# DATA QUALITY BY DESIGN: A GOAL-ORIENTED APPROACH

(Research-in-Progress)
IQ in Databases, the Web, and e-Business

**Lei Jiang**[1], **Alex Borgida**[2,1], **Thodoros Topaloglou**[1], **John Mylopoulos**[1]
[1]University of Toronto, [2]Rutgers University
leijiang@cs.toronto.edu, borgida@cs.rutgers.edu, thodoros@mie.utoronto.ca, jm@cs.toronto.edu

**Abstract**: We present a design process for incorporating data quality requirements into database schemas that is rooted in *goal-oriented requirements analysis* techniques, developed in the Requirement Engineering community over last 15 years. Goal-oriented approaches (i) offer a body of notations, techniques and processes for modeling, analyzing and operationalizing quality requirements; (ii) support representation and evaluation of alternatives in goal fulfillment; and (iii) provide automated reasoning tools for various analysis and design tasks. This paper extends existing proposals for addressing data/information quality issues during database requirements analysis and design in two ways. First, we consider a broader range of quality assurance data requirements, which can be classified as restrictive, descriptive, supportive and reflective. Second, we offer a *systematic* way to operationalize high-level, abstract quality goals into operational, concrete quality assurance data requirements plus standard operating procedures, based on a risk-based analysis. In an earlier paper, we presented a goal-oriented conceptual database design approach, focusing on deriving an ordinary conceptual schema from application-specific goals. In this paper, we take the next step to incorporate quality goals into the design process. The proposed quality design process is illustrated step-by-step using a meeting expense database example. We also show how this process fits into the overall goal-oriented conceptual database design approach to offer an integrated framework for analyzing both application-specific and quality assurance data requirements.

**Key Words**: Data Quality, Database Design, Conceptual Schema, Requirements Analysis, Goal Model, Softgoal.

## INTRODUCTION

Information Quality (IQ) aims to deliver high quality information to end-users; it covers the entire lifecycle of data acquisition, storage and utilization, and involves a range of stakeholders, such as data producers, custodians, managers and customers. One approach to study IQ problems is to view information as a product of an information manufacturing system, where each stage of the manufacturing process can be analyzed for quality concerns [4,22,20]. Databases are essential components of any information manufacturing system. Therefore, we consider data quality (DQ) at the database level as a sub-problem of IQ, mainly concerned with data acquisition and storage processes. The quality of observed, captured and stored data contributes greatly to the quality of delivered information products. The goal of this research is to tailor and apply the idea of "*quality-information-by-design*" [22] to databases. We propose to support IQ in terms of two complementary quality assurance mechanisms: (i) a set of *quality assurance data requirements* to be incorporated into database schemas at *design time*, and (ii) a set of *standard operating procedures* that are considered during schema design and carried out at *run- time*.

More specifically, we propose a *goal-oriented quality design process* to support high quality data during database requirements analysis and conceptual schema design. A necessary initial step of this process is the identification and analysis of application-specific goals (e.g., estimate meeting budget for the next fiscal year). In our past work, we have proposed a goal-oriented conceptual database design (GODB) approach [15] for this task. In this paper, we take the next step, showing how high-level quality goals (e.g., accuracy, security) themselves are represented and operationalized in terms of technical design decisions.

The rest of paper is organized as follows. We first review existing approaches to incorporate quality requirements into database schema design, and give an overview our previously proposed GODB approach. We then briefly discuss the rationale behind the present work, including a list of contributions of this paper. This is followed by a detailed description of the quality design process, illustrated step-by-step using a meeting expense database example. Next, we discuss, in general terms, different types of quality assurance data requirements, and show how the proposed quality design process fits into the overall GODB approach to offer an integrated framework for addressing both application-specific and quality assurance data requirements. Finally, we conclude and point out future research directions.

# BACKGROUND

## *Previous Work on Data Quality by Design*

It has long been accepted that DQ problems need to be recognized at the requirements analysis stage. For example, [23] introduces a set of concepts and premises for DQ modeling and analysis, and proposes a process for defining and documenting quality requirements. In [23], data requirements are divided into application data requirements (called *application attributes*), such as a person address and a stock price, and quality data requirements (called *quality attributes*). Furthermore, quality attributes are considered at two levels: *quality parameters* model qualitative and subjective dimensions by which a user evaluates DQ (e.g., credibility and timeliness); *quality indicators* capture aspects of the data manufacturing process (e.g., when, where, how data was manufactured) and provide objective information about quality of the produced data. The quality requirements analysis process starts with a conventional conceptual data modeling step, where application attributes are elicited and organized into a conceptual schema. Then quality parameters are identified and associated with certain application attributes in the schema. Finally, each parameter is "refined" into one or more quality indicators. Although a significant first step, there are some limitations to this approach. First, the final result of this process is the initial conceptual schema, tagged with various quality indicators. The process of incorporating these quality indicators to produce a new conceptual schema is missing. Second, as an important step, the transition from subjective quality parameters into objective quality indicators is left open. Although, some of the refinements are quite straightforward (e.g., from "timeliness" to "age"), others are less obvious (e.g., from "accuracy" to "collection method").

Quality requirements, once analyzed and documented, need to be incorporated into a conceptual schema during the schema design stage. The Quality Entity Relationship (QER) approach [21] addresses the first limitation mentioned above by providing a mechanism to embed quality indicators into conceptual schemas. The QER approach introduces two generic entities, *DQ Dimension* and *DQ Measure*. *DQ Dimension*, with attributes *name* and *rating*, models all possible quality dimensions (e.g., accuracy) for an application attribute (e.g., address) and their values (e.g., accuracy = "8"). *DQ Measure* is used to represent the interpretation for these quality values (e.g., "1" for very inaccurate, "10" for very accurate). This approach focuses only on a particular class of quality indicators – those used to *directly record* the actual or estimated DQ assessment. It can not accommodate quality indicators that are used to *indirectly*

*indicate* DQ of application attributes (e.g., "collection method" for "accuracy"), or to *ensure* quality of application attributes (e.g., "last audit date" for "accuracy").

## *Our GODB Approach: An overview*

The GODB approach we have proposed [15] covers both the analysis of initial requirements and the specification of these requirements in terms of a conceptual schema (see Figure 1). Goal-oriented requirements analysis starts with a list of *stakeholders* and their high-level goals, including both *hard goals* and *softgoals*. The former lead to application-specific data requirements, while the latter lead to quality assurance data requirements. These goals are analyzed (by humans) using three types of techniques: (a) *AND/OR-goal decomposition* for refining high-level abstract goals into lower-level, operational ones; (b) *means-end analysis* for identifying operations (modeled as *plans*) to fulfill the refined goals; and (c) *contribution analysis* for detecting lateral influence on goal fulfillment. The result is a goal model that captures not a single, but several alternative sets of data requirements (i.e., *design alternatives*), from which a particular one must be chosen to generate the conceptual schema for the database-to-be. The modeling of goals, operations and their relationships in a database design context is influenced by the TROPOS goal-oriented software development methodology [7], which is evolved from the i* framework [24] for modeling and reasoning about organizational environments and their information system requirements.

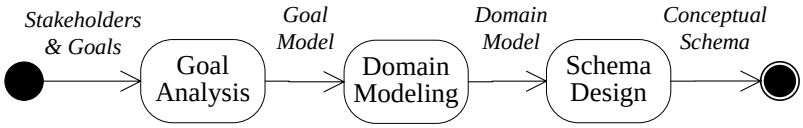


Figure 1. The GODB approach proposed in [15]

Goal-oriented schema design is divided into two stages: the modeling of the application domain and the detailed design of the conceptual schema. The *domain model* is constructed by extracting concepts, relationships and attributes from the hard goals and plans in the chosen design alternative. A domain model describes the necessary understanding of a part of the real world, and facilitates the communication of domain knowledge between developers, end-users and other stakeholders. It represents the application-specific data requirements. A *conceptual schema*, on the other hand, represents the semantics of the actual data in the proposed database; its design focuses on data specific *design issues* that are not relevant in domain modeling. DQ is one such important design issue. Consequently, we proposed a transformational approach from the domain model to the conceptual schema, using a sequence of *design operations*, some of which may be available in a knowledge base of design operations. For more details about this work, please refer to the original paper.

# RATIONALE

It is well accepted that the quality of data is strongly influenced by the quality of the schema [6] that dictates the structure of these data. On one hand, the way in which data is organized may affect its quality. For example, consider the single attribute *address* in a relational table *Person(id, name, address)*. There are several potential DQ problems associated with this design [6]. First, the value of address is an unstructured string where its different components may be ambiguous (e.g., can not be distinguish if a number in the string is a room number or a house number); second, no constraint can be specified on the components of address values (e.g., an address value without a street name will not be detected as a piece of incomplete data by traditional null-value checks); last, there is no way to enable automatic checks of accuracy in the event that there is a standard vocabulary applicable for a component (e.g., "street", "avenue", "road"). On the other hand, the way in which data is acquired also affects quality; a data acquisition process that produces high quality data may be required to produce "auxiliary" metadata. Such

metadata can either affect the structure of the data store, i.e., they are quality assurance data requirements, or expressed as what are often called standard operating procedures that guide the integrity of the data. Although these procedures are to be carried out at run time, they need to be considered during conceptual schema design.

The conceptual schema of a database is normally viewed as a formal *requirements specification* of the database [3]. Our approach is based on goal-oriented requirements analysis techniques for software design. Goal-Oriented Requirement Engineering (GORE) approaches [8,10,9,24,7] start with an early requirements step that focuses on modeling stakeholder goals, deriving from these both functional (i.e., application-specific) and non-functional (i.e., quality assurance) requirements through a systematic process. GORE provides a suitable framework for realizing the idea of *data quality by design* [23]. Quality goals (e.g., accuracy, security) are inherently "soft" in nature, i.e., there is neither clear cut definition nor criteria to decide whether a quality requirement is satisfied. GORE research has accumulated a wealth of notations, techniques and processes for modeling, analyzing and operationalizing high-level quality goals into operational and technical requirements (see, for example, [9]). Moreover, GORE supports representation and evaluation of alternatives in goal fulfillment. This feature allows one to perform cost-benefit analysis of various design alternatives to achieve the same application-specific goals, but with different level of support for quality goals. It also supports traceability of design decisions back to goals whose fulfillment was finally chosen by the human designer. Last but not least, formal goal reasoning tools are available to support various steps of the schema design process.

The GODB approach as described in [15] focuses on the application-specific (hard) goals. A general design strategy for dealing with any type of (including data quality) softgoals is also outlined and illustrated with a few examples of design operations. The open question however remains: how are these design operations defined in the first place? In this paper, we aim at answering this question by proposing the quality design process. This process starts with a set of *quality softgoals* and the *application-specific data requirements* obtained as described above; analyzes the data acquisition process for potential *risk factors* that may compromise the quality of the data of interest; identifies and selects potential *mitigation plans* against these risks; and finally merges the selected plans with those already in the goal model. The last step creates a new design alternative in the goal model that satisfies the previous top-level hard goals, but with better attention to the quality softgoals. The regular goal-oriented conceptual database design process then resumes from this new design alternative. The contributions of this work include:

1. a novel approach towards realizing the idea of data quality by design, borrowing ideas from *goal-oriented requirements analysis* paradigm for software design,
2. a wide coverage of *quality assurance data requirements*, both at the requirements analysis and conceptual schema design phases,
3. a systematic way to *operationalize* high-level, abstract quality goals into operational, concrete data requirements and standard operating procedures, based on a *risk-based analysis*, and
4. applicability of *formal reasoning tools* to support part of the design process.

These add to the benefits of the original GODB approach, which include the consideration of goals form *multiple stakeholders*, the *exploration and evaluation of alternative ways* to fulfill these goals, and the *explicit traceability* from higher level goals to technical design decisions.

## THE QUALITY DESIGN PROCESS

Before we describe the quality design process in detail, we first introduce the meeting expense database example and the diagrammatic goal modeling notation[1] that will be used throughout the rest of the paper.

[1] Note that the diagrammatic notation is intended solely for ease of use. There is a corresponding formal representation, which

## *The ExpDB Example*

Employees of a particular organization travel to cities in different countries, and participate in various meetings. A meeting expense management system monitors the spending on meetings in order to (a) reimburse its employees attending these meetings, (b) estimate the meeting budget for next fiscal year. Here we describe the design of a database component (called *ExpDB* thereafter) for a travel expense management system. This example is adopted from [9].

According to the GODB approach, the design of *ExpDB* starts by constructing a goal model. A goal model is a forest of goal/plan AND-OR decomposition trees with contribution edges between nodes of different trees and means-end edges connecting goal and plan nodes. A portion of the goal model is shown in Figure 2. The top-level hard goal *G1* is refined into sub-goals using AND-decomposition, which means that in order to fulfill this goal, one has to achieve all its sub-goals (i.e., *G1.1 ~ G1.3*). *G2* is OR-decomposed so that achieving any of its sub-goals is sufficient to fulfill the top goal. To achieve *G1.1*, a single plan *P1.1.1* is identified at this moment; it is linked to *G1.1* through a means-end edge, and is further refined into three subplans *P1.1.1.1 ~ P1.1.1.3*. Various contribution edges exist in the goal model. For example, the full, positive (shown as a dashed arrow labeled with "++") contribution from the goal *G1.2* to the goal *G2.1* means that the fulfillment of the former is considered sufficient to achieve the later. The partial, negative (shown as a dashed arrow labeled with "–") contribution from the plan *P1.2.1* to the softgoal *S1* means performing this plan contributes negatively (to some degree) to the satisfaction of the softgoal.

This goal model depicts a reimbursement process to fulfill the top-level goal *G1*: the employee collects the vouchers of expenses related directly to the meeting (e.g., travel, boarding, registration), and fills in a reimbursement request form with a summary of all the expenses. The employee then submits the form to the manager, who signs and forwards it to the secretary. The secretary is then responsible for entering the form into the system. The system finally generates an expense report at a specified time in a particular format, and issue reimbursement cheques accordingly.

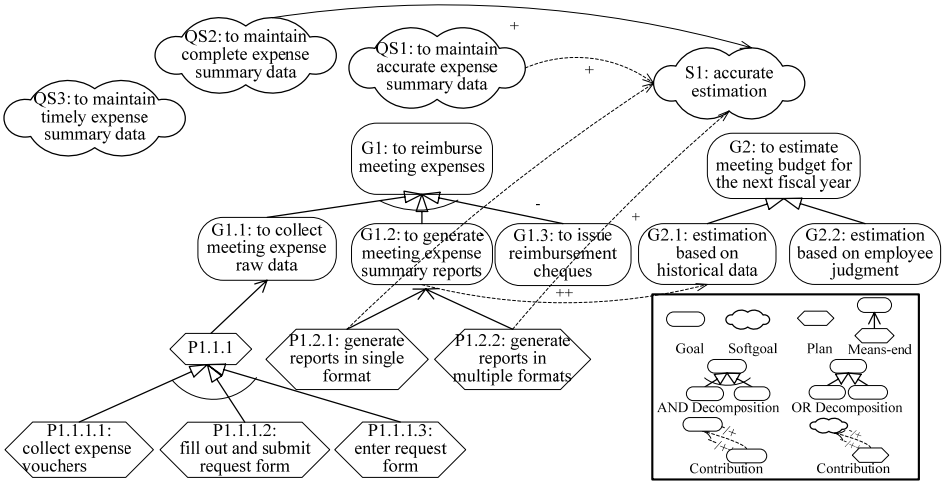


Figure 2. A portion of the goal model for *ExpDB*.

The goals and plans in the goal model "name" the concepts, relationships and attributes that can be systematically derived [15] to form the domain model for *ExpDB*. A portion of the domain model is shown in Figure 3. To support the reimbursement process described above, it is sufficient to include only

---

supports formal inference. See Step 6 in the next section for a discussion of the formal reasoning capability of goal-oriented approaches.

a few elements from the domain model in the final conceptual schema of *ExpDB*. One possible conceptual schema design is shown in Figure 4[2]. Although simple and intuitive, this design does not take into account nor respond to the quality softgoals (i.e., *QS1 ~ QS3*) at all.

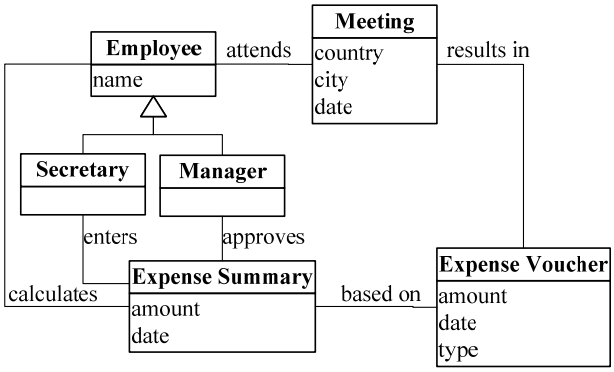


Figure 3. The domain model derived for the goal model.

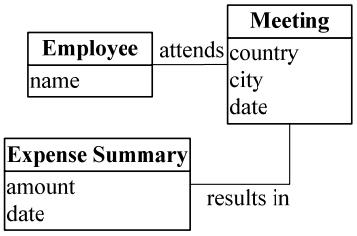


Figure 4. A conceptual schema for ExpDB without considering quality softgoals.

## *The Detailed Quality Design Process*

From the above discussion, hopefully the reader has got a sense how the GODB approach is used to derive an ordinary conceptual database schema from application-specific hard goals, *without* considering quality issues. When the quality softgoals in the goal model are also taken into consideration, other variants of this basic reimbursement process can be derived and added to the goal model, providing alternative ways to fulfill *G1.1*, with different levels of support to these softgoals. These variants may require additional concepts, relationships and attributes to be incorporated into the conceptual schema, leading to different schema design. The quality design process is all about how this is can be done in a systematic way. The following steps are defined to achieve this:

1. Characterize the data acquisition process
2. For each leaf-level quality softgoal *QS*
   2.1. Identify and characterize the application data concerned by *QS*
   2.2. Identify the risk factors to *QS* in each step of the data acquisition process
   2.3. Develop mitigation plans for each identified risk factor
   2.4. Identify lateral contributions from the mitigation plans to other quality softgoals
3. Evaluate and select mitigation plans to be supported by schema design
4. Integrate the selected plans to the original goal model, and follow the normal goal-oriented conceptual database design approach.

Below we illustrate each step with the *ExpDB* example, focusing on the accuracy softgoal (*QS1*).

***Step 1: Characterize the data acquisition process***

[2] In this paper, we use UML Class Diagrams to represent both domain models and conceptual schemas. For the sake of clarity, we omit details (e.g., attributes, constraints) in the diagrams that are not relevant to our discussion.

In a data acquisition process, an observer makes an observation, which may be recorded and manipulated before being entered into a database. Note that,

1. the observer, recorder and enterer have their own goals that may affect the objectivity of their respective tasks,
2. various instruments that are used during the process (e.g., observation instrument, recording media) may have certain limitations (or biases) due to their intrinsic properties (e.g., number of significant digits, error margin) or environment factors (e.g., time, location, altitude),
3. the database's ability to store the observation is limited by its schema design (e.g., presence or absence of certain fields, and the number of decimal places for numeric data fields).

In the *ExpDB* example, the employee plays both the roles of data observer and recorder, with the goal of maximizing meeting expense reimbursement. She (a) "observes" various expenses concerning meeting-related events, such as airline ticket purchase, hotel booking and meeting registration, (b) calculates the total amounts during the reimbursement request event, and (c) "records" detailed expenses on the original vouchers (or a separate piece of paper), and expense summary on the reimbursement request form. The secretary plays the role of data enterer whose main concern is to finish assigned tasks in an efficient way. Therefore, she (d) usually enters the meeting summary data in a batch mode. All these factors have the potential to reduce the quality of the observation being finally stored in the database.

***Step 2: Characterize application data***

Each quality softgoal has one or more *topics* that correspond to the application-specific data about which this quality is concerned. In this step, we characterize these application data along following dimensions:

- *Data value*: numeric, date vs. character-based, atomic vs. composite, primary vs. derived, etc.
- *Data domain*: enumerable vs. non-enumerable, standardized vs. non-standardized, etc.
- *Types of value defects*: inconsistent representation, syntactic vs. semantic, etc.

This characterization helps us understand the nature of these application data, identify various ways how the quality of the data can be compromised, and define quality assurance mechanisms. A part of this knowledge is application-independent, and therefore could be put into a library for later reuse. For example, composite values could be decomposed into their components, which are stored and verified separately (recall the person address example discussed in the introduction section). As another example, for application data with an enumerable and standardized domain, a control vocabulary could be used to ensure the syntactic (but not semantic) accuracy of the data at data entry time.

In the ExpDB example, the quality softgoal QS1 concerns meeting expense summary data. According the domain model (Figure 3), this includes the monetary amount of the total expense (*ExpenseSummary.amount*) and the date when it is reported (*ExpenseSummary.date*). The former has numeric, atomic and derived data values where its data domain is non-enumerable and non-standardized. Moreover, in this particular application, expense summary data are expected to be mainly syntactically valid (e.g., ensured by syntax checkers during data entry) but may be semantically wrong.

***Step 3: Identify risk factors***

Based on the characterizations of the application data and its acquisition process, the risk factors that may compromise the quality of the application data in each of the acquisition steps are identified. A few risk factors identified in the *ExpDB* example are shown in Table 1, some of which are further explained below. During observation time, because the ultimate goal of the employee is to maximize meeting expense reimbursement (which conflicts with one of the softgoals of *ExpDB*: accurate estimation of meeting budget for the next year fiscal year), the employee may report expenses that do not result directly

from the meeting (*R1*). During recording and manipulation time, since the meeting expense summary data is *derived* from expense detail data, there is a potential that the calculation may be wrong (*R2*). During data entry time, typographical errors are the most common sources of data defects in the database (*R5*). This is especially true when the secretary's private goal (i.e., efficiency) conflicts with the accuracy softgoal.

| |
|---|
| *During observation time:*<br>*R1:* The employee considers expenses that do not result directly from the meeting (e.g., visiting a nearby place or friend, before or after the meeting). |
| *During recording and manipulation time:*<br>*R2:* The employee miscalculates the total amount of the expense summary<br>*R3:* The employee fills in incorrect summary data in the request form<br>*R4:* The employee makes a reimbursement request long time after the trip and the original expense vouchers are lost |
| *During data entry time:*<br>*R5:* The secretary enters summary data incorrectly |

Table 1. Examples risk factors for *ExpDB*

***Step 4: Develop mitigation plans***

For each risk factor identified above, one or more mitigation plans can be developed to either (a) reduce the likelihood of occurrence of the risk factor, or (b) reduce its impact on the quality of data. Table 2 shows a few mitigation plans that are defined for the risk factors listed in Table 1.

| |
|---|
| *MP1:* Verify that any meeting expense occur within the meeting date ± one day. |
| *MP2:* Verify that meeting expense summary is consistent with expense details. |
| *MP3:* Verify that the reimbursement request date is within one month of the meeting date. |
| *MP4:* Require that expenses summary data be entered at least twice, possibly by different secretaries and/or at different times. |
| *MP5:* Perform audit where the manager periodically goes through a sample of expense summary data newly entered into the database in order to identify suspicious expense patterns, possibly with reference to the original expense vouchers. |

Table 2. Examples of mitigation plans for *ExpDB*

Note that *MP1*, *MP2*, *MP3* and *MP5* can be performed either manually by the manager or secretary before data entry, or automatically by the system at data entry time (or periodically). In most the cases, the automatic versions of these plans require additional data to be maintained by the *ExpDB*. For example, for *MP2*, it is necessary to enter both the meeting expense summary and expense voucher data into the *ExpDB* in order to perform the automatic verification of consistency between these two. Moreover, the manual versions of these plans may also lead to additional data requirements; this will be elaborated in Step 7. The correspondence between mitigation plans and the associated risk factors are shown in Table 3.

| Mitigation Plan | Risk Factors |
|---|---|
| *MP1* | *R1* |
| *MP2* | *R2 ~ R4* (performed manually), *R2 ~ R5* (performed automatically) |
| *MP3* | *R4* |
| *MP4* | *R5* |
| *MP5* | *R1~ R5* |

Table 3. Correspondence between mitigation plans and risk factors

***Step 5: Realize lateral contributions***

The mitigation plans identified above all contribute positively (with different degrees) to the satisfaction of the quality softgoal *QS1*. In this step, we try to identify positive / negative contributions from these mitigation plans to *other* softgoals in the goal model. First, mitigation plans *MP1 ~ MP3* contribute negatively to the quality softgoal *QS3* since the extra verification steps compete with the mainstream data acquisition steps for the manager or secretary's time and attention. This is true even if they are performed automatically by the system. The reason for this is that, as discussed previously, they all require extra data to be entered into the database in first place. Second, *MP2*, when performed automatically, contributes positively to the softgoal *S1*. This is because recording expense voucher data allows generating expense summary reports not only by employee and meeting, but also by type of expenses (e.g., hotel). Third, both *M2* and *M4* contribute negatively to the quality softgoal *QS2*. In the first case, employees who lose their expense vouchers cannot get reimbursement even if they remember the detailed expenses correctly; in the second case, the secretary may forget to re-enter the summary data causing that expense summary to be omitted when the summary report is produced. These contributions are shown in Figure 5 (contributions to *QS1* are omitted for the sake of clarity).

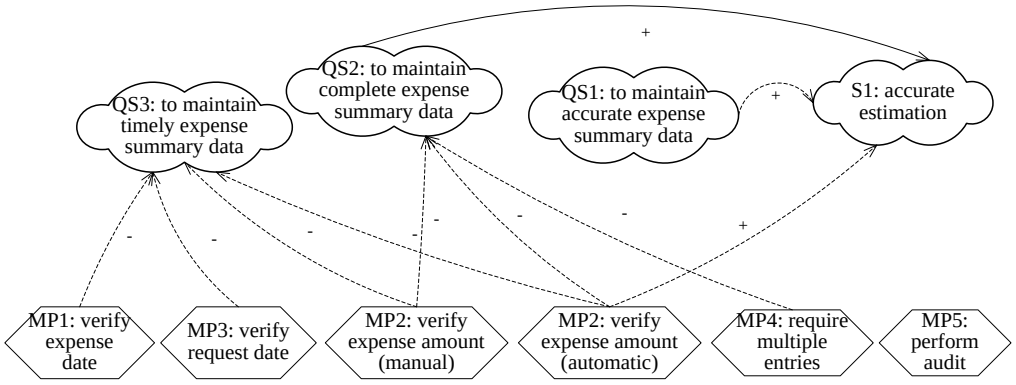

Figure 5. Lateral contributions from mitigation plans to softgoals.

***Step 6: Evaluate and select mitigation plans***

It is not always possible or desirable to simply integrate all identified mitigation plans into the data acquisition process because:

- For any DQ problem, the levels of tolerance may vary depending on the type of applications or type of application data [23]. For example, a 30-minute delay in stock price is more critical to a stock trading system than to market analysis application. Likewise, for a student registration system the accuracy of academic history data is more important than that of demographical data. In both cases, user requirements decide when a quality softgoal is sufficiently satisfied.
- There are may be multiple mitigation plans for the same risk factor, each with a different cost.
- Quality softgoals may conflict with one another, and a mitigation plan may have positive contributions to some softgoals and negative ones to others.
- Different risk factors may have different likelihoods of occurrence.

In summary, users' requirements for quality and cost-benefit tradeoffs need to be carefully evaluated when selecting mitigation plans. Evaluation can be carried out manually or automatically (given proper tool support). Only the chosen mitigation plans require schema design support.

**Formal Reasoning with Quality Softgoals, Risks and Plans**

It might seem that reasoning with quality softgoals, risks and mitigation plans is inherently qualitative, and therefore unsuitable for formal reasoning, especially in view of the possible *conflicting evidence*.

However, this appearance is deceptive. The Tropos project offers a formal framework [19] where one can encode diagrams such as Figure 2 into propositional logic by taking a component G (of any type) in the diagram, and instead of introducing a single propositional symbol *G*, use four symbols: *Partially_Satisfied_G*, *Fully_Satisfied_G*, *Partially_Denied_G*, and *Fully_Denied_G*. If *H* then negatively contributes to *G* in a weak manner (i.e., there is a contribution edge labeled with a "–" from *H* to *G*) then the propositional implication *Partially_Satisfied_H* $\rightarrow$*Partially_Denied_G* is added to the theory, among others, while if the negative link from H to G is strong (i.e., a "––" edge), then *Fully_Satisfied_H* $\rightarrow$ *Fully_Denied_G* is also added to the theory. Similar axioms are added for decompositions and means-ends edges in the goal model. One can now use standard propositional abductive reasoning to find, for example, minimal sets of mitigating plans that cover all or only the most important selected quality softgoals.

The Goal-Risk framework [1,2] extends the Tropos formal goal modeling and reasoning with risk-related concepts, such as risk events (and their likelihood of occurrence, and severity once occurred) and treatment plans (and their effectiveness to reduce the impact of risk events). It also proposes a risk analysis process that automatically selects a subset of all possible plans that satisfies the top-level hard goals, with total risk and cost below specified thresholds. Further work needs to be done to adapt this framework to support automatic evaluation of mitigation plans during conceptual schema design.

***Step 7: Integrate mitigation plans into a goal model***

Selected mitigation plans describe steps to be performed in addition to those in the original data acquisition process, with the purpose of providing quality assurance for the acquired application data. These steps are normally termed in practice *standard operating procedures* (SOPs). SOPs are integral part of the quality assurance process as they represent sequences of human and machine executed actions that guarantee the implementation of a quality property or policy. A database then needs to provide support to state an SOP (and its parameters) and record its execution (its values for each run). In this step, we merge the quality assurance process with the data acquisition process in the original goal model (Figure 2). For demonstration purpose, we assume all mitigation plans *MP1 ~ MP5* have been selected in the previous step. Figure 6 shows a portion of the resultant new goal model, rooted at Goal *G1.1*. In this goal model, a new plan *P1.1.2* is created by merging *P1.1.1* with *MP1* (manual), *MP2* (manual), *MP3* (automatic), *MP4* and *MP5* (manual); it provides an alternative way to achieve Goal *G1.1* with better attention to Softgoal *QS1*. In *P1.1.2*, modified or added subplans (compared to *P1.1.1*) are shaded.

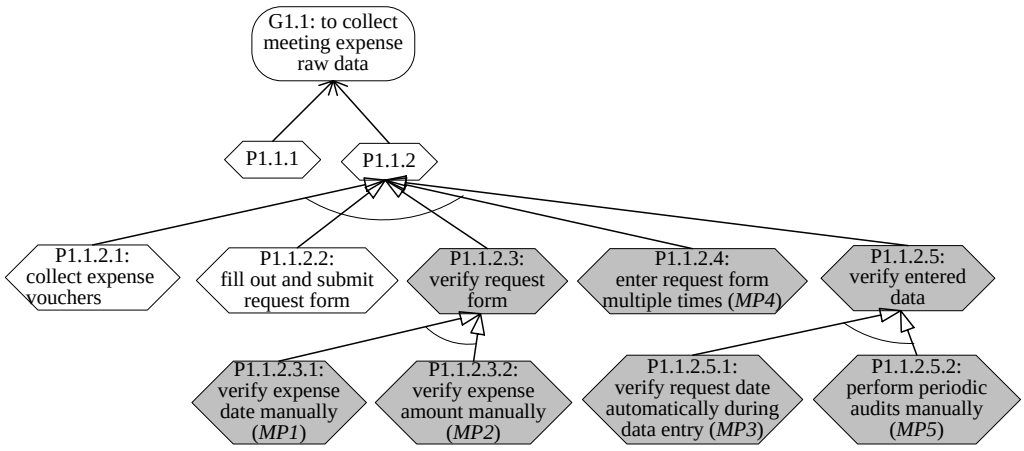

Figure 6. A portion of the new goal created by merging P1.1.1 and MP1 ~ MP5.

The improved reimbursement process can be described as follows: the employee collects the vouchers of expenses related directly to the meeting, and fills in a reimbursement request form with a summary of all

the expenses. The employee then submits the form and *all expense vouchers* to the manager. The manager signs and forwards them to the secretary who first checks *(a) if any expense voucher date is within the meeting date ± one day, and (b) if meeting expense summary is the sum of all the expense voucher amounts*. If no error is found, the secretary is then responsible for entering the form into the system *at least twice*. The system finally generates an expense report at a specified time in a particular format, and issue reimbursement cheques accordingly. The manager *occasional goes through a selected sample set of the expense summary data* newly entered into the database to identify suspicious expense patterns.

This new process requires new entities, relationships and attributes to be included in the conceptual schema, in addition to those shown in Figure 4. For example, although manual verification of expense date (P1.1.2.3.1) and expense amount (P1.1.2.3.2) does not require expense voucher data to be entered into *ExpDB*, the fact that the secretary has performed the verification process needs to be recorded. This can be done by including a *verifies* relationship with attributes *signature* and *date* between *Secretary* and *Expense Summary*, as shown in Figure 7.

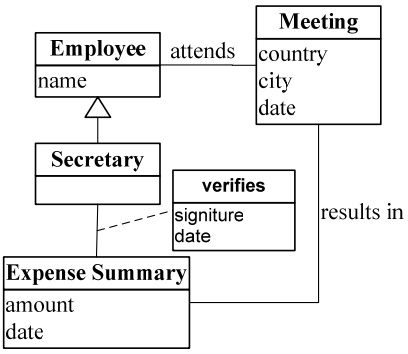


Figure 7. A portion of the conceptual schema of *ExpDB* that supports manual verification.

The final conceptual schema that supports all subplans of *P1.1.2* is shown in Figure 8. The entity *Confirmation* is used to record the number of times the same expense summary data has been entered by the secretary. The intention is that any expense summary data which has not been confirmed will be ignored by the application (e.g., when generating the expense summary reports). The entity *Audit* and its associated relationships are used to support the auditing activities performed the manager.

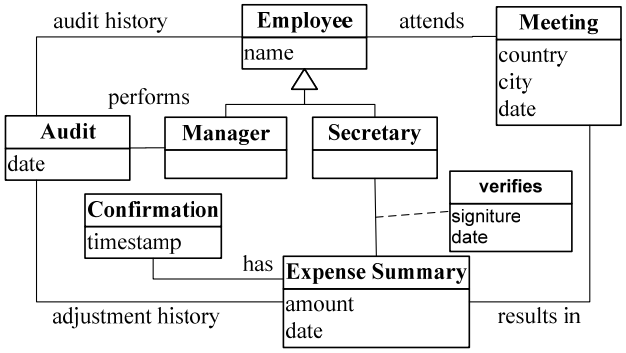


Figure 8. The final conceptual schema of *ExpDB* that supports *P1.1.2*.

## DISCUSSION: HOW THINGS FIT TOGETHER

In this section, we summarize important concepts that have appeared in previous discussion, and present a classification of quality assurance data requirements. We also show how the quality design process fits

into the overall GODB approach to offer an integrated framework for addressing both application-specific and quality assurance data requirements

## *Quality assurance data requirements*

From the discussion in the previous section, it is reasonable, for analysis purpose, to consider processes that satisfy stakeholder goals at three levels. At the topmost level, the business process (*BSProc*) represents the set of all activities performed by an organization in order to realize its value. During requirements analysis, an initial version of *BSProc* is obtained by analyzing the hard goals in the goal model, and is used to derive application-specific data requirements (*AppData*) for the database-to-be. In the *ExpDB* example, this is the reimbursement process we depicted in Figure 2 and the resulting *AppData* is in Figure 4. At the middle level, a data acquisition process (*DAProc*)[3] can be separated out from *BSProc* and undergoes a risk-based analysis. The result is a set of mitigation plans that can be used to provide quality assurance for the corresponding *AppData*. In the *ExpDB* example, the acquisition process for the meeting expense data is characterized in the first step of the quality design process, and the resulting set of mitigation plan is shown in Table 2. At the bottom level, the selected mitigation plans collectively characterizes a quality assurance process (*QAProc*). An analysis of this process produces quality assurance data requirements (*QAData*) to be combined with *AppData* identified earlier. In the *ExpDB* example, the *QAProc* for meeting expense data is depicted in Figure 6, and the derived *QAData* is shown in Figure 8.

We can further classify quality assurance data requirements into four categories:

- *Restrictive QAData* (*ResQAData*) are constraints on *AppData* that cannot be simply expressed as integrity constrains (e.g., key, cardinality constraints) in the conceptual schema, and often lead to elicitation of *metadata*. For example, the mitigation plan *MP1* ~ MP3 (see Table 2) imply three such constraints.
- *Descriptive QAData* (*DesQAData*) characterize activities in *QAProc*, providing evidence that these quality assurance activities have been carried out successfully. The relationship *verifies* in the final conceptual schema for *ExpDB* (Figure 8) is an example of *DesQAData*.
- *Supportive QAData* (*SupQAData*) represent extra data required or produced by *QAProc*. For example, the relationships *audit history* and *adjustment history* in the final conceptual schema for *ExpDB* (Figure 8) are both updated by each audit activity and used for the selection of sample data for the next audit (e.g., employees who have not been audited recently are likely to be included in the next audit).
- *Reflective QAData* (*RefQAData*) support recording of quality assessment (actually measured or estimated) for *AppData* in the database. This is the type of quality assurance data supported in [21].

## *The GODB Approach Extended*

The extended GODB approach, addressing both application-specific and quality assurance data requirements, is summarized in Figure 9. Our approach concurs with the *data quality separation principle* [21] which states that application-specific and quality assurance data requirements are modeled separately. Moreover, the *QAProc* derived from the risk-based analysis not only (a) adds *QAData* to the conceptual schema, but also (b) augment the initial *BSProc* with a set of SOPs for quality assurance. These SOPs accompany the conceptual schema at design time, and serve as the recipes that need to be followed at run-time. Therefore, the goal model *also* severs the purpose for documenting these SOPs and may be useful in quality assurance activities beyond schema design (e.g., in assigning responsibilities and

[3] Data maintenance and utilization processes also belong to this level; but here we focus only on quality of the data stored in a database.

monitoring performance of these SOPs at run-time).

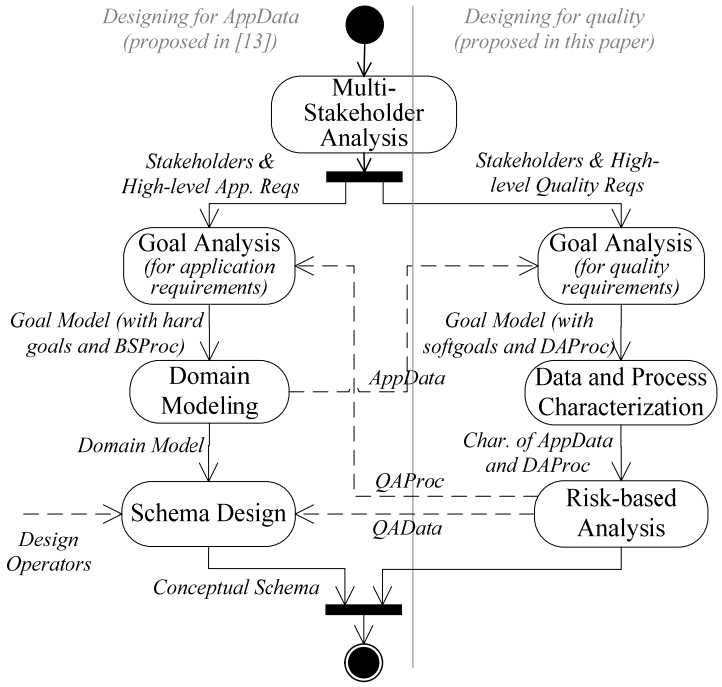


Figure 9. The extended GODB approach, covering both application-specific and quality assurance data requirements

# CONCLUSION AND FUTURE WORK

We have presented a novel approach towards realizing the idea of data quality by design, building on our previous work on the GODB approach. We draw ideas from Tropos, a goal-oriented software development methodology, and its extension for performing risk-based analysis. Quality is usually defined as *fitness for use*. This implies that quality of data should be evaluated in a way relative to the purpose of its use. [11] defines the term "relativity of data quality" as "a functional dependency of all its aspects on the *purpose and circumstances of operations* where those data serve as resources", and calls for *teleological* methods to DQ problems. Our goal-oriented approach provides exactly such a framework for analyzing purposes (modeled as goals), and their manifestation as operations (modeled as plans) in the context of database design, overcoming the limitations of existing data quality by design proposals.

The benefits of our extended GODB approach are discussed in the introduction section and summarized below:

- the formal modeling of *quality softgoals* of *multiple stakeholders*,
- the *systematic exploration and evaluation of alternatives* in goal fulfillment,
- the support for *explicit traceability* from higher level goals to technical design decisions,
- a broader coverage of *quality assurance data requirements* (restrictive, descriptive, supportive and reflective), thus addressing the limitation of [21],
- a *systematic goal operationalization* mechanism based on *risk analysis*, thus addressing the second limitation of [23][4], and
- the existence of *formal representation* of the diagrammatic models in our figures, and concomitant *reasoning tools* that support automating part of the design process.

[4] These limitations are discussed in the background section.

In the end, we offer an integrated framework for addressing both application-specific and quality assurance data requirements during database requirements analysis and conceptual schema design.

This work is being extended along several different directions. First, as mentioned earlier, an investigation of frequently occurring categories of DQ, together with standard risks and mitigation plans for them, will provide a library ("ontology") that can be the basis of a much more systematic, less omission-prone methodology, using a computer-supported tool to elaborate goal/risk/mitigation diagrams. Second, a full analysis in a practical setting may result in a very large and complex model of quality softgoals, risk factors, mitigation plans with various cost-benefit characteristics and interrelationships. Formal analysis tools are essential. The Goal-Risk framework [1,2] is intended for automated agents and does not distinguish application- and quality-related risk factors. Further research is required to tailor this framework to support human designers. Last, our approach to quality design can be extended to address data governance issues (e.g., privacy, security), where both data acquisition and utilization processes need to be analyzed.

# REFERENCES


[1] Y. Asnar, P. Giorgini, and J. Mylopoulos. Risk modelling and reasoning in goal models, technical report dit-06-008. Technical report, DIT - University of Trento, 2006.

[2] P. Giorgini, Y. Asnar. Modelling risk and identifying countermeasures in organizations. In *Proceedings of 1st International Workshop on Critical Information Infrastructures Security (CRITIS 06)*, page 5566. Springer, 2006.

[3] Paolo Atzeni, Stefano Ceri, Stefano Paraboschi, and Riccardo Torlone. *Database Systems - Concepts, Languages and Architectures*. McGraw-Hill Book Company, 1999.

[4] Donald Ballou, Richard Wang, Harold Pazer, and Giri Kumar Tayi. Modeling information manufacturing systems to determine information product quality. *Manage. Sci.*, 44(4):462–484, 1998.

[5] V.R. Basili, G. Caldiera, and H.D. Rombach. The goal question metric approach. In *Encyclopedia of Software Engineering - 2 Volume Set*, pages 528–532. John Wiley & Sons, Inc., 1994.

[6] Carlo Batini and Monica Scannapieco. *Data Quality: Concepts, Methodologies and Techniques (Data-Centric Systems and Applications).* Springer, 1st edition, 2006.

[7] Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004.

[8] J. A. Bubenko, D. Brash, and J. Stirna. EKD user guide. Technical report, Kista, Dept. of Computer and Systems Science, Royal Institute of Technology (KTH) and Stockholm University, Stockholm, Sweden, 1998.

[9] Lawrence Chung, Brian A. Nixon, Eric Yu, and John Mylopoulos. *Non-Functional Requirements in Software Engineering*. Kluwer Publishing, 2000.

[10] Anne Dardenne, Axel van Lamsweerde, and Stephen Fickas. Goal-directed requirements acquisition. *Sci. Comput. Program.*, 20(1-2):3–50, 1993.

[11] Zbigniew J Gackowski. Operations Quality of Data and Information: Teleological operations research-based approach, call for discussion. In Proceedings *of the 2005 International Conference on Information Quality (ICIQ'05)*, page 20-39, 2005.

[12] Richard Hull and Roger King. Semantic database modeling: survey, applications, and research issues. ACM Comput. Surv., 19(3):201–260, 1987.

[13] Matthias Jarke, Maurizio Lenzerini, Yannis Vassiliou, and Panos Vassiliadis. *Fundamentals of Data Warehouses*. Springer, 2nd edition, 2003.

[14] Manfred A. Jeusfeld, Christoph Quix, and Matthias Jarke. Design and analysis of quality information for data warehouses. In *Proceedings of 17th International Conference on Conceptual Modeling (ER'98)*, pages 349–362, 1998.

[15] Lei Jiang, Thodoros Topaloglou, Alex Borgida, and John Mylopoulos. *Goal oriented conceptual database design. In Proceedings of the 15th IEEE International Requirements Engineering Conference (RE'06)* (to appear), 2007.

[16] Axel van Lamsweerde and Emmanuel Letier. From object orientation to goal orientation: A paradigm shift for requirements engineering. In *Proceedings of the Monterey'02 Workshop*, pages 4–8. Springer-Verlag, 2003.

[17] Shamkant B. Navathe. Evolution of data modeling for databases. *Commun. ACM*, 35(9):112–123, 1992.

[18] Jack Olson. *Data Quality: The Accuracy Dimension*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[19] Roberto Sebastiani, Paolo Giorgini, and John Mylopoulos. Simple and minimum-cost satisfiability for goal models. In Advanced Information Systems Engineering, 16th International Conference, CAiSE 2004, Riga, Latvia, *volume 3084 of Lecture Notes in Computer Science*, pages 20–35. Springer, 2004.

[20] G. Shankaranarayanan, Richard Y. Wang, and Mostapha Ziad. Ip-map: Representing the manufacture of an information product. In *Proceedings of the 2000 Conference on Information Quality*. MIT, 2000.

[21] Veda C. Storey and Richard Y. Wang. Modeling quality requirements in conceptual database design. In *Proceedings of Third Conference on Information Quality (IQ 1998)*, pages 64–87, 1998.

[22] Richard Y. Wang. A product perspective on total data quality management. *Commun. ACM*, 41(2):58–65, 1998.

[23] Richard Y. Wang, Henry B. Kon, and Stuart E. Madnick. Data quality requirements analysis and modeling. In *Proceedings of the Ninth International Conference on Data Engineering*, pages 670–677, Washington, DC, USA, 1993. IEEE Computer Society.

[24] Eric S. K. Yu. Towards modeling and reasoning support for early-phase requirements engineering. In *Proceedings of the 3rd IEEE International Symposium on Requirements Engineering (RE'97)*, page 226,Washington, DC, USA, 1997. IEEE Computer Society.