HOW TO GIVE FEEDBACK ON DATA QUALITY: A STUDY IN THE FOOD SCIENCES

Complete Research

Weber, David, Department of Computer Science, ETH Zurich, Switzerland, weber@inf.ethz.ch

Presser, Karl, Department of Computer Science, ETH Zurich, Switzerland, presser@inf.ethz.ch

Norrie, Moira C., Department of Computer Science, ETH Zurich, Switzerland, norrie@inf.ethz.ch

Abstract

Data quality is a critical factor in scientific information systems, especially taking into account the fact that the methods used to capture data are constantly being revised and improved which means that data collected over time may have variable quality. We present an approach that we implemented for giving feedback on data quality and report on a study of its use in the food sciences. We distinguish between two main types of data quality feedback, one concerning the validation of data at input time and the other with analysing the quality of data already stored in a database. We propose a general data quality framework and analysis toolkit that allows users to configure both the data quality metrics and how these metrics are visualised. We describe how the toolkit was integrated into a system for the management of food composition data before presenting the results of a questionnaire-based study used to evaluate both the data quality framework and how feedback on data quality is presented to the users.

Keywords: Data Quality, Feedback, Information Systems, Food Science Data

1 Introduction

Despite the absence of a clear definition, it is widely agreed that data quality (DQ) is of major importance to information systems. It is especially important in the management and processing of scientific data since the methods used to capture data are constantly being revised and improved which means that the quality of data may vary over time. In previous work (Presser et al., 2014), we introduced a DQ requirements framework and have shown the importance of the different DQ dimensions for various user groups in the food science domain. Here, we address the issue of how to give feedback about DQ to users of an application.

Most information systems do not allow specific measurements for DQ requirements to be defined, but rather rely only on a set of constraints which validate to either true or false. Therefore, either a user enters valid data and is able to store it in the database or the data is rejected. In many cases, this is not sufficient, as it might be appropriate to distinguish several levels of DQ or even use a continuous range from very poor DQ to very good DQ. This requires another concept for validation as well as metrics for DQ. Further, one needs to examine how and when users should be given feedback about DQ based on such metrics.

We introduce two types of feedback to distinguish between direct feedback on individual entities given during data input and feedback resulting from an analysis of the database. We refer to the former as *Data Quality Input Feedback (DQIF)* and the latter as *Data Quality Analysis Feedback (DQAF)*. To evaluate our approach, we first implemented a DQ analysis toolkit for measuring and visualising DQ and integrated it into a system for managing food science data. The toolkit offers a variety of charts and tables that can be used to provide users with feedback on DQ and help them identify areas where action might be required. It also provides tools to drill down on the DQ issues and identify individual problems. We then carried out

a survey involving several food science data experts across Europe to evaluate our proposed means of providing feedback on DQ.
In this paper, we present the main features of our DQ analysis toolkit along with the results of our survey. We focus on the most interesting insights and conclusions concerning how DQ requirements should be defined, who should define them and how feedback should be given to users.
Sect. 2 gives an overview of previous work on DQ feedback as well as background on DQ related to food composition databases. Sect. 3 then explains how our DQ feedback enables users to get an overview of the quality of the data and how specific problems in the data can be identified. Details of the implementation are give in Sect. 4 before presenting the evaluation of the system based on a questionnaire in Sect. 5. A discussion of the outcomes and implications for future work are given in Sect. 6 along with concluding remarks.

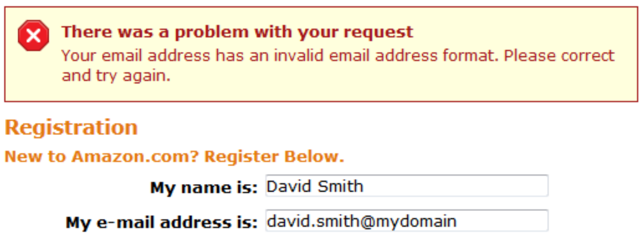
2 Background

Data quality (DQ) is a research area still gaining in importance. The term DQ, or information quality which is often used as a synonym, is seen as one of the keys for business performance. Yet there is neither a precise nor a unique definition of DQ. Nevertheless, it is clear that data considered to be of poor quality in some sense can cause several types of problems in an information system (e.g. Wang, Reddy, et al. (1995), Wand and Wang (1996) and Madnick et al. (2009)). Efforts have been made to define DQ in terms of a wide variety of DQ dimensions (Batini and Scannapieco, 2006), each of which captures a specific aspect of DQ such as accuracy or currency and contributes to a measure of the overall DQ. Wang and Strong (1996) identified 118‘DQ attributes’ and recognised that, in the end, it is always the data consumer who has to decide whether the data is fit for a particular use.
How can good quality of data in an information system be ensured? Most obviously, we think first of restricting the input process so that data of poor quality cannot be entered into the system. This can be done at the UI level, within application programs or at the database level using constraints. Alongside data input validation, there exists the possibility of analysing the quality of the data already stored in the database. Users (or power users) can then be informed about any problems and take actions to deal with them. Such an analysis is related to DQ assurance or data cleaning which is concerned with the correction and improvement of data in databases as described in Ganti and Sarma (2013).
For both types of validation, referred to as DQIF and DQAF, respectively, it is necessary to give feedback to the user to enable the user to correct the data. We will look at the requirements of each of these in turn before going on to discuss what has been done so far on DQ in the domain of food sciences.

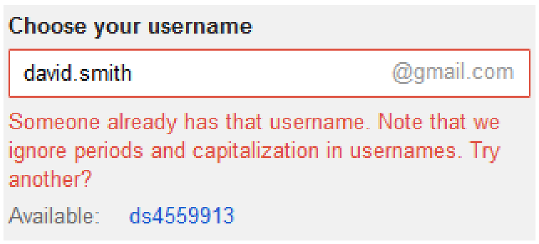
DQIF There exist many possibilities for UI data input validation feedback. We give an overview of the most common techniques and distinguish three aspects of the feedback: (1) where, (2) how and (3) when.

1. In a UI, usually there exists an input area and many input components such as form fields. Feedback can be given close to the input data (e.g. above or to the right of a field or as a tool tip) or outside of the input area (above or below).
2. How to give feedback also depends on where the feedback is given. Textual feedback or feedback with symbols (e.g. a stop sign) can be given anywhere. The feedback can be supported by colouring the various components such as messages, field labels or the field itself. Usually green, yellow and red are used to show the different states from valid to invalid data. When colouring a field, there exists variations such as a glow effect or change to the line or background colour.
3. Validation and feedback can take place either when a single data item is entered or after all data in an input area has been entered. A validation time could, for example, be defined as when the focus changes from one field to another or when pressing a specific button. It is also possible to check the input as the user is entering it in the case of an input text field.

If the data submitted by the user is valid, it is good practice to let the user know that everything went as planned. If it is invalid, the user should be informed (1) that an error has occurred, (2) where the error


There was a problem with your request
Your email address has an invalid email address format. Please correct and try again.
Registration
New to Amazon.com? Register Below.
My name is: David Smith
My e-mail address is: david.smith@mydomain

(a) Amazon


Choose your username
david.smith @gmail.com
Someone already has that username. Note that we ignore periods and capitalization in usernames. Try another?
Available: ds4559913

(b) Gmail

Figure 1: Various feedback variants

occurred and (3) how the error can be repaired. Thus, feedback not only involves highlighting where an error occurred but also providing information about what is missing or needs to be changed in order for the data to be valid or of higher quality.
Figure 1a shows an example of validation feedback outside of the input area supported by a warning symbol and a surrounding red line for the message area. Fig. 1b and Fig. 2 show examples of feedback next to the input data, in both cases with examples of correct input. In Fig. 1b, the fields are also highlighted with a red line colour, whereas in Fig. 2 a small bar shows the quality of the password field as the data is entered.

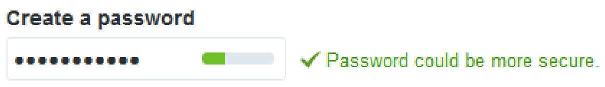

Create a password
•••••••••••
✓ Password could be more secure.

Figure 2: Twitter

Researchers have proposed a number of ways of making it easier for developers to specify validation conditions as well as feedback to be given. PowerForms (Brabrand et al., 2000) is a proposal for a declarative and domain specific language for client-side form field validation. While editing a form field, the data is checked against the specified regular expression and small traffic lights show the validation status in three phases. On submit, validation violations are displayed within a JavaScript alert box. Book et al. (2009) state that to facilitate the ease of use of UIs, users should be guided in ways such as highlighting and describing invalid input, and showing / hiding or enabling / disabling particular UI widgets. They present a formal model and prototype (Cepheus) for enabling the specification of user input evaluation rules and interface responses by domain experts. Groenewegen and Visser (2013) present a sublanguage of WebDSL to handle data validation rules. Error messages are either shown directly at the input field if the input is not well-formed or causes a violation of a data invariant, or, at the form element that triggered the execution process (e.g. a submit button) if a data invariant was violated during execution.

DQAF In contrast to DQIF, the user gets feedback about multiple or all entities in a database at the same time in DQAF. Therefore, the feedback should include ways of identifying areas where a correction might be required. Many different visualisations based for example on charts and tables can be used to give the appropriate feedback. It is also important to provide a means of navigating from overviews to individual data entities and their associated violations in order to identify problems.
Data cleaning tasks such as the removal of duplicate records, data standardisation and data profiling are described in Ganti and Sarma (2013). The authors provide a guide offering practical advice on options available for building or choosing a data cleaning solution. Such solutions also include the automated correction of invalid data once the defects have been determined. The prime example used in data cleaning is the domain of address data where tools such as SQL Power DQguru[1], OpenRefine[2] and Wrangler (Kandel et al., 2011) have been developed.
Methods for error detection that go beyond integrity analysis are reviewed and presented in Maletic and Marcus (2000). The applicable methods include: statistical outlier detection, pattern matching, clustering and data mining techniques.

[1] http://www.sqlpower.ca/page/dqguru
[2] http://openrefine.org

Studies on data feedback already exist in domains other than food science. For example, the study of Bradley et al. (2004) in the healthcare domain illustrates the diversity of hospital-based efforts for data feedback and highlights successful strategies and common pitfalls in designing and implementing data feedback to support performance improvement. They came up with seven key themes such as'The source and timeliness of data are critical to perceived validity'.
The initial DQ evaluation procedures developed by the US Department of Agriculture (USDA) were manual processes to assess the quality of analytical data for iron, selenium and carotenoids in foods. In the course of a redesign of the software system used at the USDA, these procedures were taken a step further and a generic system was developed (Holden et al., 2002). The five original evaluation categories Sampling Plan, Number of Samples, Sample Handling, Analytical Method and Analytical Quality Control were maintained, but the quality assessment questions were made more objective. According to the answers to these questions, a single numeric Quality Index (QI) is assigned to a nutritional component. At aggregation, a Confidence Code (CC) is assigned to the combined value, which is calculated as the weighted mean of the individual values from the different sources of data. The CC is derived from the QIs by summing up the adjusted ratings of the individual values.
The European Food Information Resource (EuroFIR) developed a QI (Salvini et al., 2007) similar to that of USDA. In fact, they adopted the five categories from the USDA QI and additionally added Food Description and Component Identification. For all of the seven categories, a set of questions is defined. There are 34 questions in total, which, all except one, can be answered with Yes, No or Not Applicable.
Although several conceptual frameworks for DQ have been proposed, there is still a lack of general tools and metrics to measure and control the quality of data in practice. As a first step in this direction, we carried out a detailed study of DQ requirements for an information system to manage food composition data (Presser et al., 2014). Our users included system designers and developers as well as food compilers and project managers. In addition to determining which dimensions of DQ specified in existing conceptual frameworks users consider important in assessing the reliability of data, we also asked users to assess the importance of various criteria related specifically to empirical data.
Based on the results of this analysis, we integrated enhanced functionality for defining, validating, measuring and visualising DQ into FoodCASE[3], a system for managing the Swiss Food Composition Database (SFCD) which is maintained by the Swiss Food Information Resource (SwissFIR)[4] of ETH Zurich and the Federal Office of Public Health[5]. Generally, FoodCASE is used to document data for studies in the food sciences, such as foods with their associated analysed values for nutrients and contaminants. The stored data is then usually used for further assessment. Based on constraint definitions, FoodCASE provides the measurements, calculations and visualisation components for visualising DQ and helps users monitor constraints and DQ. The concept and implementation includes the validation and feedback for both *DQIF* and *DQAF*.
In addition to performing constraint validation during transaction execution as supported in traditional databases, it is also possible to trigger DQ analysis which will use constraints to measure the quality of data. Our DQ framework is general and application developers can define their own DQ measurements and configure customised DQ feedback. We identified 156 DQ requirements specific to the food science domain which are validated in FoodCASE and to which the end user gets DQ feedback. In the following sections, we provided details of the DQ analysis toolkit developed in the context of the FoodCASE project and report on an evaluation of what and how DQ feedback is given to users.

[3] http://www.foodcase.ethz.ch
[4] http://www.swissfir.ethz
[5] http://www.bag.admin.ch

3 Approach

Our DQ analysis toolkit is based on the concept of a Requirement-Oriented Data Quality Model (RODQ) developed by Presser (2012). A RODQ model is a conceptual model that describes the DQ requirements of an information system independently of its implementation. In this respect, it complements other established modelling languages such as ERM[6] and UML[7]. Furthermore, the RODQ model describes how to assess the DQ.

The central element of an RODQ model is the DQ Requirements (DQRs) where Presser (2012) distinguishes three types of DQRs:

- A hard constraint (HC) is a DQR which absolutely must be fulfilled. If an HC is violated, the data is invalid and cannot be used. Therefore, the system should enforce HCs and not allow input data to be stored unless all HCs are satisfied.
- A soft constraint (SC) is a DQR which is highly recommended to be fulfilled. If an SC is not satisfied, DQ decreases. However, it might not always be possible to adhere to all SCs. Hence, they cannot be enforced by the system.
- An indicator is a property that can be used to estimate DQ, rather than a constraint that can be clearly satisfied or not. A typical example is the age of the data. If the data is old, it is likely to be outdated, but it is still possible that it is correct.

As already introduced, we distinguish between two types of validation and DQ feedback for the user, namely DQIF and DQAF. Both make use of the DQRs defined in the information system. DQIF is only concerned with the DQRs of the input data entity which can include relations to other entities whereas DQAF analyses all DQRs in the information system. For each type, we will present an example from the FoodCASE system and describe briefly how feedback is given to the user. Due to space limitations, we will only give an overview of the main features. A detailed description of the toolkit can be found in Mock (2011).

Figure 3 shows an example of the DQIF for a food entity. On every screen where data can be edited, there exists a DQ evaluation panel at the bottom. This panel lists all problems of the current data record. Problems are divided into two categories: errors and warnings. Errors correspond to the violation of HCs and are displayed on the panel in red: If any are present, the data cannot be saved. Warnings corresponding to the violation of SCs and presence of low quality indicators are displayed on the panel in orange. No additional background colour change is triggered for warnings.

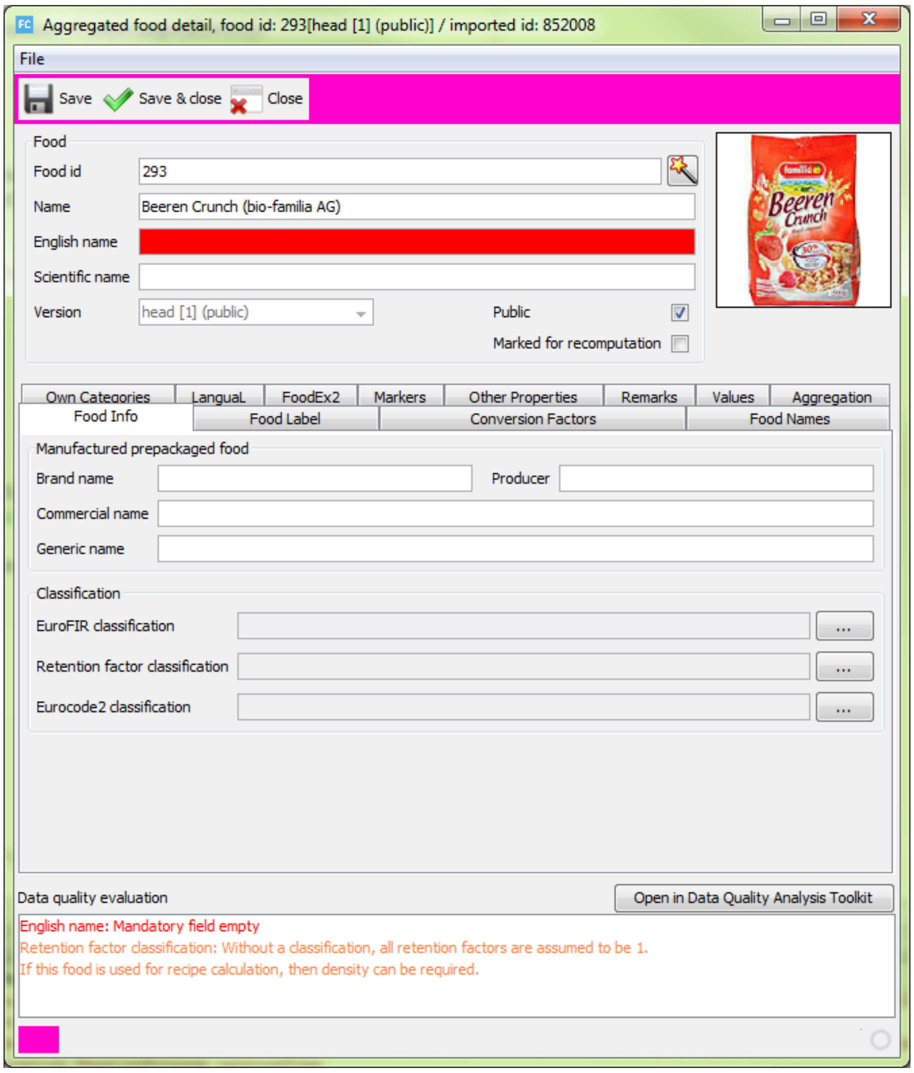
The example shows a food entity with the violation of one HC and two SCs. The appropriate DQRs are: (1) Attribute'English name' is mandatory and cannot be empty (HC), (2) The attribute'Retention factor classification' is not mandatory but recommended. It is possible to leave it empty but consequently all retention factors are assumed to have the value one (SC), (3) The attribute'Density' (not visible in the screenshot since it is on a subtab of the input area) is not mandatory but can be required if the food is used for a recipe calculation (SC). On the upper right of the DQ evaluation panel, a shortcut button provides quick access to the data record in the DQ analysis toolkit.

Concerning the *where*, *how* and *when* of giving feedback introduced in Sect. 2, we decided to give all textual feedback in a single location below the input area with the colouring indicating the severity of the problems. We only use the evaluation panel for textual feedback since the error and warning messages are often rather long and it would be difficult to show an abbreviation close to the appropriate field with such a high density of information and UI components as in FoodCASE. We also indicate where HCs have been violated by changing the background colour of appropriate fields. This is only done for HCs to keep the colouring within the input area to a minimum. The validation is triggered when the focus is moved from a field and consequently feedback is given immediately on data input for single components in the input area. We do not check the input as the user is entering data since we wanted to reduce the

[6] Entity-Relationship-Model

[7] Unified Modelling Language

client server communication for performance reasons. The DQ evaluation panel also shows hints for how problems can be repaired if this information has been provided with the appropriate DQR.



Aggregated food detail, food id: 293[head [1] (public)] / imported id: 852008

Food

Food id	293
Name	Beeren Crunch (bio-familia AG)
English name	🟥
Scientific name	
Version	head [1] (public)
Public	✓
Marked for recomputation	

Manufactured prepackaged food

Brand name	
Producer	
Commercial name	
Generic name	

Classification

EuroFIR classification	
Retention factor classification	
Eurocode2 classification	

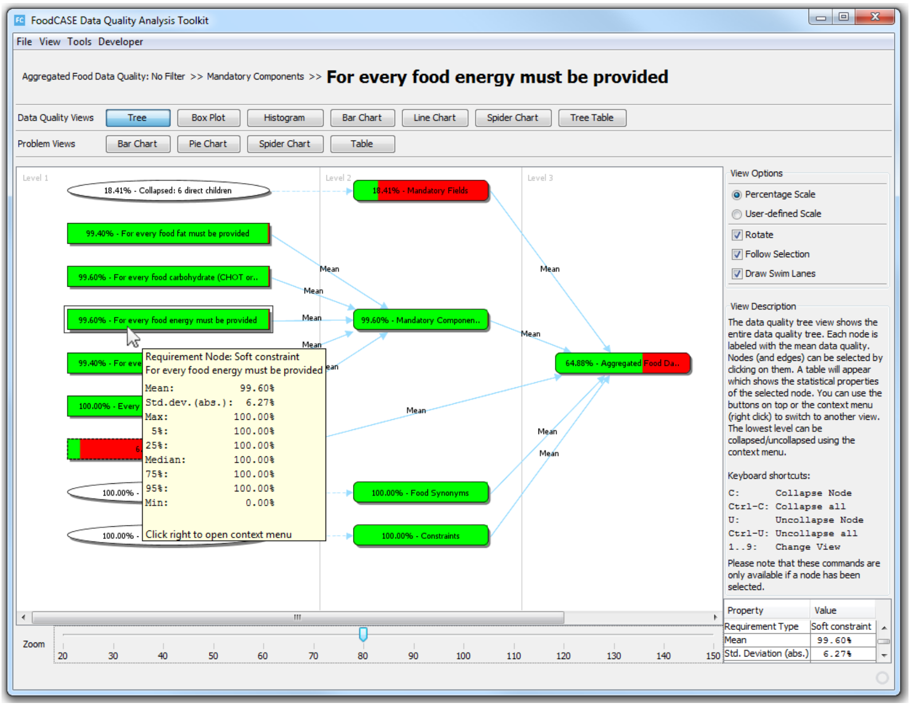
Data quality evaluation

English name: Mandatory field empty
Retention factor classification: Without a classification, all retention factors are assumed to be 1.
If this food is used for recipe calculation, then density can be required.

Figure 3: DQIF

For the DQAF, our DQ analysis toolkit first runs a DQ assessment where all relevant data is fetched from the database, all DQRs checked, the presentation of the DQ prepared and the results stored in the database for later analysis. Thus, it can be considered as a snapshot of the current DQ in the system. A DQ assessment can be triggered manually by the system administrator or automatically by a timer. Upon completion, a variety of different charts and tables support the user in judging where action is needed.

In our toolkit, every DQR is assigned a type (HC / SC / indicator) which determines how the associated DQ measures will be rendered. The type of the DQR will also be taken into account by the users when specifying the weights (importance) of the DQRs. The DQ analysis toolkit provides a total of eleven different views which can be divided into two categories: (1) *DQ views* provide an overview of the DQ by visualising the DQ of either a selected tree node and its direct children or the entire DQ tree. (2) *Problem views* on the other hand provide the possibility to drill down on the DQ issues and identify individual problems. The toolkit provides flexible means of allowing users to configure their own analysis tree definitions and analysis runs. It is beyond the scope of the paper to describe this in detail, but we show an example of one of the many possible visualisations in Fig. 4 where the feedback is shown as a tree. For a food entity, there exists a DQR specifying that a value for the attribute energy should be provided since various calculations depend on it. This is defined as an SC since it is not something that can be enforced by the system at all points in time. The example shows a custom tree view for aggregated food entities to analyse mandatory components in detail. The DQ tree view shows the entire DQ tree with each node labelled with the mean DQ. Nodes (and edges) can be selected by clicking on them. A table will appear which shows the statistical properties of the selected node. To switch to another view, the buttons at the top or the context menu can be used. The lowest level can be collapsed / uncollapsed using the context menu or keyboard shortcuts. All nodes are rendered as progress bars indicating the DQ of the criteria they represent. The DQR‘For every food energy must be provided’ is displayed as a subnode of‘Mandatory Components’ and‘Aggregated Food Data’. By default, all the DQ values in the DQ analysis toolkit are displayed as percentages. In this example, it can be seen that only 99.6% of the data fulfil this DQR.


FoodCASE Data Quality Analysis Toolkit
File View Tools Developer
Aggregated Food Data Quality: No Filter >> Mandatory Components >> For every food energy must be provided
Data Quality Views
Tree
Box Plot
Histogram
Bar Chart
Line Chart
Spider Chart
Tree Table
Problem Views
Bar Chart
Pie Chart
Spider Chart
Table
Level 1
Level 2
Level 3
18.41% - Collapsed: 6 direct children
99.40% - For every food fat must be provided
99.60% - For every food carbohydrate (CHOT or..
99.60% - For every food energy must be provided
99.40% - For eve
100.00% - Every
100.00% -
100.00% -
18.41% - Mandatory Fields
99.60% - Mandatory Componen..
100.00% - Food Synonyms
100.00% - Constraints
64.88% - Aggregated Food Da..
Mean
Zoom
20
30
40
50
60
70
80
90
100
110
120
130
140
150
View Options
Percentage Scale
User-defined Scale
Rotate
Follow Selection
Draw Swim Lanes

```
Requirement Node: Soft constraint
For every food energy must be provided
Mean:            99.60%
Std.dev.(abs.):   6.27%
Max:            100.00%
 5%:            100.00%
25%:            100.00%
Median:         100.00%
75%:            100.00%
95%:            100.00%
Min:              0.00%

Click right to open context menu
```

View Description

The data quality tree view shows the entire data quality tree. Each node is labeled with the mean data quality. Nodes (and edges) can be selected by clicking on them. A table will appear which shows the statistical properties of the selected node. You can use the buttons on top or the context menu (right click) to switch to another view. The lowest level can be collapsed/uncollapsed using the context menu.

Keyboard shortcuts:

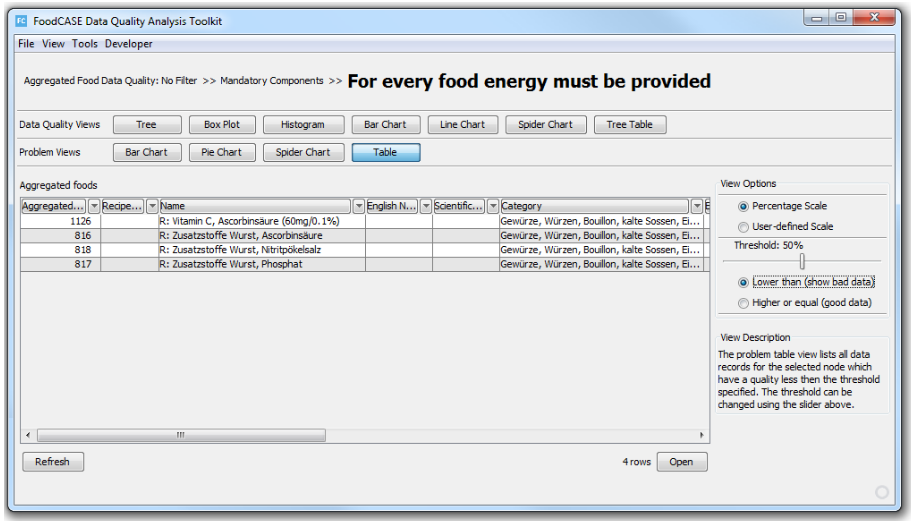
```
C:      Collapse Node
Ctrl-C: Collapse all
U:      Uncollapse Node
Ctrl-U: Uncollapse all
1..9:   Change View
```

Please note that these commands are only available if a node has been selected.

Property	Value
Requirement Type	Soft constraint
Mean	99.60%
Std. Deviation (abs.)	6.27%

Figure 4: Tree visualisation of DQAF

From this overview, it is possible to drill down on the DQ issues and identify individual problems in the data. The user can switch to the problem table view which is depicted in Fig. 5. It shows that four aggregated foods have been identified where the component energy is missing. By double-clicking on a row, the data record is opened in the appropriate FoodCASE entity editor screen and the problem can be fixed. We have chosen to visualise the DQAF with the most common charts such as Box Plot, Histogram, Bar Chart, Line Chart, Spider Chart, Pie Chart, Table and Tree Table since we think that with this choice of visualisations it is possible to analyse the various facets of DQ very well and that these types are already familiar to users. The Tree serves as our main DQ view since it gives the best overview and navigation options.


FoodCASE Data Quality Analysis Toolkit
File View Tools Developer
Aggregated Food Data Quality: No Filter >> Mandatory Components >> For every food energy must be provided
Data Quality Views
Tree
Box Plot
Histogram
Bar Chart
Line Chart
Spider Chart
Tree Table
Problem Views
Bar Chart
Pie Chart
Spider Chart
Table
Refresh
Open

Aggregated foods

Aggregated...	Recipe...	Name	English N...	Scientific...	Category
1126		R: Vitamin C, Ascorbinsäure (60mg/0.1%)			Gewürze, Würzen, Bouillon, kalte Sossen, Ei...
816		R: Zusatzstoffe Wurst, Ascorbinsäure			Gewürze, Würzen, Bouillon, kalte Sossen, Ei...
818		R: Zusatzstoffe Wurst, Nitritpökelsalz			Gewürze, Würzen, Bouillon, kalte Sossen, Ei...
817		R: Zusatzstoffe Wurst, Phosphat			Gewürze, Würzen, Bouillon, kalte Sossen, Ei...

4 rows

View Options

[✓ Percentage Scale] User-defined Scale

Threshold: 50%

[✓ Lower than (show bad data)] Higher or equal (good data)

View Description

The problem table view lists all data records for the selected node which have a quality less then the threshold specified. The threshold can be changed using the slider above.

Figure 5: Problem table view

According to Presser (2012), a DQR is always associated with a DQ object, where a DQ object corresponds to a real world object on which the DQ check should be performed. In our DQ analysis toolkit, the DQ objects are the database entities corresponding to the business concepts of food science data. In the following, we will refer to these entities as the DQ entities. Each DQR has to map every data record in the DQ entity to a DQ value between 0.0 (worst) and 1.0 (best). If a DQR is not applicable to a certain data record, NULL may be returned. For example, if a database table contains data about customers, a DQR could be defined as‘For every person, last name and first name must be provided’.

Now if there is a customer for whom only the last name but not the first name is known, the DQR is only partially fulfilled. So the DQ could be defined to be 0.7 (70%) because usually the last name is more important than the first name. If a customer is not a person but a company, the DQR is not applicable, so NULL should be returned.

Once all DQRs are gathered, similar DQRs can be grouped together. This process can be repeated recursively ending up with a DQ tree. The root node will then be a single number representing the overall DQ. Since the grouping of the DQRs may be a matter of individual taste, it is possible to define different DQ analysis trees using the same DQRs. Similarly, the importance of the DQR may be controversial. Because of this, the weight of each DQR can be specified in each tree definition independently.

4 Implementation

The FoodCASE system is composed of six components which are depicted in the high level architecture presented in Figure 6: (1) A database which stores all the business data of the application. (2) An application server which runs EJB3 session beans. They provide services which can be used by the client modules. The main responsibility of the EJB layer is to take care of persistence and make it transparent to the clients, so that they can work directly with the business objects. (3) A Content Management System (CMS) allowing the food compilers to manage the food composition data in the system. (4) An administration module for the system administrators. Among other settings, this module contains the user and the thesauri administration. The latter includes the units of measurement that can be used in the system, the food components and a lot of different food and component categorisations. (5) A web page which allows the public to query information about the composition of the foods available in Switzerland. (6) A web service to export a single food item or the whole database as a EuroFIR Food Data Transport Package (FDTP) which is a standardised XML[8] format for food composition data interchange in Europe.

As user interface, FoodCASE provides a Java Swing GUI and Java Web Start. The backend is implemented using EJB3[9] session beans running on a JBoss AS 7.1.1.Final[10] application server. A PostgreSQL 9.3[11] database serves as the persistent data storage. Most of the persistence logic is implemented in JPQL[12]. Some batch operations use plain SQL for the sake of better performance. With the choice of Java and its Web Start environment, our fat client can be easily run on any platform without a difficult installation and update procedure. We required the most advanced open source application server and relational database and therefore chose JBoss and PostgreSQL at the beginning of the project.

The data quality (DQ) requirements are defined in custom Java validation classes for the DQIF whereas the DQAF makes use of SQL statements for the validation. A detailed description of the implementation is provided in Mock (2011).

As an indication of the performance of the DQAF integrated in FoodCASE, the calculation of all DQ requirements represented in a complex DQ tree for the Swiss Food Composition Database, including the subsequent rendering, takes several minutes, e.g. a quarter hour.

[8] Extensible Markup Language
[9] Enterprise JavaBeans: http://www.oracle.com/technetwork/java/javaee/ejb/index.html
[10] http://jbossas.jboss.org/
[11] http://www.postgresql.org
[12] Java Persistence Query Language

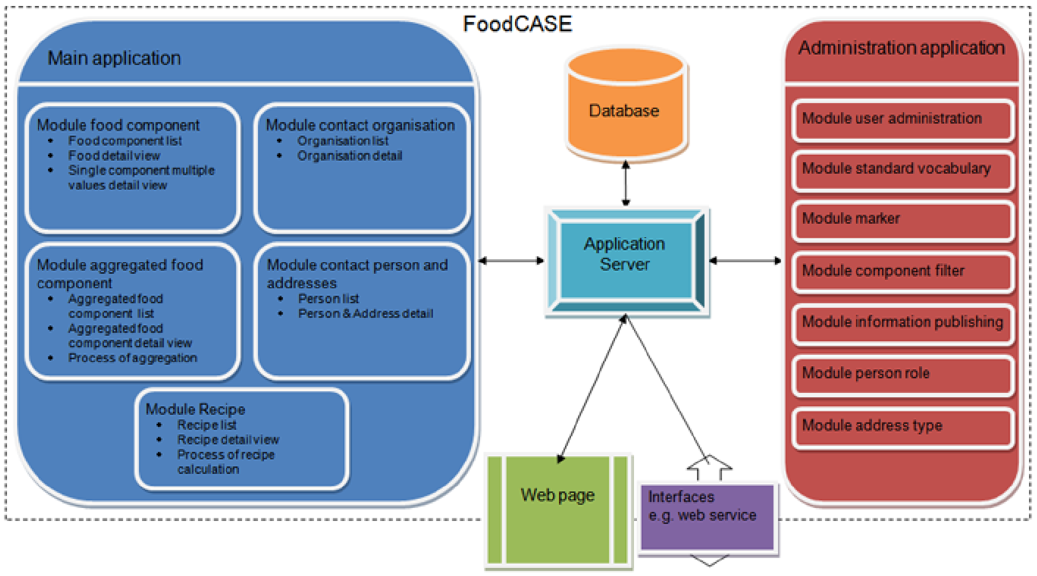

FoodCASE
Main application
Module food component
Food component list
Food detail view
Single component multiple values detail view
Module contact organisation
Organisation list
Organisation detail
Module aggregated food component
Aggregated food component list
Aggregated food component detail view
Process of aggregation
Module contact person and addresses
Person list
Person & Address detail
Module Recipe
Recipe list
Recipe detail view
Process of recipe calculation
Database
Application Server
Web page
Interfaces e.g. web service
Administration application
Module user administration
Module standard vocabulary
Module marker
Module component filter
Module information publishing
Module person role
Module address type

Figure 6: Architecture of the FoodCASE system. The six modules are highlighted in different colours.

5 Evaluation

To evaluate how the data quality (DQ) feedback of our information system is perceived by users and how it can be improved, we presented our information system *FoodCASE* with the integrated DQ analysis toolkit to food science experts and subsequently conducted a survey.
As part of an annual general assembly meeting of the European project Total Diet Study Exposure[13] (TDS-Exposure), we organised a workshop with several food science data experts from different countries in Europe. The participants had a variety of roles including food compiler, head of unit, project coordinator, exposure assessment researcher, food database manager, research assistant, scientific manager / assistant, post doctoral researcher and PhD student. We used half a day to present the information system with the different possibilities for managing scientific food data explained by using screenshots of the various graphical user interfaces and showing how to enter data for various use cases. After the presentation of the information system and a subsequent extensive discussion in the plenum, we provided a questionnaire which the participants had to fill out. A total of 16 participants completed the questionnaire, which, although a relatively small number, included food science experts from most of the partners involved in the TDS-Exposure project.
The questionnaire included 49 questions on various topics such as ‘DQ in general’, ‘Who should be able to define DQ requirements’, ‘How should DQ requirements be defined’, ‘How should DQ feedback be given to users’, ‘Distinction between hard constraints (HC) and soft constraints (SC)’ and ‘Personal information and user skills’. For the sake of simplicity, we only distinguished between HCs and SCs in the questionnaire by including the indicators in the category of SCs. For each question, the participants could choose from a five-level Likert-type scale if they ‘Strongly agree’ (SA), ‘Agree’ (A), are ‘Neutral’ (N), ‘Disagree’ (D) or ‘Strongly disagree’ (SD).
We present here the most interesting findings from the survey. First, we give an overview of the results categorised by the question topics mentioned above excluding the ‘Personal information and user skills’ topic. Due to space limitations, we present the results in tabular form rather than using detailed charts. The entries in the tables designate the percentages of participants giving a particular answer. At the end of this section, we summarise the results by providing a short list of the most important key findings for DQ feedback.

[13] http://www.tds-exposure.eu/

#	Finding	SA%	A%	N%	D%	SD%
1	An information system should clearly offer the functionality to analyse and improve DQ.	25	75	-	-	-
2	Over 80% think that data changes should be stored with their data provenance.	43.75	37.5	12.5	6.25	-
3	Over 60% think that the information system should give continuous DQ feedback and request adjustment of the data.	25	37.5	37.5	-	-
4	For about 50% of the participants it is necessary to have an analysis functionality at a certain point of time. The rest seems to be confident with just the input data validation.	25	31.25	43.75	-	-
5	For 50% of the users DQ should be displayed with diagrams and graphs.	12.5	37.5	50	-	-
6	For over 50% it would be preferable to have an automatic adjustment mechanisms that corrects data of poor quality but there exist clearly votes against such an automation.	6.25	50	25	6.25	12.5
7	75% think that a data analysis mechanism should be of high performance.	18.75	56.25	25	-	-
8	75% agree that the access to data should be controlled and restricted.	43.75	31.25	18.75	6.25	-
9	The possibility to recall the change history of data is clearly important.	37.5	62.5	-	-	-

Table 1: DQ in general

#	Finding	SA%	A%	N%	D%	SD%
1	Almost 90% think that the institution using the software should be able to define DQ requirements.	12.5	75	12.5	-	-
2	Also almost 60% think that it would be preferable if a power user, being a non IT person, should be able to define DQ requirements.	18.75	37.5	37.5	-	6.25
3	Whereas it seems not preferable that an IT administrator in the institution who has some programming skills should be able to define DQ requirements.	-	31.25	43.75	12.5	12.5
4	And whereas it seems not preferable that only developers should be able to define DQ requirements.	-	6.25	18.75	62.5	12.5
5	And whereas it seems not preferable that every user should be able to define DQ requirements.	6.25	6.25	31.25	18.75	37.5
6	Over 50% think that it is not preferable that every user should learn a simplified programming language to define basic DQ requirements.	6.25	31.25	6.25	43.75	12.5

Table 2: Who should be able to define DQ requirements

#	Finding	SA%	A%	N%	D%	SD%
1	For most of the users it is not necessary that DQ requirements are defined by means of graphical elements.	6.25	37.5	56.25	-	-
2	Almost 60% think that DQ requirements should be defined by means of textual definition language.	18.75	37.5	43.75	-	-

3	Almost 70% think that it is preferable that DQ requirements should be managed in one place and in a unified way.	37.5	31.25	25	6.25	-
4	Almost 67% think that it should be possible to somehow combine and aggregate DQ requirements to get an overall statement about DQ and 25% disagree with that.	6.25	50	18.75	25	-
5	All users agree that it should be possible to rate the relevancy of each DQ requirement because not all requirements have the same importance.	25	75	-	-	-
6	Over 90% think that a summary such as ’5 of 9 DQ requirements are satisfied‘ would motivate users to increase DQ.	25	68.75	6.25	-	-
7	Only 50% think that it should be possible that single DQ requirements can be deactivated to be able to reduce the time to save a data record and 25% disagree with that.	18.75	31.25	25	12.5	12.5
8	It is clearly not preferable to give the possibility to deactivate all DQ requirements.	6.25	-	31.25	31.25	31.25
9	75% think that if users are able to define their own DQ requirements, it would be helpful to see other users’ DQ requirements.	18.75	56.25	18.75	-	6.25
10	But only 50% think that it would be helpful if users could mark poor or good DQ directly on data elements.	-	50	35.71	14.29	-

Table 3: How should DQ requirements be defined

Note that only 14 of the participants answered the question leading to result #10 of Table 3.

#	Finding	SA%	A%	N%	D%	SD%
1	Over 90% think that a panel at the bottom of a GUI (as presented in FoodCASE) is a good solution for the DQ feedback in an application.	46.66	46.66	-	6.66	-
2	Almost 20% would prefer a panel above the input area but also 25% disagree with that.	12.5	6.25	56.25	25	-
3	Over 60% agree that it would be better if DQ feedback was displayed close to the corresponding field(s) instead of the DQ panels at the bottom or at the beginning and nobody disagrees with that.	12.5	50	37.5	-	-
4	50% think that if the DQ feedback is next to the field(s), then a simple text message is fine.	6.25	43.75	37.5	12.5	-
5	An accentuation such as highlighting or colouring the corresponding field(s) is rated by over 80% as helpful.	25	56.25	18.75	-	-
6	75% think that the usage of colours to indicate different types of DQ requirements such as relevance are helpful.	18.75	56.25	25	-	-
7	Again the large majority (here over 80% instead of over 90% as in the definition part) think that providing a short DQ summary such as 7 of 13 points are reached, would motivate users to improve DQ.	18.75	62.5	12.5	6.25	-

Table 4: How should DQ feedback be given to users

Note that only 15 of the participants answered the question leading to result #1 of Table 4.

#	Finding	SA%	A%	N%	D%	SD%
1	Almost 90% agree that the distinction between HCs and SCs is useful.	37.5	50	12.5	-	-
2	75% disagree that the distinction between HCs and SCs is not necessary.	-	-	25	50	25
3	75% disagree that HCs should not exist although almost 20% agree.	6.25	12.5	6.25	25	50
4	Also 75% disagree that SCs should not exist although 12.5% agree.	-	12.5	12.5	25	50
5	Almost 70% think that the number of HCs should be as small as possible.	25	43.75	25	6.25	-
6	Almost 70% prefer to have more SCs than HCs.	12.5	56.25	31.25	-	-
7	Less than 45% think that a user should be able to configure whether only HCs or both HCs and SCs should be regarded in the feedback.	-	43.75	31.25	18.75	6.25
8	Only 25% think that the information system should give feedback about the SCs only on demand and over 40% disagree with that.	6.25	18.75	31.25	43.75	-

Table 5: Distinction between HCs and SCs

Based on the results above, we now present a short list of the most important key findings for DQ feedback in the domain of food science data.

1. An information system should clearly offer the functionality to analyse and improve DQ. (Tab. 1, Finding # 1)
2. Access to data should be controlled and restricted. At the same time, providing users with the possibility to recall the change history of data is clearly important where changes should be traceable to users. (Tab. 1, Finding # 2, 8, 9)
3. It should be possible for the institution using the software to define DQ requirements, but this should be done by a non-IT person and limited to 'power users'. (Tab. 2, Finding # 1, 2, 3, 4, 5)
4. DQ requirements should be managed in one place and in a unified way. (Tab. 3, Finding # 3)
5. It should be possible to rate the relevancy of each DQ requirement because not all requirements have the same importance. (Tab. 3, Finding # 5)
6. It is clearly not preferable to provide the possibility to deactivate all DQ requirements. (Tab. 3, Finding # 8)
7. A summary such as '5 of 9 DQ requirements are satisfied' would motivate users to increase DQ. (Tab. 3, Finding # 6 and Tab. 4, Finding # 7)
8. A panel at the bottom of a GUI is a good solution for the DQ feedback but, for a lot of users, it would be preferable if DQ feedback was displayed close to the corresponding field(s). (Tab. 4, Finding # 1, 3)
9. An accentuation such as highlighting or colouring the corresponding field(s) and the usage of colours to indicate different types of DQ requirements is considered helpful. (Tab. 4, Finding # 5, 6)
10. The distinction between HCs and SCs is useful and both types should exist. (Tab. 5, Finding # 1, 2, 3, 4)

11. The number of HCs should be as small as possible with most users prefering to have more SCs than HCs. (Tab. 5, Finding # 5, 6)

6 Conclusions

We have designed and implemented a general data quality (DQ) analysis toolkit which allows DQ in information systems to be measured and visualised in a customised way. To evaluate the toolkit, we integrated it into the FoodCASE system for managing food composition data and solicited feedback from experts in the domain.

The toolkit allows users in a particular domain to define their own DQ requirements (DQR) and configure the metrics for measuring DQ as well as how results of DQ analysis are visualised. In the case of the food science domain, this means that the system is not limited to the quality measures as defined by the USDA and EuroFIR and can easily be extended to meet the requirements of a particular institution or group of users. For example, we have identified and defined 156 DQRs for the food science domain in total which our toolkit is able to validate and for which DQ feedback can be provided. Furthermore, it is possible to define whether the current DQ should be analysed or a historical snapshot is used. Once it is recognised that there are problems in a certain area of the data, the toolkit allows the user to quantify the extent of the problem and drill down on them.

Based on feedback from experts in the food sciences, we presented a list of key findings indicating the needs for users to get DQ feedback. Based on these findings, we conclude with a summary of the features that our toolkit already supports followed by proposals for future extensions and enhancements.

Our information system FoodCASE already provides the following: (1) The functionality to analyse and improve DQ. (2) The access to data is controlled and restricted. (3) It is possible to recall the change history and each change of data is traceable to a given user. (4) DQRs can be defined by certain power users. (5) It is possible to rate the relevancy of each DQR by defining custom DQ feedback trees with custom weights of certain DQRs. (6) The definition of DQRs as hard constraints, soft constraints or indicators is possible and the user is free to decide how to categorise individual DQRs. (7) Feedback distinction between different types of DQRs is visualised in the DQ evaluation panel at the bottom of each screen.

Based on the evaluation outcome, we plan to: (1) Define and manage the DQRs in one place in a unified way, probably by changing the DQR definitions to annotations in the entities and performing the validation by the means of BeanValidation (Java Specification Request 349)[14]. (2) Provide a DQ summary in each screen such as '5 of 9 DQRs are satisfied' in order to motivate users to increase DQ. (3) Keep our evaluation panel for detailed feedback but add short DQ feedback close to the appropriate fields, probably by an additional accompanied symbol such as a light bulb with attached tooltip.

Acknowledgement

The work was financially supported by the European FP7 programme as part of the TDS-Exposure project. We want to thank Reto Mock for his work that contributed to the DQ analysis toolkit and the participants of the TDS-Exposure project for their support.

[14] https://jcp.org/en/jsr/detail?id=349

References

Batini, C. and M. Scannapieco (2006). *Data Quality: Concepts, Methodologies and Techniques*. Data-Centric Systems and Applications. Springer-Verlag New York, Inc.

Book, M. et al. (2009). "Specification and Control of Interface Responses to User Input in Rich Internet Applications." In: *Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering*. Washington, DC, USA, 321–331. URL: `%7Bhttp://dx.doi.org/10.1109/ASE.2009.10%7D`.

Brabrand, C. et al. (2000). "PowerForms: Declarative client-side form field validation." *World Wide Web* 3 (4), 205–214.

Bradley, E. et al. (2004). "Data feedback efforts in quality improvement: lessons learned from US hospitals." *Quality & safety in health care* 13 (1), 26–31.

Ganti, V. and A. D. Sarma (2013). *Data Cleaning: A Practical Perspective*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers.

Groenewegen, D. M. and E. Visser (2013). "Integration of data validation and user interface concerns in a DSL for web applications." *Software & Systems Modeling* 12 (1), 35–52.

Holden, J. M. et al. (2002). "Development of a Multi-nutrient Data Quality Evaluation System." *Journal of Food Composition and Analysis* 15 (4), 339–348.

Kandel, S. et al. (2011). "Wrangler: Interactive Visual Specification of Data Transformation Scripts." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 3363–3372.

Madnick, S. et al. (2009). "Overview and Framework for Data and Information Quality Research." *Data and Information Quality* 1 (1).

Maletic, J. I. and A. Marcus (2000). "Data cleansing: Beyond integrity analysis." In: *Proceedings of the Conference on Information Quality*, pp. 200–209.

Mock, R. (2011). "Data Quality Analysis for Food Composition Databases." MSc Thesis. Zurich, Switzerland: ETH Zurich.

Presser, K. (2012). "A Requirement-Oriented Data Quality Model and Framework of a Food Composition Database System." PhD thesis. Zurich, Switzerland: ETH Zurich.

Presser, K. et al. (2014). "A Study of Data Quality Requirements for Empirical Data in the Food Sciences." In: *European Conference on Information Systems (ECIS)*. Tel Aviv, Israel.

Salvini, S. et al. (2007). *Guidelines for Quality Index Attribution to Original Data from Scientific Literature or Reports for EuroFIR Data Interchange*. URL: `http://www.eurofir.org/?page_id=2667`.

Wand, Y. and R. Wang (1996). "Anchoring Data Quality Dimensions in Ontological Foundations." *Communications of the ACM* 39 (11).

Wang, R., M. Reddy, et al. (1995). "Toward quality data: An attribute-based approach." *Decision Support Systems* 13 (3-4).

Wang, R. and D. Strong (1996). "Beyond Accuracy: What Data Quality Means to Data Consumers." *J. Manage. Inf. Syst.* 12 (4), 5–33. ISSN: 0742-1222.