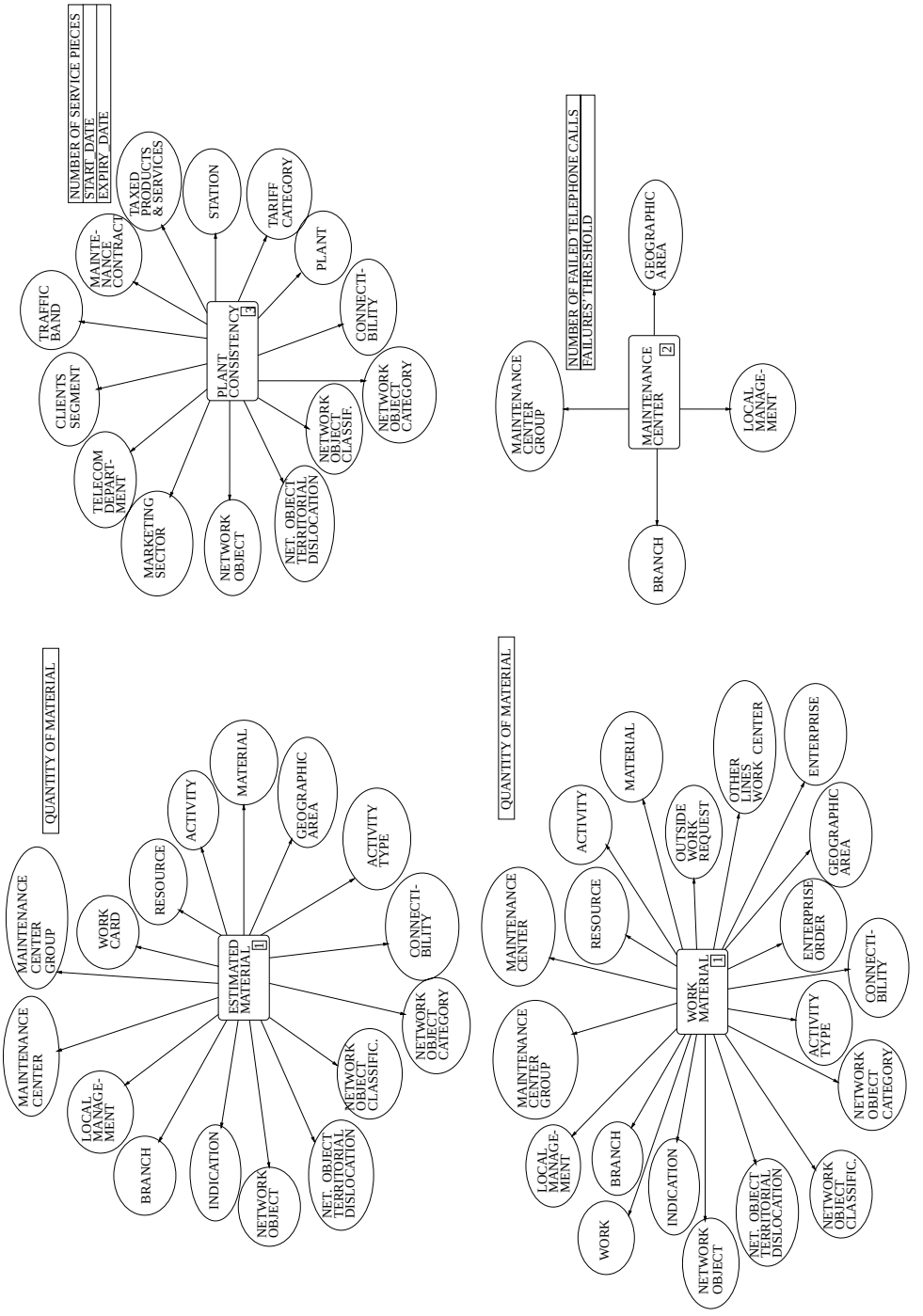


Fig. 5. Star join graphs (Part I).

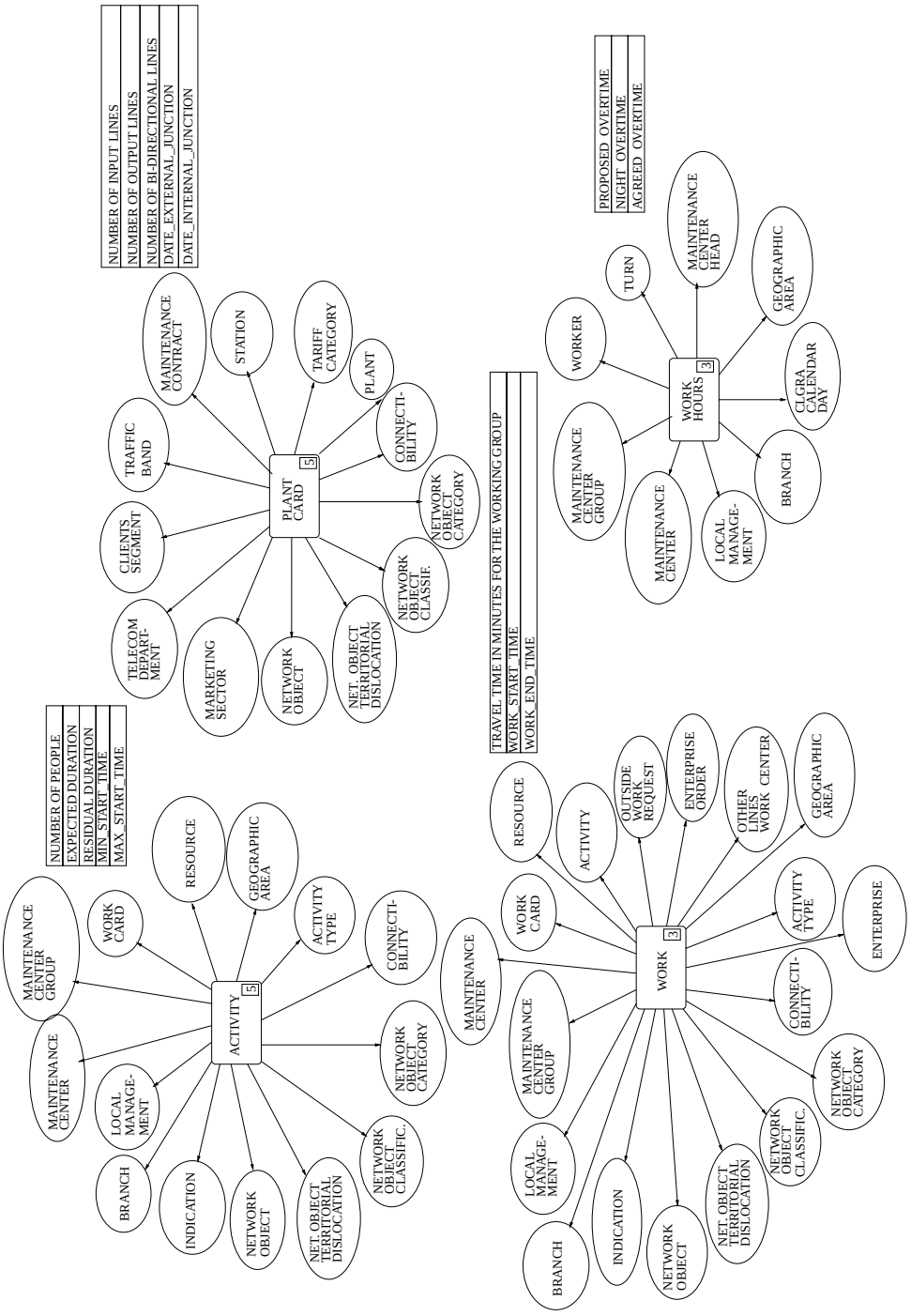


Fig. 6. Star join graphs (Part II).

that did not emerge from user requirements, but that the system may easily make available.

Integration is carried out in three steps:

Terminology analysis: Before integration, top-down and bottom-up schemas have to be converted to a common terminological idiom: this is accomplished defining how GQM terms are mapped to system's terms.

Schema matching: After terminology mapping, candidate solutions are compared one-by-one with ideal solutions representing GQM goals. A match occurs when the ideal and candidate solutions have the same fact; the metrics considers the number of their common fact attributes and dimensions.

Ranking and selection: Eventually, some of the candidate star schemas are selected, based on the goal's priority as well as on the metrics of the matching.

Each step is detailed below and is applied to our case study.

6.1 Terminology Analysis

Terminology analysis is applied to all the terms of the ideal stars (GQM terms) and performs their mapping to the terms which appear in candidate stars. The mapping is *direct* when the GQM term corresponds to a concept that is also immediately available in the system; it is *indirect* when the GQM term corresponds to a concept that can be derived by computing a function of several concepts that are available in the system; we say that the GQM term is *computed*. The same GQM term can be mapped to several system terms, both directly and indirectly. It may also happen that a GQM term is different from all the system terms, so that specific user requirements cannot be satisfied. Initially, the mapping considers the facts that are the centers of ideal and candidate stars, and extends subsequently to their attributes and dimensions.

Example (Continued). For instance, let us consider the ideal star $IG1$ of goal $G1$ (Figure 2) and the eight candidate stars of Figure 5 and Figure 6. Let $CG1$ denote the *Work* candidate join graph and $CG2$ denote the *Activity* join graph. First of all, the term $Work_{IG1}$ (the name of the fact entity) is directly mapped to the terms $Work_{CG1}$ and $Activity_{CG2}$ (the names of the central nodes of the candidate star schemas $CG1$ and $CG2$).

Terminological investigation is then extended to the fact attributes and dimensions in $CG1$ and $CG2$. For example, attribute $ResourceUse_{IG1}$ is directly mapped to $NumberOfPeople_{CG1}$, once it is understood that resources of the *Work* ideal schema are concerned only with personnel; $Duration_{IG1}$ corresponds either to a direct mapping to *Expected Duration* $CG2$ or to an indirect mapping to the difference function applied to $WorkStartTime_{CG1}$ and $WorkEndTime_{CG1}$. The existence of alternatives in the name mapping is an indicator of ambiguities in the requirements,

Table III. Thesaurus for Goal G1 (Ideal Schema)

GQM Term	Star Join Graph Term	GQM Term Mapping
$Work_{IG1}$	$Work_{CG1}$	*Direct*
	$Activity_{CG2}$	*Direct*
$Duration_{IG1}$	*Expected Duration*$_{CG2}$, $WorkStartTime_{CG1}$, $WorkEndTime_{CG1}$	*Direct* *Indirect (difference)*
$ResourceUse_{IG1}$	$NumberOfPeople_{CG1}$	*Direct*
$AverageTimeOfWork_{IG1}$	$WorkStartTime_{CG1}$, $WorkEndTime_{CG1}$	*Indirect (average)*
$ObjectType_{IG1}$	$NetworkObject_{CG1}$	*Direct*
$NumberOfCurrentWorks_{IG1}$	*instances of*$Work_{CG1}$	*Indirect (count)*
$NumberOfWorksWithinNDays_{IG1}$	*instances of* $Work_{CG1}$	*Indirect (count)*
$NumberOfWorksPerTimeUnit_{IG1}$	*instances of* $Work_{CG1}$	*Indirect (count)*
...	...	...

which need to be further clarified at this stage (e.g., by means of additional interviews with the users).

GQM attributes that count the fact instances typically do not have a direct mapping, but can be obtained indirectly as simple query manipulations of the instances of candidate schemas. An example is the $AverageTimeOfWork_{IG1}$, that can be obtained by considering for each work the difference between attributes $WorkStartTime_{CG1}$ and $WorkEndTime_{CG1}$, then counting the number of instances of $Work_{CG1}$, and finally applying the average operator.

In the end, terminology analysis terminates by building a thesaurus with the correspondence between GQM terms and system terms. Table III shows a sketch of the thesaurus developed for $IG1$ in the case study.

6.2 Schema Matching

To assist the designer in the choice of the best system model, the ideal GQM schemas are matched against the candidate star schemas. The matching is based on the analysis of the number of elements of candidate schemas that correspond, via the terminology analysis, to elements of the ideal GQM schema. We preliminarily require a correspondence between the facts that are the centers of ideal and candidate stars; this correspondence is mandatory (if it is not satisfied, the ideal and candidate schemas are focused on different topics and therefore are not comparable). A set of metrics has been identified to compare and rank schemas, when several candidate schemas meet this criterion.

—The *matching attributes* metrics count the number of fact attributes in a given candidate star schema that correspond to attributes in the ideal star schema.

—The *matching dimensions* metrics count the number of dimensions in a given candidate star schema that correspond to dimensions in the ideal star schema.

Table IV. Schema Matching for Goal G1 (Schema)

Ideal (GQM) Schema	Candidate Schema	Matching Attributes	Matching Dimensions	Additional Attributes	Additional Dimensions
$Work_{IG1}$	*Work*	5	7	1	12
$Work_{IG1}$	*Activity*	6	7	3	9
$Work_{IG1}$	*Work Material*	0	5	1	15

—The *additional attributes* metrics count the number of fact attributes in a given candidate star schema that do not correspond to attributes in the ideal star schema. They represent additional observations that may be offered to the analyst.

—The *additional dimensions* metrics count the number of dimensions of a given candidate star schema that do not correspond to dimensions in the ideal star schema. They represent additional dimensions that may be offered to the analyst.

Note that the first two metrics are indeed the most important ones, because they measure how a candidate schema is capable of delivering the information that was identified during users interviews. The next two metrics are complementary, as they indicate additional observations that may or may not be relevant to users, but could be performed with relatively small overhead (because they are offered by the underlying data). In our experience, these metrics are indeed important. If the overhead induced by adding them is very small, users prefer to add facts and dimensions to their specifications in the expectation that they will become useful for the analysis later on.

In some cases, one solution dominates the others, because it offers better indexes for all the four metrics. In general, a solution which dominates all the other ones according to the first two metrics is strongly advised.

Example (Continued). Table IV illustrates the indexes of the four metrics relative to the ideal schema $IG1$. We observe that the second candidate ($CG2$, *Activity*) *dominates* the first ($CG1$, *Work*) according to the first two metrics, while the third candidate *Work Material* is dominated by $CG1$ and $CG2$ according to the first two metrics and indeed has no matching facts, and thus can be excluded at this stage.

6.3 Ranking and Selection

This phase completes the design process. At this stage, each goal is ranked according to its priority, as indicated by the users. Moreover, for each goal a candidate star schema is selected. Then, the addition of facts and dimensions beyond the ones resulting from the requirements is decided. Finally, a decision is taken about the candidate star schemas, or data marts, to be implemented. Ranking is accomplished according to the needs and constraints of users. In our case study this was accomplished through interviews and brainstorming meetings. We believe that this kind of activity can hardly be performed automatically, as the ranking is mainly

Table V. Cardinalities and Sizes of Candidates Matching Goal G1

Candidate	Cardinality	Size
Work	10,000,000	5GB
Activity	5,000,000	2GB

based on motivations that are usually identified and assessed by the managers and owners of the process being studied. The selection of the most suitable candidate(s) can be based on cost modeling, on decision trees, or on other qualitative criteria. Other suitable criteria concern the optimized selection of views to be materialized in the data warehouse.

Example (Continued). In our case study, goal G1 was selected for implementation using the data mart represented by the *Activity* schema. Out of the remaining three goals, G2 and G3 were found unfeasible (there was no matching between the ideal facts and the real star schema facts), and G4 was found partially feasible (only partial coverage of the fact attributes). In particular, the goal G4 was covered partially by the *Plant Card* schema, for the fact attributes concerning the network usage, but the other fact attributes (costs and incomes of the network) were not available in the data marts extracted from the operational database.[2]

In order to complement the above decision process, we found useful the addition of gross estimates of the cardinalities and sizes involved in the data mart. Sizes are specified under the assumption of adding only the matching fact attributes. These estimates give a physical characterization of the data mart, which complements the logical characterization summarized in Table IV. For instance, quantitative data relative to the same candidates are reported in Table V. They give an additional motivation in selecting the *Activity* schema over the *Work* schema, because the former is smaller in size.

7. SUMMARY DESCRIPTION OF THE PROPOSED METHOD

The previous sections have introduced a method to design data warehouse schemas; this section will provide the reader with a summary description of the proposed method, meant to facilitate and support the application of the proposed method. We abstract out irrelevant details, emphasize the main steps and critical issues, and provide guidance and support to the data warehouse designer; we also indicate when a specific activity can be accomplished manually, automatically, or semiautomatically.

The method is structured in three steps: creation of ideal star schemas (top-down phase), derivation of candidate star schemas from database conceptual schemas (bottom-up analysis), and matching of the ideal schemas with star join graphs (integration and ranking).

Top-Down Phase.

(1) Collect user requirements through interviews (manual).

(2) Represent requirements as GQM goals:

(a) For each goal, fill in the goal definition, i.e., define the object of the study, the purpose, the quality focus, the viewpoint, and the environment (manual).
(b) For each goal, detail the goal by filling in the abstraction sheet: define the quality focus, the variation factors, the baseline hypotheses, and the impact on baseline hypotheses (manual).
(c) Compare and assess the goals: identify similarities and implications among goals to reduce them to a manageable number, i.e., merge related goals and drop goals that are subsumed by others (manual).

(3) Derive ideal schema fragments for each goal:
(a) Map quality focus into the facts in the ideal star schema, i.e., its components are the attribute of the fact (automatic).
(b) Map variation factors into the star schemas dimensions (automatic).
(c) Use the baseline hypotheses to derive the queries onto the data warehouse (semiautomatic). Notice that the impact on baseline hypotheses is the set of expected query results.

Bottom-Up Analysis.

(4) Obtain the Star Join Graphs:
(a) Map the E/R schema into a connectivity graph (automatic).
(b) Run the Snowflake Graph Algorithm, to find all the possible snowflake graphs (automatic).
(c) Convert each snowflake graph into a star join graph (automatic).

Integration and Ranking.

(5) Match the ideal schema with the star join graphs:
(a) Identify common terms between the ideal schema and the star join graphs, through direct/indirect mapping (manual/semiautomatic, depending on the features of the dictionary used to perform the match).
(b) Match ideal and candidate schemas, taking into account the matching attributes of the schemas, the matching dimensions, the additional attributes, and the additional dimensions (semiautomatic).
(c) Rank the solutions (manual).

8. EVALUATIONS AND CONCLUSIONS

In conclusion, we examine the pros and cons of several features of the proposed method, and outline future directions of development.

First, we discuss formal versus informal aspects of our method. The first phase (top-down) is based on an informal and conversational approach, while the second phase (bottom-up) is more formal and rigorous, and yields to an integration phase where a formal metric can actually be applied to match requirements against candidate solutions. The top-down phase deals with distilling and consolidating user requirements; this activity is by its nature a complex interpersonal process, where information is progressively collected and refined from inherently informal and incoherent sources.

Conversely, the second and third phases start from a formal and structured information (i.e., the schema) and can therefore be supported by a rigorous and automatable technique. The choice of GQM as a guiding methodology for the top-down phase is justified by the need of exploiting a goal-oriented approach to requirement elicitation. The advantage of GQM with respect to other goal-oriented techniques lies in the intrinsic orientation of GQM toward eliciting qualitative and quantitative factors characterizing a complex phenomenon.

The interaction with the potential users of the data warehouse has been carried out using interviews. Other approaches could be fruitfully applied to collect information on users' needs and expectations. We relied on interviews for several reasons, including a limitation in the time and resources that we could invest in the study. We consider the application of more sophisticated techniques (such as in-field studies) as a promising enhancement to the method proposed in this paper.

Another remark concerns the structure of the design process induced by the proposed method. For the sake of simplicity, we presented it as a sequence of steps, as in the waterfall lifecycle; but of course, a designer could proceed adopting a spiral lifecycle, as suggested by Boehm [1988]; alternatively, one might decide to explore only a specific goal in detail, if other goals are patently infeasible to achieve.

It is indeed essential to emphasize the role of domain analysis as an important aid to achieve an effective goal characterization. As in any requirement engineering activity, users' requirements can be properly captured if the software engineer has good knowledge of the concepts and principles of the domain being observed. In our approach, domain analysis plays an important role in the formalization of goals, in the construction of the thesaurus, and in the final ranking of the identified goals and solutions. We did not explore any specific techniques to represent and exploit domain-specific information (except for the thesaurus). Therefore, another area of further investigation is represented by the integration of our approach with specific techniques for domain knowledge representation.

An important factor that must be taken into account to evaluate the effectiveness of the approach proposed in this paper is its ability to scale in the context of real settings. Some steps can be completely automated, but certainly others have necessarily to rely on human intervention, as the design requires the capture of users' requirements or some decision-making activity. We argue that this does not represent a limitation to the applicability of the approach to large problems; indeed, the proposed method is instrumental to master the complexity of large settings, as the adoption in the top-down phase of GQM goals constitutes an effective means to organize the problem space.

As for the duration and cost of applying the proposed method, in the case study the duration of the design activity was around three months. The effort requested was approximately 100 man/days, which included 60 man/days spent by our team, and 40 man/days spent by data warehouse users and the members of the company's information system department.

In conclusion, the experience we have gathered by applying the proposed method in our case study is encouraging. The combination of top-down and bottom-up steps helps in coherently evaluating both the coverage of users' requirements and the feasibility of the data warehouse design. In the case study, we started from a situation where the size and complexity of the operational database, and the ambiguity of users' requirements, could lead to designing very different data warehouses. The proposed method was indeed essential to direct the designer toward a solution that is both efficient to implement and consistent with users' critical requirements. The users' feedback was very positive: the development of a coherent and comprehensive design proved to be extremely useful to make their goals explicit, and to effectively guide the decision process of the company management.

We plan for future work is to explore the problems and issues in data warehouse design that have been discussed in this concluding section. In addition, we would like to extend the approach presented in this paper by considering richer criteria for the integration phase. Basically, at this stage we compare the ideal and candidate schema by assuming that the topology of the candidate schema can be manipulated and used only in a limited way (e.g., by computing counts and sums). Indeed, by applying classical relational operations (e.g., joins) it would be possible to create new relationships that might increase the chances to find a candidate schema (e.g., a portion of a real database) that fits the needs of the data warehouse to be created. We plan to study this issue further and to derive an enhanced set of criteria and guidelines to support the data warehouse designer in a more effective and comprehensive way.

REFERENCES

AGRAWAL, R., GUPTA, A., AND SARAWAGI, S. 1997. Modeling multidimensional databases. In *Proceedings of the 13th International Conference on Data Engineering* (ICDE '97, Birmingham, UK, Apr. 7–11), A. Delis, C. Faloutsos, and S. Ghandeharizadeh, Eds. 232–243.

ANTON, A. I. AND POTTS, C. 1998. The use of goals to surface requirements for evolving systems. In *Proceedings of the 20th International Conference on Software Engineering* (ICSE '98, Kyoto, Japan, Apr.). IEEE Press, Piscataway, NJ, 157–166.

BASILI, V., CALDIERA, G., AND ROMBACH, D. 1994. Goal/question/metric paradigm. In *Encyclopedia of Software Engineering (vol. 1 A-N)*, J. J. Marciniak, Ed. Wiley-Interscience, New York, NY, 528–532.

BATINI, C. AND LENZERINI, M. 1984. A methodology for data schema integration in the entity-relationship model. *IEEE Trans. Softw. Eng. 10*, 6 (Nov.), 650–664.

BOEHM, B. W. 1988. A spiral model of software development and enhancement. *IEEE Computer 21*, 5 (May), 61–72.

CABIBBO, L. AND TORLONE, R. 1998. A logical approach to multidimensional databases. In *Proceedings of the International Conference on Extending Data Base Technology* (EDBT '98, Valencia, Spain, Mar.). 183–197.

CALVANESE, D., GIACOMO, G. D., LENZERINI, M., NARDI, D., AND ROSATI, R. 1998. Information integration: Conceptual modeling and reasoning support. In *Proceedings of the 6th International Conference on Cooperative Information Systems* (CoopIS '98, New York, NY, Aug.). 280–291.

CALVANESE, D., GIACOMO, G. D., LENZERINI, M., NARDI, D., AND ROSATI, R. 1998. Source integration in data warehousing. In *Proceedings of the 9th International Workshop on Database and Expert System Applications* (DEXA Workshop '98, Vienna, Austria, Aug. 24–28). IEEE Computer Society Press, Los Alamitos, CA, 192–197.

CHAUDHURI, S. AND DAYAL, U. 1997. An overview of data warehousing and OLAP technology. *SIGMOD Rec. 26*, 1, 65–74.

DARIMONT, R., DELOR, E., MASSONET, P., AND VAN LAMSWEERDE, A. 1997. GRAIL/KAOS: an environment for goal-driven requirements engineering. In *Proceedings of the 1997 International Conference on Software Engineering* (ICSE '97, Boston, MA, May 17–23), W. R. Adrion, Chair. ACM Press, New York, NY, 612–613.

EZEIFE, I. 1997. A uniform approach for selecting views and indexes in a data warehouse. In *Proceedings of the International Database Engineering and Applications Symposium* (IDEAS '97, Montreal, Canada, Aug.). 151–160.

FRANCALANCI, C. AND FUGGETTA, A. 1997. Integrating conflicting requirements in process modeling: A survey and research directions. *Inf. Softw. Technol. 39*, 3, 205–216.

FRANCONI, E. AND SATTLER, U. 1999. A data warehouse conceptual data model for multidimensional aggregation. In *Proceedings of the Workshop on Design and Management of Data Warehouses* (DMDW '99, Heidelberg, Germany, June).

FUGGETTA, A., LAVAZZA, L., MORASCA, S., CINTI, S., OLDANO, G., AND ORAZI, E. 1998. Applying GQM in an industrial software factory. *ACM Trans. Softw. Eng. Methodol. 7*, 4, 411–448.

GARCIA-MOLINA, H., LABIO, W. J., AND YANG, J. 1998. Expiring data in a warehouse. In *Proceedings of the 24th International Conference on Very Large Data Bases*, A. Gupta, O. Shmueli, and J. Widom, Eds. Morgan Kaufmann, San Mateo, CA, 500–511.

GOLFARELLI, M., MAIO, D., AND RIZZI, S. 1998. Conceptual design of data warehouses from E/R schemas. In *Proceedings of the 31st Hawaii International Conference on System Sciences* (HICSS '98, Kona, Hawai).

GUPTA, H. AND MUMICK, I. 1999. Selection of views to materialize under a maintenance-time constraint. In *Proceedings of the International Conference on Database Theory* (ICDT '99, Jerusalem, Israel).

GYSSENS, M. AND LAKSHMANAN, L. V. S. 1997. A foundation for multi-dimensional databases. In *Proceedings of the 23rd International Conference on Very Large Data Bases* (VLDB '97, Athens, Greece, Aug.). 106–115.

HARINARAYAN, V., RAJARAMAN, A., AND ULLMAN, J. D. 1996. Implementing data cubes efficiently. *SIGMOD Rec. 25*, 2, 205–216.

INMON, W. H. 1996. *Building the Data Warehouse*. 2nd ed. John Wiley and Sons, Inc., New York, NY.

JARKE, M., LENZERINI, M., AND VASSILIOU, Y., EDS. 1999. *Fundamentals of Data Warehousing*. Springer-Verlag, Berlin–Heidelberg, Germany.

KIMBALL, R. 1996. *The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses*. John Wiley and Sons, Inc., New York, NY.

LABIO, W. J., ZHUGE, Y., WIENER, J. L., GUPTA, H., GARCÍA-MOLINA, H., AND WIDOM, J. 1997. The WHIPS prototype for data warehouse creation and maintenance. *SIGMOD Rec. 26*, 2, 557–559.

LEHNER, W. 1998. Modeling large-scale OLAP scenarios. In *Proceedings of the International Conference on European Database Technology* (EDBT '98, Valencia, Spain). 153–167.

MYLOPOULOS, J., CHUNG, L., AND NIXON, B. 1992. Representing and using nonfunctional requirements: a process-oriented approach. *IEEE Trans. Softw. Eng. 18*, 6 (June), 488–497.

THEODORATOS, D. AND SELLIS, T. 1998. Data warehouse schema and instance design. In *Proceedings of the 17th International Conference on Conceptual Modeling* (ER '98, Singapore). 363–376.

THEODORATOS, D. AND SELLIS, T. 1999. Designing data warehouses. *Data Knowl. Eng. 31*, 3 (June), 279–301.

VASSILIADIS, P., BOUZEGHOUB, M., AND QUIX, C. 1999. Towards quality-oriented data warehouse usage and evolution. In *Proceedings of the International Conference on Advanced Information Systems Engineering* (CAiSE '99, Heidelberg, Germany). 164–179.

WIDOM, J. 1995. Research problems in data warehousing. In *Proceedings of the 1995 International Conference on Information and Knowledge Management* (CIKM, Baltimore, MD, Nov. 28–Dec. 2), N. Pissinou, A. Silberschatz, E. K. Park, and K. Makki, Eds. ACM Press, New York, NY, 25–30.

Received: March 1999; revised: February 2000; accepted: April 2001