International Journal of Scientific and Technical Advancements

ISSN: 2454-1532

Measuring Data Quality through Traditional Software Metrics Techniques

Balvir Singh[1], Amarjeet singh[2]

[1]Research Scholar, Department of Computer Science, H.P.U. Shimla

[2]Department of Computer Science, H.P.U. Shimla

Email address: [1]balvir.thakur@gmail.com

Abstract**—**In earlier days, the main focus of software developer was on functionality, through ad-hoc processes they developed the software. Due to this the quality of data produced by these software was not good. Software was not considered a concrete value. Everyone agreed on it importance, but it was not considered as a possession. But at present software is a part of the balance of an organization. Data is slowly following the same process. The information owned by an organization in an important part of its assets. Generally, the methods and techniques are applies to software but data itself has often been considered as an external problem. 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 understand what good quality means. Hence, we need to measure data quality in order to find how valuable the information is, and how to improve it. It is impossible to make empirical evaluations if we do not agree on what (and how) should be measured. In this paper we measure information quality using metrics. We apply traditional software metrics techniques to measure data quality. We use the Goal-Question-Metric (GQM) methodology and apply this on the database of university examination system of Himachal Pradesh University as a case study.

Keywords**—** Data/Information quality; goal-question-metrics(GQM); data metrics.

I. Introduction

In earlier days software was not considered a concrete value in an organization. Everyone agreed on the importance of software, but it was not considered as a good, as a possession. In those days, the value of software was associated with its cost. Nowadays, software is assets for an organization. Data is slowly following the same process. In fact, more people are now talking about “the value of information.” New organizations have information as it main asset. Organizations have information that may help them to improve their work, make decisions, and increase their profits. This information is usually stored in large databases, managed via software applications.

Himachal Pradesh University has computerised its examination system to process the result of students. We used the database of this system for analysis. The university examination system's main focus is to process the result of graduate and post graduate students. This system has two phases, namely pre-examination and post examination phase. In pre-examination phase collect the information of students studying in various colleges and university campuses. The post examination phase prepare history sheet of student and process their result effectively.

Now, how could an organization know that it has the right information at the right time? How could an organization evaluate its information, in order to take corrective actions? That is a matter of *data quality* (DQ). How could the quality of the information in an organization be measured, in order to decide if data can be trusted?

Data itself has often been considered as an “external” problem. Quality techniques for validation and verification usually assume that an external agent provides data, and concentrate only on software itself. The problem of bad information quality was not considered a problem of software professionals: it was rather a problem of the users of the system.

Our claim is that software engineering should take into account data quality issues, in order to prevent, detect, and solve problems in the use of systems caused by bad quality of data. Some of these problems are human in nature; others can be addressed using standard software techniques e.g. data standardization, verification, profiling/monitoring, matching, and integration with other tools.

In our work, we are interested in measuring data quality and we used the data of university examination system. Measuring the quality of the information will help us to know its value, and also its pitfalls. We agree with the well-known proverb: "if you can’t measure it, you can’t manage it”. We propose to measure information quality using metrics defined using traditional software engineering techniques. Metrics have been studied in software engineering [5]. In particular, we base our work on the GQM methodology [2].

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 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.

Our main scope is to apply these techniques to data in digital format, data stored in computers systems. This data is usually organized in databases, at least at a logical level. Thus, we concentrate on models that capture databases [14], [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 recent years, researchers have been studying data quality problems from the perspective of the data generation processes [16], [12, [13]. They have identified problems in data, and tried to associate them with problems in the process that lead to this data.

In [9], Redman gives a introduction to DQ issues. He points out the importance of understanding data quality requirements from the beginning, but from a management perspective. Other approaches [12], [13] follow the same management perspective. They use ad-hoc metrics for specific database applications (medical, student, etc.), and most of them rely on user satisfaction as a measure of quality. In this paper, we present a framework for defining and using data quality metrics. The outcome of our work is a suitable set of metrics that establish a starting point for a systematic analysis of data quality. Moreover, we 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, a methodology for metrics definition in the software development process. Section 4 we make use of these quality metrics to analyse the database of university examination system of Himachal Pradesh University. Section 5 describes how to incorporate data quality metrics to evaluate, improve, and maintain levels of quality in the organization. In section 6 we present our conclusions and suggest future work.

II. WHAT IS DATA QUALITY ?

Data quality (DQ) is a term to describe the quality of the content of information systems. It is often pragmatically defined as: "The fitness for use of the information provided." Most information system practitioners use the term synonymously with information Quality. Quality is not an absolute concept. In the case of data quality, we may want to take into account only specific 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 it might give to an organization. As we have said, the quality of data depends on several aspects. We present several attributes that we think have to be measured in order to determine the quality of our data. These attributes or dimensions have been taken from [16, 12] following the point of view of the value of the data, i.e., our pragmatic view of data quality. We present an informal definition for each of the attributes considered.

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 trustable, i.e., it can be taken as true information.

Amount of data: The number of facts stored.
Consistency: There is no contradiction between the data stored.

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

Timeliness: Data is updated in time; update frequency is adequate.

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

Unambiguous: Each piece of data has a unique meaning.
Accuracy: Each piece of data stored is related to a real world datum in a precise way.

Objectivity: Data is objective, i.e., it does not depend on the judgment, interpretation, or evaluation of people.

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

Usefulness: The stored information is applicable for the organization.

Usability: The stored information is usable by the organization.

III. ELEMENTS OF THE GQM APPROACH

GQM [2] is a framework for the definition of metrics. GQM is based on the assumption that in order to measure in a useful way, an organization must:

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

We have chosen this framework because it is a top down approach that provide guidelines to define metrics, without a priori knowledge of the specific measures. Following GQM, we first are able to state which dimensions characterize our notion of data quality. Then, we can ask questions characterizing each dimension, without giving a precise (formal) definition -that is sometimes impossible-, only focusing on their relevant characteristics from our point of view. Finally, we give metrics (some objective, some others based on people appreciation) to answer these questions, giving us a more precise valuation of the quality of our data.
A *goal* in GQM is defined in a precise way. A goal is defined for an *object*, with a *purpose*, from a *perspective*, in an *environment*. For example (in a software organization): “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 point of view, and the environment is the composition of the maintenance staff. A goal in GQM is posed at the conceptual level. 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. For example (in the context of the goal stated above): “What is the current change processing time?” A question in GQM is posed at the operational level.

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. For example: “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 (for example, the answer for the question "how many times do you consult the data per day?").

IV. Data Quality Metrics

In this section, 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. As we had said, 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;
- a fixed data model;
- 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 want to use data for.
- a fixed set of "temporal" attributes: A subset of the database attributes may have deadlines, i.e., they may have to be updated in order to be accurate. It is called set of temporal attributes. In order 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.

We have grouped the dimensions with respect to the object they are related to. As a consequence, we have two groups of dimensions: set of data dimensions, and data model dimensions. This classification may be useful when trying to correct the quality problems found. Next, we present a few of data quality metrics we defined using the GQM framework. To do this, we have chosen a subset of the dimensions presented in Section 2. First, we introduce several goals that we have identified for a particular dimension. Then, we include questions and metrics for each goal. Last, we give an operational technique to compute the associated values.

In this work, we want to present a framework in order to define data quality metrics and analyze them. New dimensions can be used, new questions can be asked, new metrics can be given, and different techniques can be used. In Table 1 and Table 2 we show the application of GQM to derive some data quality metrics for the set of data object and the data model object, respectively.

Instance Dimension Metrics

Reliability (The data stored is trustable; it can be taken as true information.) We propose to measure the reliability of data as the relation between the number of obtained answers that conform the expected answers and the total number of obtained answers. Therefore, in order to measure reliability, we need to characterize the expected answers for fixed query sets, as we do in functional test with expected values for inputs. Several techniques may be developed to characterize expected answers, for example using predicates, or estimating number of elements returned for each query, or defining the answers in terms of the instance. We call this approach Functional Data Test [3], for its analogy to Functional Testing in traditional software engineering.

Relevance (Every piece of information stored is important in order to get a real world representation.) We want to know if we are storing useless data. We propose to measure the percentage of elements from the total number of elements that are never returned as answers. It is easy to count the number of tuples that are not part of an answer, for a certain query. We know the number of tuples in the tables involved, and we can count the number of tuples returned. One point here is: what does "never" mean? How many times we need to query the database to have an accurate estimate? These questions can only be answered empirically, after conducting adequate experimentation on real systems. The technique "Query sets" refers to this notion of running queries and performing a calculation based on the results obtained.

Usefulness (The stored information is applicable for the organization.) We measure usefulness by the rate in which data is actually consulted. So we need to measure how often the organization uses the data stored, and the benefits obtained from that use. When it is possible to identify a subset of critical data, we can measure the times that data is accessed with respect to the total data accesses. We can also measure the percentage of accesses to query data over all accesses (including updates). If data is only inserted, deleted, or modified, but never consulted, we may infer that there is a problem. We expect that data is used more than modified.

Timeliness (Data is updated in time; update frequency).We are interested in measuring how accurate is our data with respect to time (deadlines). If we know which attributes are time dependent, and how to know if specific values are outdated, we may define a test over the data to estimate the number of records that are outdated. We call this technique Temporal Test.

Table 1. Derivation of data quality metrics for the set of data object.

Goal	Question	Metric	Technique
Object: Set of data **Purpose:** Evaluate **Quality:** Reliability **Perspective**: User **Environment:** -fixed data set -fixed query set	Do the obtained answers conform the expected answers?	Number of answers that conform the expected answers / Total number of answers	"Functional Data Test" DCF
Object: Set of data **Purpose:** Evaluate **Quality:** Relevance **Perspective:** User **Environment:** -fixed data set -fixed query set	Is there data never queried?	% of tuples never returned as answers	Query set
Object: Set of data **Purpose:** Evaluate **Quality:** Usefulness **Perspective:** User **Environment:** -fixed data set	How many times is the stored information queried every day?	Number of accesses to query the data (not including modifications)	LOG of database activities
	Is the data stored used at decision time?	% of decisions made using stored data	DCF
	Is there any difference in having or not having the data at decision time (i.e., does data help to make "profitable" decisions)?	Rupees earned in decisions made using the data stored (per time unit) - Rupees earned in decisions made without using data stored (per time unit)	DCF
Object: Set of data **Purpose:** Evaluate **Quality:** Timeliness **Perspective:** User **Environment:** -fixed data set -fixed query set -T: Set of temporal attributes	How often is data updated?	Number of update operations per unit of time	LOG of database activities
	Which percentage of data is updated?	Number of records with attributes in T updated (per time unit) / Number of records in the database	LOG of database activities
	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 attribute in T	Temporal Testing

Table 2. Derivation of data quality metrics for the data model object.

Goal	Question	Metrics	Technique
Object: Data model **Purpose:** Evaluate **Quality:** Conciseness **Perspective:** User **Environment:** -fixed data model -fixed query set	Are there attributes (e.g., tuples, columns) that are never accessed?	Number of attributes never accessed	Query set
	Does dependency exist between certain attributes (i.e., may one attribute be computed in terms of others)?	Number of dependent attributes / total number of attributes	Query set
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 (does not fit in the model)	DCF
	Is every field format the right one to store the expected data?	Number of times a value for an attribute could not be stored in the correspondent field	DCF

V. Measuring and Analyzing Data Quality

Once we have defined our data quality metrics (i.e., what and how to measure) we want to use them. We can simply take our relational database (University Examination Database), identify the dimensions we are interested in, choose the appropriate metrics and techniques depending on specific considerations, apply them, and analyze the results. This is a useful approach, especially when the system is already in production, the database is implemented. There is a lot of data loaded, and we want to have a picture of the current situation in order to decide what to improve. Next we present the steps we need to follow to evaluate the quality of our data:

1. Choose the interesting dimensions: In fact, not all the dimensions are relevant in every situation.

2. Choose or define the questions that characterize the dimensions: Each dimension has several aspects that characterize it. Not every aspect is important in every situation.

3. Choose or define the metrics and techniques to answer each question: Depending on the system implementation, the resources, deadlines, etc., one technique or another should be selected to answer the same question.

4. For each metric, define values or ranges representing good and bad quality data: This is a crucial point. Not in all the cases a clear notion of good quality and bad quality can be identified. Probably, at the beginning these notions will be based on intuition. A measurable notion of data quality has to be constructed by repeatedly measuring, storing, and comparing results and organization revenues. See point 9.

5. If subjective metrics have been chosen, define appropriate DCF and data collection procedures: In fact, in the case of subjective metrics the questions to be answered are probably different for each system.

6. Apply the techniques to the correspondent objects (data model, database instance, documentation, etc.), when possible.

7. Collect the information using the DCF.

8. For each metric, determine if data quality is acceptable or not, and take the appropriate corrective actions: If the obtained values do not agree with the expected ones (see point 4), there are several possibilities. For example, if the value is “not so bad” we can decide to do nothing. If it is bad, but the cost of correcting it is higher than the expected benefits, do nothing. If the dimension affected is crucial, or we decide to correct the data anyway, we may decide to clean the data and to improve the data generation process.

9. Store the obtained results: At the beginning, when we start measuring the quality of the data, we may have no idea of the desirable standards of quality we want. However, we have to start our measurement procedures. So it is important to define acceptable ranges, when possible, and to start the “data quality history” in the organization, in all cases. We have to define the appropriate container to store the results, and to store the “state of the organization” too. In order to extract useful conclusions, we need to know how data affects the organization.

10. Go to step 1: We will need to measure, and to correct every time we see that something is wrong, and to check if everything is fine. It is possible that the acceptable ranges or the dimensions of interest change from time to time. So, it is important to review them.

This “algorithm” is slightly naive. We believe that its simplicity shows the capability of the technique used to the problem. The procedure presented above is not a data quality plan. It only deals with measuring the quality of data at certain points that can help in deciding which corrective or preventive actions to implement. In order to reach maintain high levels of data quality, it has to be part of a broader plan, that takes into account all the aspects of data quality in the organization (see [9]).

Another approach is to see the dimensions we are interested in as data quality requirements. These requirements can be assessed from the beginning of the software development process, in the same way that we have functional and non-functional requirements. So, we want to deal with them from the beginning, and incorporate them to our specification, our design, and our system implementation.

VI. Conclusion

In this work, we introduced some aspects of data quality from a software engineering perspective. We pointed out the relevance of the quality of data as it is already recognized in the information systems community, and how to deal with it. In particular, we emphasized the importance of measuring the quality of the data. We proposed to use well-known software engineering methods for metrics definition and to apply them to define data quality metrics. We identified the attributes we wanted to measure, and obtained a set of metrics and techniques to calculate them.

First of all, we need to investigate the importance of dealing with the quality of data in an independent way. On the other hand, it is necessary to study the appropriateness of the metrics here proposed. To do this, we are planning to apply these metrics to several case studies.

The dimensions we measure to determine the quality of our data may be consider as data quality requirements. We believe that these requirements should be assessed from the beginning of the software development process. In fact, they may be considered as particular non-functional requirements. So, we need a way to define and to verify them at every stage of this process [3].

The metrics presented above allow us evaluate the quality of the data at a current time. In a dynamic situation, data enter the database, is modified, deleted, etc. It may be interesting to predict how these changes may affect the current quality of the data with respect to certain dimensions. In future we also want to apply data mining techniques to improve the quality of database of University Examination System

References

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

[2] V. R. Basili and H. D. Rombach, “The TAME project: towards improvement-oriented software environments,” *IEEE Transactions on Software Engineering*, vol. 14, no. 6, June 1988.

[3] M. Bobrowski, M. Marré, and D. Yankelevich, “A software engineering view of data quality,” Submitted to the European Quality Week, 1998.

[4] W. Boehm, J. R. Brown, and M. Lipow, “Quantitative evaluation of software quality,” *Proceedings of the Second International Conference on Software Engineering*, 1976.

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

[6] W. Haebich, "A quantitative model to support data quality improvement," *Proceedings of the Conference on Information Quality*, MIT, Boston, October 1997.

[7] M. Jarke and Y. Vassiliou, "Data warehouse quality: a review of the DWQ project," *Proceedings of the Conference on Information Quality*, MIT, Boston, October 1997.

[8] J. A. McCall, P. K. Richards, and G. F. Walters, *Factors in software quality*, Rome Air Development Center, RADC TR-77-369, 1977.

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

[10] T. Redman, *The Impact of Poor Data Quality on the Typical Enterprise*, *Communications of the ACM*, vol. 41, no. 2, pp. 79-82, February 1998.

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

[12] D. Strong, Y. Lee, and R. Wang, *Data Quality in Context*, *Communications of the ACM*, vol. 40, no. 5, May, 1997.

[13] D. Strong, Y. Lee, and R. Wang, "10 Potholes in the road of information quality," *IEEE Computer*, August 1997.

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

[15] F. Van Latum, R. Van Solingen, M. Oivo, B. Hoisi, D. Rombach, and G. Ruhe, "Adopting GQM-Based measurement in an industrial environment," *IEEE Software*, pp. 78-86, 1998.

[16] Y. Wand and R. Wang, *Anchoring Data Quality Dimensions in Ontological Foundations*, *Communications of the ACM*, vol. 39, no. 11, 1996.