# Analyzing and Improving Data Quality *

**Agustina Buccella and Alejandra Cechich**
GIISCO Research Group
Departamento de Ciencias de la Computación
Universidad Nacional del Comahue
Neuquen, Argentina
{abuccel,acechich}@uncoma.edu.ar
**and**
**Gonzalo Domingo**
Proyectos de Telesupervisión y Geociencias
D.S.I. Cuenta E&P - Argentina Sur
Repsol YPF
gonzalodomingo2@hotmail.com

## Abstract

*Data quality is a research area strongly investigated during the 90's. However, few companies in Argentina apply data quality methodologies or tools during the analysis, design or implementation phases of software development process. Developers generally use techniques to design systems such as UML without considering mechanisms for future data quality problems. In this work we propose a methodology in which the data quality is an essential part of the whole software development process. Early design decisions on data quality strongly impact on the system. Our methodology defines a set of practices to be applied on the software life cycle. In addition these practices act as a means to evaluate if systems already running fulfill with minimal data quality requirements.*

## 1 Introduction

Many definitions for the term *Data Quality* have been proposed in the literature, each considering data from different perspectives. However all definitions converge on only one theory: *data quality is strongly related to the use of data* [3, 7, 9]. In this way, depending on the use of data, quality can be considered enough for some uses and not enough for others. Rules as *garbage in, garbage out*, *if inaccurate information is entered, only inaccurate information will result*, *pay now or pay more later* are still true within the data quality issues. In our work, we consider the definition of model quality of FUNDIBQ (Iberoamerican Foundation of Quality) in which data quality is defined as a set of essential characteristics of a product, service, system, or process to meet needs and expectations of interested parties. In conclusion data quality represents a point of agreement among parties.

An organization containing low data quality generates dissatisfied clients when their invoices or requests contain errors in their personal data, dissatisfied employees when they make errors because of both incorrect and uncertain information, dissatisfied managers when they have to take decisions based on these data, etc. Therefore, considering data quality as an important characteristic within an organization will avoid this type of problems and will take new benefits. For instance, improvements to support the decision making process, decrease in time to obtain information, replacement of expensive activities for cheaper ones, and a better concept of the organization's image are only some of the benefits that a good data quality provides.

Several approaches for data quality have been proposed in the literature [1, 4, 5, 6, 8, 10]. In particular, works related to our proposal can be found in [5, 6, 10]. The proposal of Wang et. al [8, 10] defines a methodology, named Total Data Quality Management (TDQM), aimed at generating high quality information products for consumers of this information. The methodology analyzes and conceptualizes information products in order to build information manufacturing systems (IMS). Thus, these IMS contain the set of functionalities of a system together with the quality controls that must be implemented. In this way, the methodology can identify possible data quality

* This work is partially supported by the UNComa project 04/E072 (Identificación, Evaluación y Uso de Composiciones Software).

problems by analyzing the way data are produced.

In the proposal described in [5], the data quality concept is based on the use of data. The author assumes that "the only way to truly improve data quality is to increase the use of that data". In this way, his work defines six rules for data quality including unused data do not remain correct for a long time, data quality is defined as how the data is used instead of how they are obtained, data quality will not be better than their more rigorous use, etc. Based on these rules, four activities are defined in order to evaluate and analyze data quality: *auditory*, *redesign*, *training*, and *measure*. The first rule consists of determining how good are data today. Redesign activity refers to evaluate critical data managed by current applications analyzing carefully the use given for each of them. Training activity aims to prepare users for understanding how important data quality is. Thus, training and educational tasks are carried out. Finally, measure activity refers to constantly judge data quality, that is, all previous activities must be repeated generating an iterative process. In comparison with our work, we propose a methodology defined as a practical guide that must be applied to every system during early phases in the software development process. A set of practices or recommendations are proposed as rules to be evaluated. These rules are not based on the use of data but on the way they are influenced over a set of dimensions of data quality. However, several of these rules or practices included in our approach are based on rules defined in [5].

Finally, in [6] data quality is measured through multiple quality dimensions such as, accessibility, completeness, reliability, consistency, etc. Control matrices are used to combine problems on data quality with quality controls, and thus allowing developers to evaluate information products. Columns of the matrix enumerate the problems in data quality that affect information products; and rows represent quality controls applied on the manufacturing information process to prevent, detect, or correct data quality problems. In this way, these controls tend to avoid errors within the information product. As in this proposal, our work is based on the fact that data quality must be an activity considered during the data life cycle. In addition, we implement the evaluation of applications by using a matrix. Then, we classify different quality practices, each one as part of one phase of the life cycle of data, analyzing how preestablished parameters are met.

This paper is organized as follows: the next section briefly introduces data quality concepts used in our approach. Section 3 presents our methodology describing a set of practices within the data life cycle. Section 4 presents a real case study applying the methodology on a real application. Future work and conclusions are discussed afterwards.

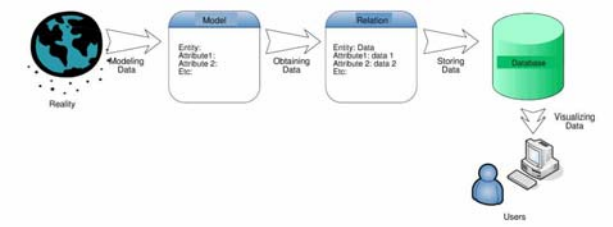


**Figure 1. The data life cycle**

## 2 Data Quality

In [7] the data life cycle is composed of four main phases: *data modeling*, *obtaining values*, *storing data*, and *visualizing data*. Figure 1 shows these phases graphically.

The first phase refers to the process of modeling a universe of discourse. The resultant abstraction represents the reality described by user requirements. This abstraction defines a logic model representing data and the flow of information between different roles of users and applications. In the obtaining values phase, data is obtained from reality through user interfaces of the system or through interfaces with other systems. Here, aspects such as masks, dates, validations, and/or rules implemented on interfaces must be taken into account. In the storing data phase, data obtained in the last phase are stored on repositories. Finally, the visualizing data phase refers to the way data are presented to users. Aspects such as inconsistencies, robustness, or identification of errors, must be considered.

In order to analyze and evaluate data quality within a software development process we consider four main dimensions [2]:

- *Accuracy*: Do data represent exactly the reality or verified sources?. It is related to the source, that is, the level of correspondence among data and the real word.

- *Completeness*: Are all needed data available? Which data are absent?. It refers to data that must be available within an information system.

- *Consistency*: Were data consistently defined and understood? It refers to the definition of standards and protocols for data. All data must be presented in a compatible format defined as the best format for the task in development.

- *Timeliness*: Are data available when required? Within this dimension the concept of volatility is defined denoting the time data remain valid.

These four dimensions are used in our work to evaluate the data quality of an organization together with practices applied on each phase of the data life cycle. Next section describes each of these practices and how they are influenced by the dimensions.

## 3 A New Approach for Improving Data Quality

In order to propose a new methodology, which consists of a set of practices, every activity within a software development process has been analyzed with respect to data quality. The different phases of data life cycle are crucial to classify and evaluate each practice in accordance with the four dimensions described in the last section.

As a result of our analysis, we define 64 practices where 27 are in the data modeling phase, 22 in the obtaining values phase, 4 in the storing data phase and 11 in the visualizing data phase. For brevity we only describe some of these practices in this paper.

### Data Life Cycle: Data Modeling

- *When a user account is deactivated, the user in charge of the flow must be notified in order to take preventive actions*: In this way, actions deriving from this change maintain the business flow updated. This practice affects timeliness because the data will remain valid during time. If the data change, users in charge of these data will be notified allowing them to early detect differences with respect to the reality. Accuracy is also affected, as this practice detects in an early phase variances between stored data and reality, minimizing the negative impact of redundant data.

- *Data must be retrieved from a data source*: Applications capturing data closer to their generation are considered as data sources. Then, an interface between a data source and this application must be implemented when the data are required by another application. In this way, the uniqueness of the process is guaranteed, improving the use and the implementation of new systems. This practice generates data that remain valid for more time supporting the timeliness dimension. In addition, consistency is also supported because changes can be early detected. The definition of data modeling does not changed when these types of systems are related to each other. This practice also benefits the completeness dimension because all elements of each data can be obtained from another system. In regards to accuracy, the better use and implementation of data improves the relation between the data and the reality. In addition, the practice tries to avoid redundant data by reducing the probability of error.

- *Avoiding redundant data among systems*: This practice guarantees not only uniqueness but also improves the use and implementation of systems. Data remain valid for more time and changes can be early detected. Thus, timeliness is supported. With respect to accuracy, this practice intents to improve the relation between data and reality avoiding same information through different systems. Again, as in the last practice, by avoiding redundant data, the probability of error is reduced.

### Data Life Cycle: Obtaining Values

- *Before executing an insertion, use "like" queries*: This practice refers to verify if a datum exists in a database before inserting it. For instance, if we want to insert a street with the name of "Rivadavia" we must first check coincidences between this name and the street name stored in the database. Then, if there are two instances with the same name, the user must choose which of them is the right one in each case. In this way, consistency is fully supported because typical problems of data formats can be avoided. The probability of errors is reduced by ruling out the possibility of typing errors.

- *Evaluating the input of data more than once when they are critical*: Critical data must be previously identified to avoid data insertion becoming tedious. Accuracy is supported because typing errors within critical data are minimized.

### Data Life Cycle: Storing Data

- *An inferred data must never be entered manually*: To avoid typing errors during data inputs applications must implement this practice. The four dimensions are supported. Timeliness because inferred data from these data also change when data sources change; consistency because the data format will be defined within the application; completeness

because these data are not entered manually and the practice must check that data sources are complete; and accuracy because the inferred rules must be modeled correctly guaranteeing also the accuracy of data in which they are based on.

- *Business rules must be part of the applications so generated data can be stored and filtered by these rules*: In this way, the rule can reject the input whether input data are not what the application is expecting. Timeliness is supported because the rule checks for unupdated inputs; consistency is supported because rules also consider the format of data; and accuracy is supported because typing errors are intended to be reduced.

**Data Life Cycle: Visualizing data**

- *The system must alert about expiration dates*: The system must inform the person who is in charge of the data, that some data are loosing validity in accordance with the logic of the application. Timeliness is supported because preventive actions can be implemented; and accuracy because the reality is again verified when data become inexact.

- *The system must verify and inform about changes on data tendency*: In this way, a change on data tendency can be identified. A functional analysis will be required by the owner of the data to determine whether the change was in fact a variation on the tendency of reality or just a mere error. Accuracy is improved by this practice because it is possible to early detect and localize errors in the data registry.

- *The supported business process must be opened to other processes (sharing data)*: This practice is more related to business than to systems. However, it is important to identify when a process must interact with other processes. In this way, the use of data is intensified.

The set of practices that we defined in our approach are classified by two criteria depending on the independence of the person performing the analysis, *Objective* and *Subjective*. The first one is performed by a quality committee in charge of evaluating the practices. The second one depends on the person performing the evaluation; it can vary in accordance with the knowledge of the domain and his/her previous experience. A score table determines an error value when a practice is not fulfilled with respect to each criteria.

The Objective Classification is defined as follows:

- *Standard practice (E)*: It is considered as a standard in the industry and practices in this classification must be implemented.

- *Good practice (B)*: Practices here must be implemented when possible. They are very important for guaranteeing data quality over each dimension they affect. However, they are not mandatory as in the first classification as sometimes they are difficult to apply. When practices are not implemented the reasons must be documented.

- *Recommendation (R)*: These practices are considered favorable. The project leader is responsible for evaluating the cost/benefit of the application for each practice.

The Subjective Classification contains the following values:

- *No error*: Practices are implemented and correctly applied on the application.

- *No effect*: The recommended practice is not implemented. However it is not an error; this decision is taken during design time and it must be documented.

- *Slight*: Situations in which data quality is slightly damaged. For instance, when the application allows entering low data quality without affecting either critical data or the success of the task that the user is performing.

- *Grave*: Situations in which low data quality can affect the success of the task. For instance an incorrect data generating an erroneous report.

- *Fatal*: Conceptual errors, incorrect application of a model, or errors hindering the finalization of a task successfully. They are dangerous errors because users never know when and why they happen.

### 3.1 Improving the Software Development Process

Our work was developed in a company in Neuquén, Argentina, named it "El Petróleo SA" (for confidentiality reasons). The company has established a predefined methodology for the software development process that must be followed by every development. However, data quality was not considered as a task during its early phases.

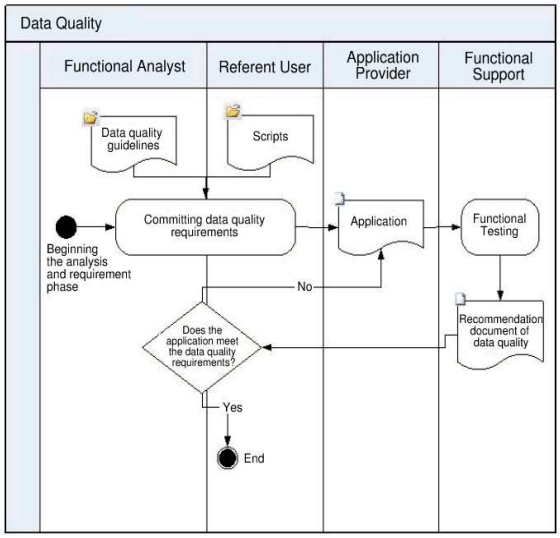


**Figure 2. Part of the the new software development process**

During our exhaustive analysis about how to improve data quality within the company, a set of set of practices and recommendations were defined. These practices changed the predefined methodology including quality controls applied during the whole process. Figure 2 shows part of the new software development process with these controls.

The Figure shows how different stakeholders interact with the development of an application. The stakeholders are: *Referent User*, *Functional Analyst*, *Application Provider*, and *Functional Support*.

During the analysis and requirements specification phase, the referent user delivers scripts describing functional requirement of the system to be built. At the same time (but in a separate process) the functional analyst determines quality guidelines. In order to define which practices will be applied, the functional analyst, together with the referent user, evaluate weights and cost/benefit of each of the practices. As a result, a document containing these quality guidelines is created and delivered to the application provider.

Then, the project team analyzes if there is a corporative solution to apply to the above needs (this step is not shown in the figure due to lack of space). If a solution exists, the project manager must agree with it. However, if a predefined solution is not available, a new document has to be generated enumerating the needs of the business, the goals of the solution and their scope.

The requirement specification and design are the first phases to be performed when the development is local. The scripts are delivered to the programming team in order to estimate the time that must be committed to each script. Then, the project team creates the schedule and the plan to be delivered. New interviews with the referent user define final priorities and delivered times. Finally, a document is written and signed by each party, i.e. the referent user and the software manager. This document is then received by the programming team together with quality guidelines. The document contains the 64 practices except for the ones that have been deleted by a design decision.

**Table 1. The score table for the objective classification**

| Score | |
|---|---|
| Standard practice | 3 |
| Good practice | 2 |
| Recommendation | 1 |
| No action | 0 |

After that, unit and integration tests are performed by the programming team. In conformity with the schedule, the source code of the application and the installation manual is delivered. To perform functional tests, the project team installs the application in a test server. In addition, the functional support uses these functional tests as guides to perform data quality tests. To do so, he/she applies the two criteria described before to obtain error values, and the subsequent recommendation for accepting or rejecting the tests. Finally, the functional support creates, based on these results, a document with recommendations in regards to data quality. In this document each practice, together with each dimension, are evaluated describing whether they are being applied or not. An explanation must be provided where a practice is not applied.

The functional analyst and the referent user have to decide then whether the recommendation document of data quality satisfies a minimal set of acceptable quality guidelines. The application finishes its development when the referent user approves the document. In the case of the document being rejected, the application returns to the programming team in order to perform all the required changes.

### 3.2 Verifying Applications

In our approach each practice is scored with respect to the objective classification (Table 1).

During the verification phase, the functional support evaluates each practice and each dimension by using the score table for the subjective classification (Table 2). After that, the error value of each practice (Table 1) times the error value for each dimension (Table 2) give us the final value for each practice. Finally, all these final error values

**Table 2. The score table for the subjective classification**

| Score | |
|---|---|
| No error | 0 |
| No effect | 0 |
| Slight | 5 |
| Grave | 15 |
| Fatal | 40 |

are added to determine the final result of the verification. If the final result is lower than 46 the application passes the test; if it is between 46 and 119 the application is accepted with observations; and if it is higher than 120 the application does not pass the test.

These thresholds have been chosen to generate a rejection (application does not pass the test) in the case where a standard practice contains a fatal error (equivalent to 40 x 3). Accordingly, any error value greater than this will be rejected. A pass will be granted with any error value lower than 46. Therefore in order to pass the test only one grave error in a standard practice will be accepted. Any combination of intermediate values will pass the test with observations.

The results of the test will be validated with the referent user. He/she can accept the application in spite of a negative result of the test. However, this must be a decision made by consensus between the referent user and the project manager. They can assume that some data quality problems will be fixed during next phases. The decision of approving or rejecting the application, the reason and even the test with the results must be documented in the recommendation document of data quality previously described.

## 4 Case Study

Our approach has been applied to an application managing the electricity of our company "El Petróleo SA". This application contains information about the electricity the company generates, buys and sells. In addition, it stores information about the electricity generators. One of the goals of this application is to generate reports for the Argentinean Secretary of Energy.

Before beginning the development of this application, a training activity was performed to explain the new software development process. In this activity the programming team, the functional analyst, and functional support learned about their role in this new process in which data quality is strongly involved. Thus, before programming the code of the application, the team received the 64 practices together with the classification of the four dimensions for each data life cycle.

Once the programming team finished the unit and integration tests, the functional support performed the functional, usability and data quality tests defined in our new development process. To do so, he/she analyzed each practice verifying errors. The final task was the writing and delivery of the recommendation document of data quality. Some observations written in this document are described below.

### Data Life Cycle: Data Modeling

- *When a user account is deactivated, the user in charge of the flow must be notified in order to take preventive actions*: The timeliness dimension contains a "slight" value due to the lack of controls related to this dimension. However, as the number of users is low in this phase a harder assessment was not necessary. For the same reason, the accuracy dimension contains a "no effect" value. An observation explaining the reasons was documented.
- *Data must be retrieved from a data source*: In the application there are data extracted from source data of another application. For instance, the market contracts in SAP. The application defines an interface to extract these data. Therefore, the practice contains a "no error" value.
- *Avoiding redundant data among systems*: The timeliness dimension contains a "slight" value and the accuracy contains a "no effect" value. As the data source extracted from SAP cannot be accessed on-line, once a day data are exported to the application. Thus, the timeliness is affected because the application can contain invalid data for at most 24 hours. However, the accuracy is not affected because imported data are slightly dynamics.

### Data Life Cycle: Obtaining Values

- *Before executing an insertion, use "like" queries*: The accuracy dimension contains a "slight" value and the consistency and completeness a "no effect" value because in this case redundant data are verified by the database.
- *Evaluating the input of data more than once when they are critical*: The accuracy dimension contains a "no effect" value because there are implemented mechanisms to manage critical data. For instance, a mail is triggered by the application when an invoice is entered. The mail is received by the people in charge of verifying and approving the payment.

### Data Life Cycle: Storing Data

- *An inferred data must never be entered manually*: This practice contains a "no error" value.
- *Business rules must be part of the applications so generated data can be stored and filtered by these rules*: This practice contains a "no error" value.

### Data Life Cycle: Visualizing data

- *The system must alert about expiration dates*: This practice contains a "no error" value. For instance, a defined rule is that once an invoice is entered into the application there are only 5 days to pay it.
- *The system must verified and inform about changes on data tendency*: The accuracy dimension contains a "slight" value because this practice is not implemented yet. However, it will be implemented during next phases in the development process.
- *The supported business process must be opened to other processes (sharing data)*: This practice contains a "no error" value.

The final result of the error values gave a score of 90 points composed of: 15 points for a standard practice with slight error, 10 points for a good practice with slight error, 10 points for a good practice with slight error, 5 points for a recommendation with slight error, 15 points for a standard practice with slight error, 10 points for a good practice with slight error, 15 points for a standard practice with an slight error, 10 points for a good practice with slight error. With this final result of 90 points, the data quality test generates an assessment of "Approve with observations".

Taking into account the above result, the application continues with its development process. Next phases should implement at least some of the observations described in the recommendation document of data quality.

## 5 Conclusions and Future Work

In our work we have assumed that the definition of data quality depends on an agreement among parties. That is, the set of features a final product must meet to satisfy the expectations of the parties. However, a poor data quality assessment is likely to be obtained when there is a lack of mechanisms to prevent and improve data quality.

In order to face the data quality problem we have proposed changes within the software development process including data quality as an activity in early phases. In addition, these activities are performed through the implementation of a set of practices to apply on both system to be developed and system in production. We have defined also a data quality test in which each practice and each dimension is scored. Accordingly, applications can be evaluated with respect to data quality on each phase of the data life cycle.

Our approach should be an integral component of an organization, of the development team, and of the mentality of their components. In order to achieve this, processes must be improved generating a change within the culture of the participants. Business-persons have to understand why time and money must be spent to check data quality. Data quality must not be something additional to the system, it must form part of the beginning of the software development process.

As future work, we are extending our approach through all systems of the company. As a result, we will be able to perform more experimental validation and to propose new practices according to the complexity and the type of application.

## References

[1] D. Ballou and H. Pazer. Modeling data and process quality in multi-input, multi-output information systems. *Management Science*, 31(2):150–162, 1985.

[2] G. Brackstone. Managing data quality in a statistical agency. *Survey Methodology*, (25):139–179, 1999.

[3] E. M. Burns, O. MacDonald, and A. Champaneri. Data quality assesment methodology: A framework. In *Joint Statistical Meetings - Section on Government Statistics*, pages 334–337, 2000.

[4] L.Pipino, Y. W. Lee, and R. Y. Wang. Data quality assessment. *Communications of the ACM*, 45(4):211–218, 2002.

[5] K. Orr. Data quality and systems theory. *Communications of the ACM*, 41(2):66–71, February 1998.

[6] E. Pierce. Assesing data quality with control matrices. *Communications of the ACM*, 47(2):82–86, February 2004.

[7] T. Redman. *Data Quality: The Field Guide*. Digital Press, January 15 2001.

[8] G. Shankaranarayanan, R. Y. Wang, and M. Ziad. Ip-map: Representing the manufacture of an information product. *MIT Conference on Information Quality*, 2000.

[9] G. Tayi and D. Ballou. Examining data quality. *Communications of the ACM*, 41(2):54–57, February 1998.

[10] R. Y. Wang. A product perspective on total data quality managment. *Communications of the ACM*, 41(2):58–65, February 1998.