A Homogeneous Framework to Measure Data Quality

Mónica Bobrowski **Martina Marré** **Daniel Yankelevich**
Departamento de Computación - FCEyN
Universidad de Buenos Aires
Pabellón I - Ciudad Universitaria
(1428) Buenos Aires, Argentina
++ 54 1 4576 3359
{monicab, martina, dany}@dc.uba.ar

ABSTRACT

Thirty years ago, software was not considered a concrete value. Today, software is (even formally) an asset for any organization. Data is slowly following the same path. The information owned by an organization is an important part of its assets. Information can be used as a competitive advantage. However, the software engineering community has long ignored data. Usually, methods and techniques apply to software (including data schema), but the data itself has often been regarded as external. Validation and verification techniques usually assume that data is provided by an external agent, and concentrate only on software. The first step to define methods and techniques to improve data quality is to measure the value of information. It is impossible to make empirical evaluations of data quality if we do not agree on what (and how) should be measured. In this work, we apply traditional software metrics techniques to measure data quality, following the Goal-Question-Metric (GQM) methodology. The outcome is a suitable set of metrics that establishes a starting point for a systematic analysis of data quality. Moreover, we perform an experiment to validate the idea of using standard software engineering tools to attack data quality issues.

Keywords

Data Quality, GQM, Data Metrics

1 INTRODUCTION

Thirty years ago, software was not considered a concrete value in an organization. Everyone agreed on the importance of software, on its virtual value, but it was not regarded as a good, as a possession. In those days, the value of software was associated with its cost. Nowadays, software is part of the balance of an organization, it contributes to its value, and we calculate the return of investment (ROI) of virtually every project.

Data is slowly following the same path. Thus, managers know that having the right information at the right time may lead to great benefits. Data can be used as a competitive advantage. This data is usually stored in large databases, and managed via software applications. However, it is not enough to have good applications: an organization needs good data to achieve its goals.

The quality of the data owned by an organization crucially affects how useful this data is. Information of poor quality will typically be used little, or may lead to incorrect or harmful decisions. "Decisions are no better than the data on which they are based" [12]. But, what does data quality mean? How could the quality of the data in an organization be measured, in order to decide if data can be trusted?

Many of the problems that result from using bad data are well known to software engineers. However, the software engineering community has long ignored data quality issues. Usually, methods and techniques apply to software (including data schemas), but the data itself has often been considered an external problem. Quality techniques for validation and verification usually assume that an external agent provides data, and concentrate only on the software proper.

Our claim is that software engineering should encompass data quality issues, to prevent, detect, and solve system problems resulting from data of poor quality. Some of these problems originate in human errors, while others can be addressed using standard software techniques. We propose this "software engineering

view" in [4]. Our position is that many data quality problems derive from bad software engineering practices. Hence the software engineering community should deal with these problems.

Recently, researchers have been studying data quality problems from the perspective of the data generation processes [18, 14, 15]. They have identified problems in the data, and tried to associate them with problems in the process that generated this data. The underlying idea is that improving the data generation and manipulation process may lead to an improvement in data, likewise the improvement of the software development process leads to an improvement of the software product.

Redman [11] gives an introduction to data quality (DQ) issues. He highlights the importance of understanding data quality requirements from the beginning of the software process, but from a management perspective. He separates data issues from software issues. He deals mainly with data, not with applications. Although data is independent from applications at a certain level, we cannot ignore that data is generated, used, and maintained by applications. As software engineers, we must have in mind data issues when designing and implementing our software systems. Moreover, the intended use of the data helps us understand systems too. We believe that certain attributes are independent from the intended use of the data, but others are not.

Redman uses the data tracking technique (well known by statistical practitioners) to measure data quality. He does not propose a way to generalize metrics, and abstract from particular problems (in fact, Redman really does not need this generalization for his purposes). Our goal is to present a general framework where we could define metrics for a large variety of applications.

Other approaches [14, 15] have a similar management perspective. They use ad-hoc metrics for specific database applications (medical, customer applications, etc.), and rely on user satisfaction as a measure of quality.

Huang et al. [8], present a tool that combine integrity constraints in relational databases with the TDQM cycle to measure the quality of data. This otherwise interesting idea has limitations and lacks a formal justification of its measures. We want to complement those metrics with more complete and less subjective measures, when possible, following a top-down approach based on the dimensions of interest.

The only concern for computer engineers regarding data quality has been the extraction of data for data warehouses. In the context of Data Warehousing, a European project investigated quality and in particular the requirements on data needed to implement a data warehouse [9]. Also reference [2] presents a model to support quality enhancement in data warehouse environments.

A classical approach to measure information is given by information theory [13]. Information is associated with the probability of a message. Hence, following this approach, the book that gives the most information in a fixed length is a random book – since it cannot be compacted, it has no redundancy. In contrast, such a book would be useless for our purposes, giving no information at all. This suggests that our approach is based on the effect that information causes in something else. Formally, we are giving a nonstandard semantics to information – a pragmatic semantics, based on usability – to measure the amount of information of a content. This pragmatic view is the basis of our approach: the value of information depends on how this information will be used.

We measure information quality with metrics defined using traditional software engineering techniques. Metrics have been deeply studied in software engineering [6]. In particular, we base our work on the GQM methodology [3]. We believe this top-down approach to metrics definition may lead to a useful set of metrics, based on dimensions that may be part of the data quality notion. Our work is related to the internal view of the information system perspective pointed out in [8] (oriented toward system design and data production), but it has some aspects of the external view too (use and effect). However, we always work with data quality issues from a software engineering perspective, taking into account the issues software engineers may help to improve. Our goal is not to have perfect quality, but to have better quality. Moreover, we want to give a systematic way of deriving measures for quality, not only to give measures.

We cannot measure data and ignore how it is organized. Certain quality characteristics are related to the organization of data, i.e., to the *data model*, and not to the data itself. The data model might affect some data quality attributes, since it defines the way data is accessed and maintained. We want to identify and measure those attributes too, and complement measures of data with information on how it is organized.

Our main scope is to apply these measurement techniques to data in digital format. This data is usually organized in databases, at least at a logical level. Thus, we concentrate on models for databases [16, 1]. Our goal is to implement metrics at least in the case of the relational model, but the measures defined are general enough to be used in other cases.

In this paper, we present a framework for defining and using data quality metrics. We introduce a set of metrics that constitute a promising starting point for a systematic analysis of data quality. Moreover, we perform an experiment to validate the idea of using standard software engineering tools to attack data quality issues.

In Section 2 we discuss what data quality means. In Section 3 we present GQM. In Section 4, we outline some issues in the application of the GQM methodology to define data quality metrics. Also, we present some of the metrics so defined. This will be done using a simple case study. We will also include the results of this experiment.

2 What is Data Quality?

It is difficult to give a universal definition of what quality means. When we talk about quality we do not always refer to the same concept. The word *quality* by itself does not have a unique connotation. We have to make assumptions on which aspects apply to a particular situation. In the case of *data* quality, we may want to take into account only specific data attributes with some specific relevance, depending on the particular context we are analyzing. In our view, the quality of data in the context of software systems is related to the benefits that the data might give to an organization.

As mentioned above, the quality of data depends on several aspects. Therefore, in order to obtain an accurate measure of the quality of data, one have to choose which *attributes* to consider, and how much each one contributes to the quality as a whole. In what follows, we present several attributes we think have to be measured in order to determine the quality of our data. These attributes, or *dimensions,* have been taken from [18, 14] following the point of view of the value of the data, i.e., our pragmatic view of data quality.

We now present an informal definition for each of the attributes considered in this paper. These definitions will serve as the basis to define our new metrics (a more extensive list is given in [14]).

Completeness Every fact of the real world is represented. It is possible to consider two different aspects of completeness: first, certain values may not be present at the time; second, certain attributes cannot be stored.

Relevance Every piece of information stored is important in order to get a representation of the real world.

Reliability The data stored is trust worthy, i.e., it can be taken as true information.

Consistency There is no contradiction in the data stored.

Correctness Every set of data stored represents a real world situation.

Timeliness Data is as up to date as needed.

Precision Data is stored with the precision required to characterize it.

Conciseness The real world is represented with the minimum information required for the goal it is used for.

Getting a "high score" in one dimension does not imply that good quality has been achieved. For example, the timeliness of data may be important only in conjunction with correctness (up to date but incorrect data has no sense, and even may damage the organization).

To apply GQM (Section 3) we classify dimensions as either direct or indirect. Direct dimensions can be measured directly from the application of metrics. Indirect dimensions get their values from those of other dimensions. This notion captures the fact that all dimensions are not independent.

In other words, our metrics define a metric space over each dimension. Indirect dimensions are those whose metric space depends on other dimensions (i.e., not part of a basis).

Following this idea, we define a standard set of dimensions to be used in every data quality measurement process, composed by:

Direct Dimensions	Completeness, Relevance, Consistency, Correctness, Timeliness, and Precision.
Indirect Dimensions	Conciseness (depends upon Relevance), Reliability (depends upon Completeness, Conciseness, Correctness, Timeliness, Precision, and Consistency).

3 Elements of the GQM Approach

GQM [3] is a framework for the definition of metrics frequently used by the software engineering community. GQM assumes that to measure (software, processes, and so on) in a useful way, an organization must:

- specify goals,
- characterize them by means of *questions* addressing their relevant attributes,
- give measurements that may answer these questions.

We have chosen this framework because it is a top down approach that provides guidelines to define metrics, without a-priori knowledge of the specific measures. Following GQM, we first state which dimensions characterize our notion of data quality. Then, we can ask questions characterizing each dimension, without giving a precise (formal) definition, and only focusing on characteristics that are relevant from our point of view. This is important because sometimes a precise definition might be difficult or even impossible to give. We do not want to cope with semantic issues on dimensions: our goal is to obtain useful metrics. Finally, we derive metrics (some objective, others based on people appreciation) to answer these questions, giving us a more precise evaluation of the quality of our data.

A *goal* in GQM is defined in a precise way, and is posed at a conceptual level. A goal is given for an *object*, with a *purpose*, from a *perspective*, in an *environment*. An example of a goal in a software organization can read: "To evaluate the maintenance process from the manager point of view in the context of a maintenance staff comprised of new programmers." In this example, the object is the maintenance process, the purpose is to evaluate, the perspective is the manager's point of view, and the environment is the composition of the maintenance staff. The notions of perspective and environment are crucial, because they place the metric in a specific context. It is immediate that a metric can be interpreted in different ways, depending on who is reading it and the meaning (s)he gives to the isolated number. But the whole approach of metrics -of measuring- is based on the idea that it is helpful to have measures, i.e., to define a metric space over the dimensions of interest. This kind of abstraction performs a dangerous simplification. To overcome the potential problems of this loss of information we need to know in which context each metric will be used. This context is given by (a.) who is going to use it and with which purpose (the perspective) and (b.) some details of the environment that are assumed to be fixed during the application of the metric and that influence the metric definition (the environment).

A *question* in GQM tries to characterize the object of measurement with respect to a selected quality issue, and to determine its quality from the selected viewpoint. An example of a question can read: "What is the current change processing time?" A question in GQM is posed at the operational level, contrasting with the conceptual nature of goals.

A *metric* in GQM is a set of data associated with every question in order to answer it in a quantitative way. Data can be *objective*, if it depends only on the object being measured and not on the viewpoint, or *subjective*, if it depends on both. An example of a metric: "Number of days spent on a user change request" may be a metric for the question presented above. A metric in GQM is posed at the quantitative level. Here, in order to have a concrete way to compute the metrics, we also give techniques associated with them.

Data collection forms (DCF) is a technique for collecting information from users, in order to compute some of the subjective metrics defined. DCF have questions to be answered by data users. They are intended to measure aspects that depend on user appreciation (sense, accessibility, etc.). The answers are often predefined ordinal metrics (including explanations for each choice, as in "1. - Low, 2. -Medium, 3. - High"), true-false, yes-no, or an absolute measure (e.g., the answer for the question "how many times do you consult the data per day?").

Of course, this approach is an alternative to an ad-hoc definition of metrics. We seek a sound justification for each metric proposed. This justification will be contradicted or supported by empirical evidence after experimentation. The way we construct the metrics ensures that every proposal comes with a justification.

There are other approaches for metric definition, e.g., [5, 10]. We have chosen GQM because of its simplicity, because it is well known and proven in software engineering applications [17], and because it fits our problem nicely, as we will show in the rest of the article.

4 Deriving and Applying Data Quality Metrics

In this section, we outline some issues on the application of the GQM methodology to define data quality metrics. Also, we present some of the metrics so defined. This will be done using an example. Thus, we will also include the results of this experiment.

We do not base our proposal on this experiment. As we mentioned before, by construction, each metric is derived top/down from goals and questions. However, we think a first feasibility experiment is needed after going further in the implementation of metrics or investing on a more complete project. We have chosen a simple case, as a starting point.

First, we describe the main characteristics of the experiment. Then, we apply the GQM framework to define data quality metrics, focusing on the dimensions used in the experiment. We present how the resulting metrics were applied in this case. Last, we include the main results and conclusions obtained.

The Experiment

We wanted to evaluate the quality of the data in the database conformed by adding the information of five electronic personal organizer. This is a simple case study, that illustrates important data quality issues:

- As users enter data using different methodologies, or even with no criteria at all, the data stored may have quality problems.
- The agendas themselves do not have format or content validations.

These issues let us validate and even refine our proposal to obtain more accurate results.

We saved the information in the telephone directories of the personal organizers in computer files. Then, we modified these directories to make them uniform in format. The end result was a relational table with all this information. We loaded this table into a relational database management system. The users were asked about how they use their organizers, so that we could determine which functions were used the most, and which data quality dimensions to consider.

Applying GQM

We now describe how to apply the GQM methodology in the context of data quality metrics definition. We give the general framework and we show how we applied it in our experiment.

As mentioned above, a goal in GQM is defined for an *object*, with a *purpose*, from a *perspective*, in an *environment*. When following GQM to define metrics for data quality, we have identified two main **objects** to be measured: the *set of data* and the *data model*. The set of data is the data actually stored in the database. The data model is how data is structured, from an implementation or a logical point of view. When defining subjective metrics, our **perspective** is always the user point of view. In fact, we are trying to measure the quality of data with respect to the organization benefits. Each goal is defined in a particular **environment**. Elements of each particular environment can be among the following:

- a fixed data set (for example, the data obtained from the agendas);
- a fixed data model (for example, the relational model we obtained in our experiment);
- a fixed query set: Some of the dimensions are related to the way data may be queried. The *query set* is a set of queries of interest. Such a set can be inferred from the applications that use or generate the data, from the intended use of data, or from the procedures the data is involved in. The set of queries is a part of the context in which data can be used and is an explicit representation on what the organization wants to use data for.
- a fixed set of attributes: A subset of the database attributes that is of interest for a particular metric. This set may be defined as "temporal", for instance if they have deadlines, i.e., they have to be updated in order to be accurate. To deal with time, we need to identify this subset and give a procedure to calculate, or estimate, when the data is out of date. Other attributes might be related to particular issues, such as precision.

We have grouped the dimensions according to the object they are related to. As a consequence, we have two groups of dimensions: data dimensions, and data model dimensions. This classification may be useful when trying to correct the quality problems found. For our purposes, we have chosen dimensions in both sets. However, we have found it more interesting and more original to deal with set of data dimensions. In fact, model dimensions have been largely studied in software engineering as part of data bases definitions, e.g., [16, 1], although our point of view is slightly different.

Next, we present some data quality metrics we defined using the GQM framework for those dimensions presented in Section 2, in the context of our experiment. First, we introduce several goals that we have identified for a particular dimension. Then, we include questions and metrics for each goal. Finally, we give an operational technique to compute the associated values. In effect, we need systematic ways to measure. However, in some cases there is no operational way to measure a specific item, or the one we identified is too complex to use. In our experiment, we found that the personal organizers do not save logs of the database activities. We believe that having such logs could be very useful to measure some quality attributes, or to simplify some of the techniques proposed. Specifically, many of the metrics implemented with DCF's would benefit from this feature.

It is important to notice that we are not trying to be exhaustive in giving all the possible goals, questions, and metrics for every data quality dimension, not even for our particular case study. New dimensions can be used, new questions can be asked, new metrics can be given, and different techniques can be used. Moreover, through experimentation we may find problems, and decide to refine, change, add, or delete some of the metrics or even some of the dimensions.

Instance Dimension Metrics

In Table 1 we show the application of GQM to derive some data quality metrics for this case study for the object *set of data.*

GOAL	QUESTION	METRIC	TECHNIQUE
Object: Set of data **Purpose:** Evaluate **Quality:** *Completeness* **Perspective:** User **Environment:** -fixed data set - fixed query set	Is all the information necessary to represent the real world stored?	% of queries with no answer (considering only queries relevant for the organization)	Number of queries with no answer * 100 / number of queries considered
Object: Set of data **Purpose:** Evaluate **Quality:** *Relevance* **Perspective:** User **Environment:** -fixed data set	Is all the data in the database relevant for the organization?	% of entries distinguished by the users as no used	Number of entries no used * 100 / (number of attributes * number of records revised)
Object: Set of data **Purpose:** Evaluate **Quality:** *Reliability* **Perspective:** User **Environment:** -fixed data set	Can we trust that the answers obtained represent the real world?	This metric will be defined in relation to *completeness, conciseness, correctness, timeliness, precision, and consistency*	100 - (100 - *completeness* + 100 - *conciseness* + MAX(100 - *consistency,* 100 - *timeliness,* 100 - *precision,* 100 - *correctness*))
Object: Set of data **Purpose:** Evaluate **Quality:** *Consistency* **Perspective:** User **Environment:** -fixed data set	Does contradiction between data exist?	Select a fixed data set and verify that inconsistencies between data in that set do not exist	Number of inconsistencies detected * 100 / (number of attributes revised * number of records revised)
Object: Set of data **Purpose:** Evaluate **Quality:** *Correctness* **Perspective:** User **Environment:** - fixed data set - a fixed set of attributes	Does data represent the real world?	Select a fixed data set and verify data either with the user or using real values	Number of errors detected * 100 / (number of attributes revised * number of records revised)
Object: Set of data **Purpose:** Evaluate **Quality:** *Timeliness* **Perspective:** User **Environment:** - fixed data set	How much data has passed its deadline?	Number of records with at least one attribute in *T* not updated (per time unit) / Number of records with at least one	Number of entries not actualized * 100 / (number of attributes revised * number of records revised)

- *T*: Set of temporal attributes		attribute in *T*	
Object: Set of data **Purpose**: Evaluate **Quality**: *Precision* **Perspective**: User **Environment**: - set of attributes that should be stored with some precision	Is data stored with the necessary precision?	Selection and verification of data by users or real values	Number of entries stored with not enough precision * 100 / (number of attributes revised * number of records revised)
Object: Set of data **Purpose**: Evaluate **Quality**: *Conciseness* **Perspective**: User **Environment**: - fixed data set	Is some data stored never used?	% of attributes distinguished by the user as not used (relevance)	% relevance
	Is there data providing no further information?	% duplicated data	# duplicated data * 100 / # attributes revised

Table 1: Derivation of data quality metrics for the *set of data* object

To measure data *completeness*, each user received a DCF. Each DCF was completed with all the names and associated data that the user considered relevant enough to store in the personal organizer. After that, we have queried the database to know whether that data was actually stored in the agenda. The percentage of completeness is given by:

$$\frac{\text{number of pieces of data not stored in the database} * 100}{\text{number of relevant data}}$$

Each user received a DCF containing some samples from the database, in order to measure data *relevance*, *correctness*, and *timeliness*. The database was ordered in different ways, and each time some data was selected. Duplicates were eliminated. In this way, samples of approximately 130 records were created.

The user revised and decided the *relevance* of each attribute instance in the DCF. The percentage of relevance is given by:

$$\frac{\text{number of entries not used} * 100}{\text{number of pieces of data revised}}$$

The user revised and decided the correctness of each attribute instance in the DCF. The percentage of correct data is given by the percentage of correct instances over the total number of instances revised, minus the percentage of data stored with no enough precision. Correctness is given by:

$$\frac{\text{number of errors detected} * 100}{\text{number of records revised} * \text{number of attributes revised}}$$

The user was asked to distinguish among data which is out of date, and data which is incorrect. Timeliness is given by:

$$\frac{\text{Number of entries not actualized} * 100}{\text{number of attributes revised} * \text{number of records revised}}$$

As we mentioned in Section 2, *reliability* is an indirect metric. In this case, reliability is given by:

$$100 - (100 - \text{completeness} + 100 - \text{conciseness} + \text{MAX}\{100 - \text{consistency}, 100 - \text{timeliness}, 100 - \text{precision}, 100 - \text{correctness}\})$$

To measure data *consistency*, we considered all the records in the database. We obtained two different measures that we combined to obtain the percentage of consistency. First, we considered the case in which the same name contained different associated data. These two or more records could be consistent (if the associated data is the same, or if one record complements the others), or incorrect (if the associated information is not consistent). Second, the same analysis was carried on for two or more records containing the same telephone number. Consistency is given by:

$$\frac{(\text{number of inconsistencies w.r.t. names} + \text{number of inconsistencies w.r.t. telephones}) * 100}{\text{number of records revised} * \text{number of attributes revised}}$$

To evaluate data *precision*, we use two heuristics. On one hand, those records with telephone field not numeric or containing at least 4 bytes long were checked with the user. All these entries contained no precise data. On the other hand, each record with no telephone number was checked with the user to understand if it was correct, incomplete, or the data was stored in an incorrect way. The records stored in an incorrect way were considered stored with no enough precision. Precision is given by:

$$\frac{\text{Number of entries stored with no enough precision} * 100}{\text{number of attributes revised} * \text{number of records revised}}$$

Conciseness is given by:

$$\frac{(\text{number of entries not used} + \text{number of entries duplicted}) * 100}{\text{number of attributes revised} * \text{number of records revised}}$$

Model Dimension Metrics

In Table 2 we show the application of GQM to derive some data quality metrics for this case study for the object *data model.*

GOAL	QUESTION	METRIC	TECHNIQUE
Object: Data model **Purpose**: Evaluate **Quality**: *Relevance* **Perspective**: User **Environment**: -fixed data model -fixed query set	Are there attributes that are never accessed?	Number of attributes never accessed	Number of attributes in the schema never used * 100 / number of attributes in the schema
Object: Data model **Purpose**: Evaluate **Quality**: *Completeness* **Perspective**: User **Environment**: -fixed data model	May all the data be represented in the model?	Number of times data could not be stored in the database (because it does not fit in the model)	DCF

Table 2: Derivation of data quality metrics for the *data model* object

In this case the schema was very simple, and no consistency problems between attributes existed. So, *conciseness* is equivalent to *relevance.*

The *relevance* of the model was evaluated by asking the users which attributes of the schema they never used. So, relevance is given by:

$$\frac{\text{number of attributes of the schema never used} * 100}{\text{total number of attributes in the schema}}$$

To measure data *completeness*, users were asked whether they tried to store data that cannot be stored in the database. This DCF was also used to ask what type of information they usually stored in the free fields of the database.

Results and conclusio

When applying our metrics to ou

With these numbers in mind, we
are completeness, conciseness, a
their organizers: they do not ent
standard ways to enter it, and the

With respect to the data model,
complete. Thinking about agen
need more predetermined attribu

Analyzing these results, we may
on the cost/benefit ratio. In our
users. Some of the alternatives a
order to consider new predefine

5 Conclusions and F

In this work, we introduced so
particular, we emphasized the in
known software engineering m
metrics. We identified the attrib
to calculate them. To validate
measures obtained reflected the
framework is adequate for the
extensive experiments to validat

This is a starting point for a syst
in order to justify the measures
start a case study of an importan

We have used a software engine
other software engineering cor
verification of a system must i
precisely what data testing mean

The dimensions we measure tc
requirements. These requiremer
process. In fact, they may be co
to define and to verify them at e

The metrics presented above al
dynamic situation, after the dat
deletions, and updates. A promi
quality of the data with respec
accuracy improvement. Other di

ACKNOWLEDGEMENTS

This research was partially supported by the ANPCyT under grant ARTE Project, PICT 11-00000-01856 and grant PICT 11-00000-00594. We would like to thank Silvia N. Morillaz for her help in data collection and preparation. We also thank Luis Gravano for his comments.

REFERENCES

[1] Abiteboul, S., Hull, R., Vianu, V.: *Foundations of Databases*, Addison-Wesley, Reading, MA, 1995.

[2] D. Ballou, G. Tayi: Enhancing Data Quality in Data Warehouse Environments, *Communications of the ACM*, Vol. 42, No. 1, January 1999.

[3] Basili, V.R., Rombach, H.D.: The TAME Project: Towards Improvement-Oriented Software Environments, *IEEE Transactions on Software Engineering*, vol. 16, no. 6, June 1988.

[4] Bobrowski M., Marré, M., Yankelevich, D.: A Software Engineering View of Data Quality, *Proceedings of 2nd European Quality Week*, November 1998, Brussels, Belgium.

[5] Boehm, W., Brown, J.R., Lipow, M.: Quantitative Evaluation of Software Quality, *Proceedings of the Second International Conference on Software Engineering*, 1976.

[6] Fenton, N.E., Pfleeger, S.L.: *Software Metrics - A Rigorous & Practical Approach*, 2nd edition ITP Press, 1997.

[7] Haebich W.: A Quantitative Model to Support Data Quality Improvement, *Proceedings of the Conference on Information Quality*, MIT, Boston, October 1997.

[8] Huang, K-T., Lee, Y., Wang, R.: *Quality Information and Knowledge*, Prentice-Hall, 1999.

[9] Jarke M., Vassiliou Y.: Data Warehouse Quality: A Review of the DWQ Project, *Proceedings of the Conference on Information Quality*, MIT, Boston, October 1997.

[10] McCall, J.A., Richards, P.K., Walters, G.F.: *Factors in Software Quality*, Rome Air Development Center, RADC TR-77-369, 1977.

[11] Redman, T.: *Data Quality for the Information Age*, Artech House, 1996.

[12] Redman, T.: The Impact of Poor Data Quality on the Typical Enterprise, *Communications of the ACM*, Vol. 41, No. 2, pp. 79-82, February 1998.

[13] Shannon, C.E., Weaver, W.: *The Mathematical Theory of Communication*, University of Illinois Press, Urbana, Ill, 1949.

[14] Strong, D., Lee, Y., Wang, R.: Data Quality in Context, *Communications of the ACM*, Vol. 40, No. 5, May, 1997.

[15] Strong, D., Lee, Y., Wang, R.: 10 Potholes in the Road of Information Quality, *IEEE Computer*, August 1997.

[16] Ullman, J.D.: *Principles of Database and Knowledge-Base Systems*, Computer Science Press, New York, 1989.

[17] Van Latum F., Van Solingen R., Oivo M., Hoisi B., Rombach D., and Ruhe G.: Adopting GQM-Based Measurement in an Industrial Environment, *IEEE Software*, pp. 78-86, January-February 1998.

[18] Wand, Y., Wang, R.: Anchoring Data Quality Dimensions in Ontological Foundations, *Communications of the ACM*, Vol. 39, No. 13, November 1996.