Management of Data Quality Related Problems
Exploiting Operational Knowledge

Mortaza S. Bargh[1], Jan van Dijk[1] and Sunil Choenni[1,2]
[1]*Research and Documentation Centre, Ministry of Security and Justice, The Hague, Netherlands*
[2]*Research Centre Creating 010, Rotterdam University of Technology, Rotterdam, Netherlands*

Keywords: Data Quality Issues, Data Quality Management, Knowledge Mapping, User Generated Inputs.

Abstract: Dealing with data quality related problems is an important issue that all organizations face in realizing and sustaining data intensive advanced applications. Upon detecting these problems in datasets, data analysts often register them in issue tracking systems in order to address them later on categorically and collectively. As there is no standard format for registering these problems, data analysts often describe them in natural languages and subsequently rely on ad-hoc, non-systematic, and expensive solutions to categorize and resolve registered problems. In this contribution we present a formal description of an innovative data quality resolving architecture to semantically and dynamically map the descriptions of data quality related problems to data quality attributes. Through this mapping, we reduce complexity – as the dimensionality of data quality attributes is far smaller than that of the natural language space – and enable data analysts to directly use the methods and tools proposed in literature. Furthermore, through managing data quality related problems, our proposed architecture offers data quality management in a dynamic way based on user generated inputs. The paper reports on a proof of concept tool and its evaluation.

1 INTRODUCTION

Organizations and enterprises that realize and operationalize data intensive applications spend a lot of efforts and resources to deal with imperfections flaws, and problems in the (large and heterogeneous) datasets that they use as raw materials. For example, in our research center of the Dutch Ministry of Security and Justice, advanced applications are designed and deployed to produce insightful reports on judicial processes and crime trends for legislators, policymakers and the public. Example applications include Public Safety Mashups (Choenni and Leertouwer, 2010) and Elapsed Time Monitoring System of Criminal Cases (Netten et al., 2014). These applications rely on various datasets – as collected and shared by our partner organizations – that are integrated by using data warehouse and data space architectures (van Dijk et al., 2013). Often such datasets contain inconsistent, imprecise, uncertain, missing, incomplete, … data values and attributes. Such problems in datasets may cause inaccurate and invalid data analysis outcomes, which can mislead data consumers eventually.

Upon detecting these problems in datasets, data analysts often report them in Issue Tracking Systems (ITSs) in order to address them later on categorically and collectively. There is no standard format for registering these problems and data analysts often describe them in natural languages in a quite freestyle form. For example, in a dedicated ITS, the data analysts in our organization have registered the following observed dataset problems: Not being able to process criminal datasets at a regional scale because the datasets were delivered at a national scale, not being able to carry out trend analysis due to lack of historical criminal data records, or not being able to run concurrent queries due to temporary datasets being distributed across various locations, a problem also reported in (Birman, 2012).

Because data analysts register observed dataset problems in natural languages, categorization of the registered problems based on their freestyle descriptions becomes tedious and challenging. On the one hand, problem descriptions belong to a "natural language space" of high dimensionality and complexity. On the other hand, finding some meaningful categories for these problem descriptions becomes another concern for data analysts. Having

Bargh, M., Dijk, J. and Choenni, S.
Management of Data Quality Related Problems - Exploiting Operational Knowledge.
DOI: 10.5220/0005982300310042
In *Proceedings of the 5th International Conference on Data Management Technologies and Applications (DATA 2016)*, pages 31-42
ISBN: 978-989-758-193-9

meaningful categories means that the problems in every category have similar solutions and can be resolved collectively. In practice, currently data analysts come up with ad-hoc, non-systematic, and expensive solutions to categorize and resolve registered problems.

Problems observed in datasets are generally related to Data Quality (DQ) issues. For instance, the problems in our datasets mentioned above are related to the DQ attributes of completeness and consistency. DQ is a field that is extensively studied in recent years, having a sound theoretical foundation and a rich set of solutions proposed in literature. It seems, therefore, promising to map the registered dataset problems to DQ issues. Hereby one can reduce complexity – as the DQ space dimensionality is far smaller than that of the natural language space – and make use of the DQ methods and tools proposed in literature directly. Mapping the registered problems to DQ issues, nevertheless, is not straightforward.

In this contribution, we aim at managing and resolving the dataset problems detected by data analysts through mapping them to DQ issues and making use of DQ management tools. (Note that we shall use terms "DQ related problems" and "DQ issues" to refer to dataset problems as described in natural language space and to refer to DQ issues as described in the DQ space, respectively.) To this end, we propose a functional architecture for

a) Semantically mapping the linguistic descriptions of such problems to DQ issues,
b) Automatically prioritizing the severity levels of DQ issues,
c) Automatically categorizing DQ related problems according to the priority levels of the corresponding DQ issues, and
d) Resolving DQ related problems based on their categories, which depend on the severity levels of the corresponding DQ issues.

When data analysts resolve these DQ related problems, they also carry out DQ management. As a by-product, therefore, the proposed architecture provides organizations with insight into their DQ issues in a dynamic (i.e., real-time) way, relying on user-generated inputs (i.e., the problem descriptions inserted by data analysts). From this perspective, our proposed architecture to map high-dimensional DQ related problems into low-dimensional DQ issues is inspired by (Davenport and Glaser, 2002) that aims "to bake specialized knowledge into the jobs of highly skilled workers" in order to take advantage of the rich body of knowledge in a field. By mapping the DQ related problems to DQ issues, we can look up the literature and tools that pertain to resolving the mapped DQ issues. Subsequently, the DQ related problems are solved according to the latest insights and tools. The current work extends our early results (Bargh et al., 2015b; Bargh et al., 2015c) by (a) a formal description and (b) some extended functions for the problem solving part. We evaluate the proposed architecture functionally and practically, the latter by design and realization of a proof-of-concept.

The paper starts with providing some background about DQ management and the related work in Section 2. Subsequently the motivations for and principles of our problem solving architecture are presented in Section 3 formally. The proposed architecture is validated by a proof-of-concept, as described in Section 4, where also some performance aspects are evaluated. Our conclusions are drawn and the future research is sketched in Section 5.

2 BACKGROUND

This section gives some background information on the functional components of DQ management, outlines the motivations of the work, and provides an overview of the related work. For an overview of DQ management methodologies the interested reader is referred to (Batini et al., 2009).

2.1 Data Quality Management

DQ can be characterized by DQ attributes, which correspond to DQ issues in our notation mentioned above. DQ attributes are defined as those properties that are related to the state of DQ (Wand and Wang, 1996). DQ Management (DQM) is concerned with a number of business processes that ensure the integrity of an organization's data during its collection, aggregation, application, warehousing and analysis (AHIMA, 2012). As mentioned in (Knowledgent, 2015): "DQM is the management of people, processes, technology, and data within an enterprise, with the objective of improving the measures of Data Quality most important to the organization. The ultimate goal of DQM is not to improve Data Quality for the sake of having high-quality data, but to achieve the desired business outcomes that rely upon high-quality data." DQM can be decomposed into DQ assessment and DQ improvement functional components, as described below.

2.1.1 DQ Assessment

This component deals with determining which DQ

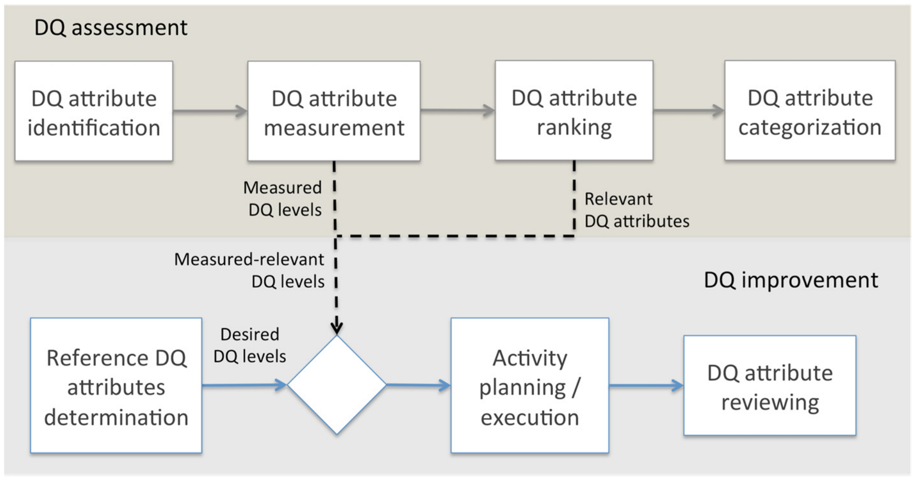

Figure 1: Functional components of DQ management.

attributes are relevant and the degree of theirrelevancy for an organization. As shown in Figure 1 (i.e., the top half) DQ assessment encompasses identification, measurement, ranking, and categorization of the DQ attributes that are relevant for an organization's data, see (Wang and Strong, 1996) or (Price and Shanks, 2004), where the latter reference provides a systematic approach to define DQ attributes. 'DQ attribute identification' is concerned with collecting possible DQ attributes from various sources like literature, data experts and data analysts. 'DQ measurement' and 'DQ attribute ranking' cover those processes that are for measuring and rating the importance of the identified attributes for the organization. 'DQ attribute categorization' deals with structuring the ranked attributes into a hierarchical representation so that the needs and requirements of the stakeholders like data managers, data experts, data analysts, and data consumers can be satisfied (Wang and Strong, 1996).

2.1.2 DQ Improvement

This component deals with continuously examining the data processing in an organization and enriching its DQ, given the relevant DQ attributes obtained from the DQ assessment. As shown Figure 1 (i.e., the bottom half), the functional components of DQ improvement include 'reference DQ attribute determination', 'activity planning and execution', and 'DQ attribute reviewing' (partly adopted from (Woodall et al., 2013)). 'Reference DQ attribute determination' identifies the organization's requirements related to the related DQ attributes, i.e., the desired DQ levels. 'Activity planning and execution' plans and carries out the activities required for improving the relevant DQ attributes to the desired level through, for example, executing a 'data cleansing' activity. Subsequently, one should also do 'DQ attribute reviewing' to validate these activities based on their dependency and measure the improved DQ attribute levels. The latter aspect of measurement can be seen as part of DQ assessment, see also (Woodall et al., 2013).

2.2 Motivation

There are software products called Issue Tracking Systems (ITSs) to manage and maintain the lists of issues relevant for an organization; issues like software bugs, customer issues, and assets. Also in our organization, i.e., the Research and Documentation Centre (abbreviated as WODC in Dutch) of the Dutch Ministry of Security and Justice, we use such an ITS to keep track of the existing DQ related problems. The WODC systematically collects, stores and enhances the Dutch judicial information directly or via its partner organizations (Bargh et al., 2015a). Considering the diversity and distribution of our data sources, we often receive the corresponding datasets containing inconsistent, imprecise, uncertain, missing, incomplete, etc. data records and attributes. Our objective for registering DQ related problems is to keep track of how and whether (other) data analysts resolve these problems based on their severity and urgency.

Data analysts write down an encountered problem P_n by a number of parameters denoted by $P_n(X_n, DS_n, MS_n, PU_n)$; $n: 1 \dots N$. Here X_n is a text describing the problem, DS_n is the desired problem severity level, MS_n is momentary problem severity level, and PU_n represents problem urgency. The momentary problem severity level MS_n can be

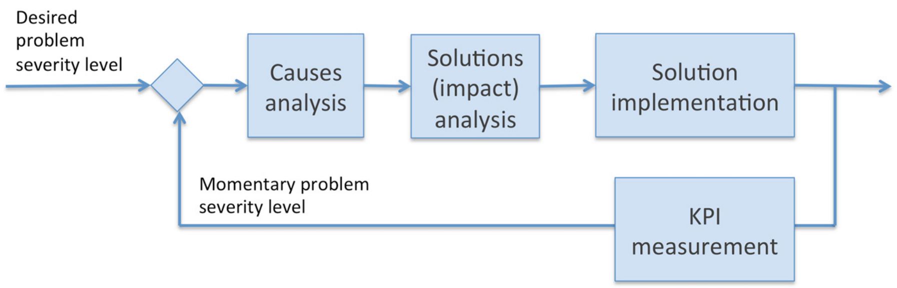
Desired problem severity level
Causes analysis
Solutions (impact) analysis
Solution implementation
Momentary problem severity level
KPI measurement

Figure 2: A framework for resolving the DQ related problems registered at the ITS.

determined subjectively as perceived by the data analyst or objectively as measured based on some data specific parameters, by using for example the approach proposed in (Jiang et al., 2009). The data analyst determines the desired problem severity level DS_n subjectively. Both DS_n and MS_n are expressed in a real number between 0 and 1, where 1 means the problem severity is the highest. We assume that $0 \leq DS_n \leq MS_n \leq 1$ and the problem is resolved when $MS_n = DS_n$ or $MS_n = 0$, which in this case the problem can be removed from the ITS. Problems can have various impacts comparatively. Therefore the weigh factor PU_n – a real value between 0 and 1 where 1 means the highest urgency – is inserted by data analysis subjectively. Variable PU_n conveys the level of the problem's urgency compared with other reported problems. Let's denote the set of problems registered at the ITS by:

$$\{P_n(X_n, DS_n, MS_n, PU_n) \mid 0 \leq DS_n \leq MS_n \leq 1\} \quad (1)$$

where n: 1 … N.

Figure 2 shows the functional components of a typical problem resolving system, status of which can be maintained in an ITS. Technical staffs, normally data analysts themselves, analyse the causes of a problem and its possible solutions in order to choose a solution based on some trade-offs. Before, during and after the realization of a solution some Key Performance Indicators (KPIs) are used to measure the momentary problem severity levels so that the impact of devised solutions can be determined via the feedback loop. Although registered problems are related to DQ attributes, the textual definitions of problems are not specified in terms of DQ attributes due to lack of knowledge or interest about DQ concepts by data analysts.

2.3 Related Work

As mentioned in Subsection 2.2, ITSs are widely used for tracking and managing various issues relevant for an organization. The tracked issues range from software bugs in software development houses like Bugzilla (Bugzilla Website, 2015) and JIRA (JIRA Software Website, 2015), customer issues in customer support call-centres/helpdesks like H2desk (H2desk Website, 2015), and assets in asset management companies like TOPdesk (TOPdesk Website, 2015). Software developers, customers, and employees of organizations use ITSs to report on the issues they face. These issues are reported in terms of the (detailed) description of the problem being experienced, urgency values (i.e., the overall importance of issues), who is experiencing the problem (e.g., external or internal customers), date of submission, attempted solutions or workarounds, a history of relevant changes, etc. Sometimes an issue report is called ticket due to being a running report on a particular problem, its status, and other relevant data with a unique reference number (as ITSs were originated as small cards within a traditional wall mounted work planning). Based on these reports, organizations take appropriate actions to resolve the corresponding problems. While there are many applications of ITSs for collaborative software development, including also management of announcements, documentation and project website, there are no applications of such systems for DQ management as we present in this contribution.

A possible feature that can be registered in ITSs is a user assigned label/tag in order to facilitate identifying and managing observed issues. In (Canovas Izquierdo et al., 2015), for example, a visualization tool is devised for facilitating the analysis and categorization of issues in open source software development projects, based on such registered labels. Labelling, when it is done appropriately, can reduce the semantic space of registered issues and facilitate mapping these issues to DQ attributes. This means that labels and tags can be used complementary to our approach for an improved mapping of DQ problems to DQ issues.

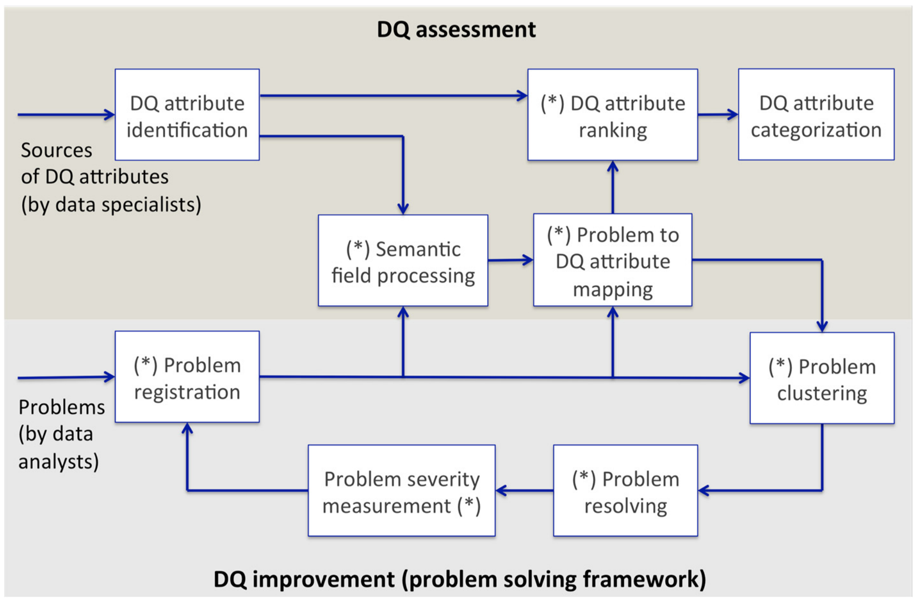
DQ assessment
DQ attribute identification
(*) DQ attribute ranking
DQ attribute categorization
Sources of DQ attributes (by data specialists)
(*) Semantic field processing
(*) Problem to DQ attribute mapping
(*) Problem registration
(*) Problem clustering
Problems (by data analysts)
Problem severity measurement (*)
(*) Problem resolving
DQ improvement (problem solving framework)

Figure 3: Functional architecture of the proposed system for resolving DQ related problems based on DQ management.

DQ management approaches proposed in literature, on the other hand, often rely on offline estimation of DQ issues and/or offline inquiries of DQ requirements. Wang and Strong (1996) propose a two-stage survey and a two-phase sorting method for identifying, ranking, and categorizing of DQ attributes in a given context. The authors developed a survey to produce a list of potential DQ attributes by a group of the participants of a workshop. Using another survey, the authors asked another group of the participants to rate the potential DQ attributes. In most organizations (including ours) gathering such a number of participants, i.e., data analysts, for surveying and sorting of DQ attributes is almost impossible due to being time consuming or having too few participants to produce valid results.

Woodall et al. (2013) propose a so-called hybrid approach for DQ management. For a set of relevant DQ attributes, the approach assesses the required level of DQ improvement by comparing the current state to a reference state. The DQ management and improvement according to the hybrid approach remains very abstract because DQ diagnostics are based on some high level strategic concepts. Similarly to the hybrid approach, our DQ management is intertwined with operational level practices of data analysts who observe and resolve (DQ related) problems. Establishing this link in our proposal, however, delivers a pragmatically dynamic DQ management, which is not the case in the hybrid approach.

All researches related to DQ assessment depend on some DQ objectives, based on which a set of relevant DQ attributes are sought. For example, the Environmental Protection Agency (EPA) approach (EPA, 2006) relies on, among others, a review of DQ objectives, a preliminary review of potential anomalies in datasets, and a statistical method to draw quantitative DQ related conclusions from the data. Our study uses the idea of translating DQ problems into the DQ issues and objectives, but by considering ‘all reported’ problems in the datasets and not just a few reported anomalies as (EPA, 2006) does. Moreover, unlike (EPA, 2006) we don’t rely on statistical methods exclusively and incorporate also the domain knowledge of data analysts. Pipino et al. (2012) use the EPA methods and additionally incorporate a subjective DQ assessment. To this end, the authors use a questionnaire to measure the perceptions of the stakeholders (e.g., database administrators) on DQ attributes. Subsequently, the approach of (Pipino et al., 2012) determines the root causes of data discrepancies and tries to improve DQ by solving these discrepancies. Also our proposal combines both subjective and objective perceptions of the stakeholders on DQ related problems, but we combine these perceptions at an operational level by using a problem solving system, and not on a DQ attribute or strategic level as (Pipino et al., 2012) does. Eppler and Witting (Eppler and Wittig, 2000) use the EPA methods and adds some extra attributes to evaluate how pragmatic every DQ attribute can be realised. Unlike (Eppler and Wittig, 2000) we do not use any additional attribute to determine how pragmatic the DQ attributes are.

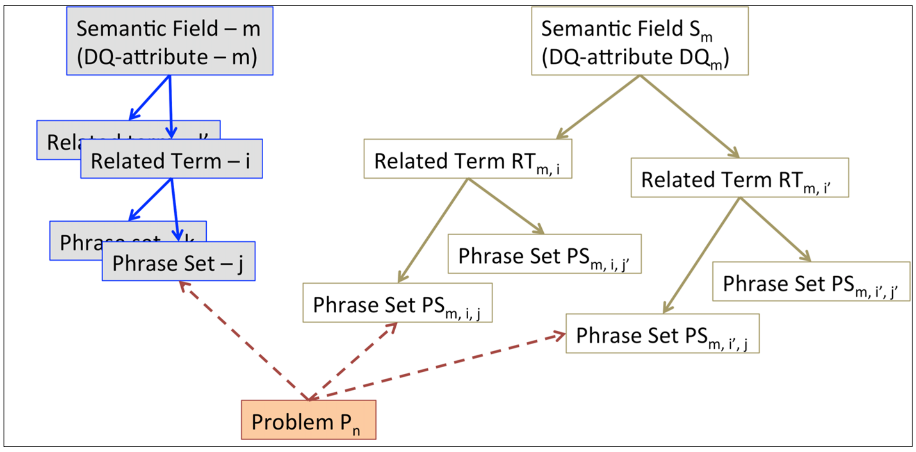

Figure 4: An illustration of the hierarchical structure of semantic fields, related terms and phrase sets; and their relation to problems (the texts in grey blocks are intentionally abbreviated).

3 PROPOSED APPROACH

Figure 3 shows our proposed system architecture, whose key functional building blocks, those marked with a *, are described in the following formally.

3.1 Data Quality Assessment

DQ assessment starts with a literature study by data specialists to enlist potential DQ attributes and ends up with categorizing the selected and ranked DQ attributes. The ranking of DQ attributes, which we innovatively base on the set of problems registered in the ITS, will be described in the following.

3.1.1 Semantic Field Processing

A semantic-field is a set of conceptually related terms (Kornai, 2010). Every semantic-field, which corresponds to only one DQ attribute in our setting, comprises a number of 'related terms'. Every related term, in turn, corresponds to a number of 'phrase sets'. Every phrase set comprises a number of phrases that appear in problem descriptions. The set of semantic-fields, related terms and phrase sets are summarized in a so-called 'Semantic-Field Processing Table (SFPT)'. Formally, every DQ attribute DQ_m (where $m: 1 \dots M$) can be described by a distinct semantic field S_m that consists of some semantic field attributes called related terms $RT_{m,i}$. In other words,

$$DQ_m \equiv S_m = \{RT_{m,i} | i: 1 \dots M_m\}, \qquad (2)$$

where $m: 1 \dots M$. In turn, every related term $RT_{m,i}$ can be described by some phrase sets $PS_{m,i,j}$ as

$$RT_{m,i} = \{PS_{m,i,j} | j: 1 \dots M_{m,i}\}, \qquad (3)$$

where $m: 1 \dots M;\ i: 1 \dots M_m$. Every phrase set $PS_{m,i,j}$ comprises some set members / short phrases $PH_{m,i,j,k}$ as

$$PS_{m,i,j} = \{PH_{m,i,j,k} | k: 1 \dots M_{m,i,j}\}. \qquad (4)$$

Domain experts define these semantic-fields, related terms, phrase sets, and short phrases in a way that the short phrases can be found in problem descriptions of data analysts; any related term can be related to only one semantic-field / DQ attribute; and any phrase set can be related to only one related term. Thus, as illustrated in Figure 4, we assume that there is a tree structure among 'semantic fields', 'related terms', and 'phrase sets'. Due to the tree structure depicted above, there are no related terms that are common among semantic-fields / DQ attributes, and there are no phrase sets that are common among related terms.

$$RT_{m,i} \neq RT_{m',i'} \qquad \forall\, m \neq m' \text{ or } i \neq i' \qquad (5)$$
$$PS_{m,i,j} \neq PS_{m',i',j'} \quad \forall\, m \neq m' \text{ or } i \neq i' \text{ or } j \neq j'.$$

Note that short phrases in phrase sets may appear in multiple phrase sets.

3.1.2 Problem to DQ Attribute Mapping

When a problem description contains all short phrases of a phrase set, one can map the problem to the corresponding related term and, in turn, to the corresponding DQ attribute uniquely. Based on condition (5), phrase sets are unequal (see also the illustration in Figure 4). This property and the hierarchical relation among phrase sets, related terms and semantic fields guarantee that every phrase set can identify only one related term, thus one semantic field / DQ attribute. As a problem description X_n may include more than one phrase sets, however, the corresponding problem P_n can be associated with more than one related term and thus to more than one DQ attribute.

Assume that the semantic fields identified for problem P_n are denoted by set

$$S(P_n) \subseteq \{S_1, S_2, \cdots, S_M\}; \;\; n: 1 \dots N. \qquad (6)$$

Then, problem P_n can be mapped to DQ attributes DQ_m if $S_m \in S(P_n)$, where $m: 1 \dots M$. For problems P_n and DQ attributes DQ_m where $n: 1 \dots N$ and $m: 1 \dots M$, one can define the problem to DQ attribute mapping in terms of a *association matrix* as

$$A = \left[a_{n,m}\right]_{N\times M}$$

$$\text{where } a_{n,m} = \begin{cases} 1 & \text{if } S_m \in S(P_n), \\ 0 & \text{otherwise.} \end{cases} \qquad (7)$$

Note that if $a_{n,m} = 0$ for all $m: 1 \dots M$, i.e., when $S(P_n) = \emptyset$, then problems P_n cannot be mapped to any DQ attribute. In this case we say that the mapping for this problem has resulted in a miss. The number of such miss outputs should be zero ideally.

For improving DQ attributes, as we will see in the following sections, we need to take into account the momentary and desired severity levels of problems, i.e., the DS_n and MS_n parameters of problem P_n registered in the ITS. Therefore, we define the *weighed association matrix* as

$$A_w = \left[aw_{n,m}\right]_{N\times M}$$

$$\text{where } aw_{n,m} = a_{n,m} \cdot (MS_n - DS_n). \qquad (8)$$

The problems registered in the ITS, furthermore, can have various urgency and importance levels, denoted by weight PU_n for problem P_n with a real value between 0 and 1 (remember that low or zero urgency issues are minor and should be resolved as time permits). Such a factor can be applied to Relation (8) by replacing $MS_n - DS_n$ with $PE_n.(MS_n - DS_n)$ to obtain the *extended weighed association matrix* as

$$A_{ew} = \left[aew_{n,m}\right]_{N\times M}$$

$$\text{where } aew_{n,m} = a_{n,m} \cdot PU_n \cdot (MS_n - DS_n). \qquad (9)$$

3.1.3 DQ Attribute Ranking

This functionality determines the priority values of DQ attributes based on the (extended weighted) association matrix, which is in turn derived from the problem descriptions, problem desired and actual severity levels, and/or problem urgencies. Given the (extended) weighted association matrix in Relation (8) or (9), the *dynamic DQ rank* of attribute DQ_m for $m: 1 \dots M$ is defined as:

$$R_m^d = \frac{\sum_{n=1}^{N} aw_{n,m}}{\sum_{n=1}^{N}\sum_{m=1}^{M} aw_{n,m}} \text{ or } \frac{\sum_{n=1}^{N} aew_{n,m}}{\sum_{n=1}^{N}\sum_{m=1}^{M} aew_{n,m}}. \qquad (10)$$

As the elements of the (extended) weighted association matrix (i.e., $aw_{n,m}$ or $aew_{n,m}$) are dependent of the momentary problem severity level MS_n, which changes as problems are resolved by data analysts, the DQ rank in Relation (10) is a dynamic value depending on the problem resolving process. As a special case of DQ ranking in relation (10), we define the *static DQ rank* based on the association matrix in (7) for $m: 1 \dots M$ by:

$$R_m^s = \frac{\sum_{n=1}^{N} a_{n,m}}{\sum_{n=1}^{N}\sum_{m=1}^{M} a_{n,m}}. \qquad (11)$$

The static DQ rank defined in relation (11) is just dependent of having a problem in the ITS or not. The underlying assumption is that a problem is removed from the ITS as soon as it is resolved. This static DQ rank is called static because it does not change as the resolving of a problem progresses unless it is removed from the ITS.

3.2 Data Quality Improvement

Our DQ improvement largely corresponds to the problem-resolving system, as shown in Figure 3. By solving the registered problems, data analysts also improve the corresponding DQ attributes and therefore carry out DQ management. DQ improvement comprises a number of functions, as shown in Figure 3, which are elaborated upon in the following.

3.2.1 Problem Clustering

Registered problems can be clustered according to some criteria in order to reuse those solutions that address similar problems and, consequently, to yield efficiency and optimization. Our proposal for problem clustering is to use the associations among problems and DQ attributes because the resulting clusters can benefit from those DQ specific knowledge and solutions proposed in the literature. As defined in Relations (7-9), the problem to DQ attribute mapping results in some (weighed) association values between pairs of (problem P_n, DQ attribute DQ_m) as follows:

$$(P_n, DQ_m) = \qquad (12)$$

$$\begin{cases} a_{n,m} & \text{see (7)} \\ aw_{n,m} = a_{n,m} \cdot (MS_n - DS_n) & \text{see (8)} \\ aew_{n,m} = a_{n,m} \cdot PE_n \cdot (MS_n - DS_n) & \text{see (10).} \end{cases}$$

We specify every problem P_n by the vector $\big((P_n, DQ_1), (P_n, DQ_2), \cdots, (P_n, DQ_M)\big)$ in M dimensional DQ attribute space, where its elements are defined in Relation (12) for $m{:}\,1 \ldots M$. We call these vectors as 'association vector', 'weighed association vector', or 'extended weighed association vector' of problem P_n, respectively.

The ((extended) weighed) association vectors are fed as inputs to the component 'problem clustering' as shown in Figure 3. In order to find similarity between problems one can calculate the distance between every pair of such vectors, using for example the hamming distance or Euclidian distance. The pairwise distances can be used to cluster the corresponding problems. The resulting clusters encompass those problems that share similar behaviours in terms of DQ attributes. In order to address registered problems one can prioritize problem clusters, for example based on their sizes and weighs, and apply (and/or develop new) solutions that address these problem clusters according to the priority of the problem clusters.

Alternatively, one can *classify* problems in terms of existing solutions, instead of clustering them based on some behavioural similarity in the DQ attribute spaces. For example, assume a software tool resolves/addresses a specific subset of DQ attributes. Availability of such tools that are specific to a subset of DQ attributes inspires us to consider classifying the registered problems in terms of the DQ attributes that are addressed by some powerful software tools. In the following, we propose a method for choosing appropriate solutions, which resembles such a classification case.

3.2.2 Problem Resolving

Resolving of problems requires applying solutions, each of which encompasses a number of activities. Previously we specified problems in the DQ attribute space, i.e., by mapping problems to DQ attributes using the ((extended) weighed) association vectors and Relation (12). On the other hand, most solutions – including software tools and DQ improvement processes – can be characterized in terms of those DQ attribute issues that they address/resolve. Therefore, we propose to specify such solutions based on the DQ attributes that they address. To this end, assume every solution S_k is represented by a *solution association vector* $S_k = (s_{k,1}, \cdots, s_{k,m}, \cdots, s_{k,M})$ where for $m{:}\,1 \ldots, \mathrm{M}$ we have

$$s_{k,m} = \begin{cases} 1 & \text{if } S_k \text{ addresses DQ attribute } DQ_m \\ 0 & \text{otherwise.} \end{cases} \quad (13)$$

Here we assume solution S_k either addresses DQ attribute DQ_m or not, i.e., $s_{k,m}$ takes a binary value. One can alternatively assume a real value for parameter $s_{k,m}$ in interval $0 \le s_{k,m} \le 1$, denoting the fraction that solution S_k can (potentially) resolve the DQ attribute issue DQ_m in the organization. Hereto, for example, the approach of (Jiang et al., 2009) can be used. Considering the dynamic or static rank of every DQ attribute, see Relations (10) and (11) respectively, one can define the normalized benefit of solution S_k for the organization as:

$$BF_k = \frac{1}{M}\begin{cases} S_k \cdot R^d = \sum_{m=1}^{M} s_{k,m} \cdot R_m^d & \text{dynamic} \\ S_k \cdot R^s = \sum_{m=1}^{M} s_{k,m} \cdot R_m^s & \text{static,} \end{cases} \quad (14)$$

where upper scripts d and s demote dynamic and static DQ management, respectively.

On the other hand, one must balance the benefits of a solution, as characterized in Relation (14), against its costs. Various solutions inflict various costs on an organization. Let weight SC_k denote the normalised cost of solution S_k for the organization, by normalised we mean taking a real value between 0 and 1, where low or zero values represent those low or zero cost solutions. The cost benefit value of a solution can be defined as

$$CB_k = SC_k - BF_k \qquad \text{for } k{:}\,1 \ldots \mathrm{K}. \quad (15)$$

Ideally one should prioritize solutions based on Relation (15) and apply those solutions that yield the lowest cost benefit values as defined in Relation (15).

3.2.3 Problem Severity Measurement

KPIs can be defined and used to measure the momentary severity of problems. As shown in Figure 3, this functional block closes the loop of our current problem-resolving system and provides a feedback about the momentary status of registered problems, i.e., enables our dynamic DQ management.

In order to create objective KPIs we observe that often in practice DQ related problems are detected because some phenomena, for example the number of crimes committed per a time interval, are quantified differently from two (or more) data sources. Assume $X_t = \cdots, x_{t-1}, x_t, x_{t+1}, \cdots$ and $Y_t = \cdots, y_{t-1}, y_t, y_{t+1}, \cdots$ are time series that denote the measures of the same phenomenon using two different sources/datasets at consequent time intervals (yearly, monthly, daily etc.). Ideally, $x_t = y_t$ for all t, but due to DQ issues the data analyst observe discrepancies between these readings and reports the problem in the ITS. The difference time series $Z_t = X_t - Y_t = \cdots, x_{t-1} - y_{t-1}, x_t - y_t, x_{t+1} - y_{t+1}, \cdots$ can be a KPI in time intervals, as shown in Figure 5. For our DQ management one can normalize the difference time series to derive problem severity level at a given moment t by

$$z_{t,\text{ norm}} = \frac{|x_t - y_t|}{\max(x_t, y_t)}, \ \max(x_t,\ y_t) > 0. \quad (16)$$

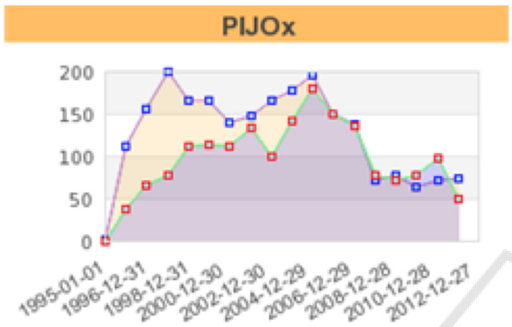

Figure 5: Visualizations of two time series.

Sometimes it is more realistic to base problem severity level on the last l differences observed, i.e., on a history of measurements. Therefore, a smoothed problem severity level at a given moment t can be defined by

$$\bar{z}_{t,\text{ norm}} = \frac{\sum_{i=t-l+1}^{t} |x_i - y_i|}{\sum_{i=t-l+1}^{t} (max(x_i, y_i) - t_h)}, \quad (17)$$

where t_h is an appropriate threshold value – for example, it can be set as the possible minimum value for amount $max(x_i,\ y_i)$ over i (for example, when counting objects, this could be zero; for financial variables, the minimum could be negative).

The momentary or smoothed problem severity levels defined in Relations (16) and (17) can be visualized by a Gauge or Dial chart as shown in Figure 6. Subjective measurements, where data analysts assign a problem severity level according to their insight at a given moment, can be another method for determining KPIs. Such a subjective measurement can be useful when, for example, combining multiple and heterogeneous measures as defined in Relations (16) and (17).

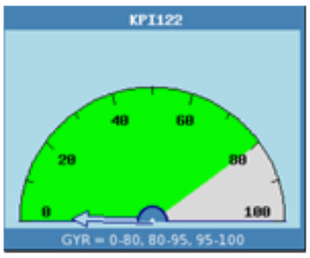

Figure 6: Visualizations of the resulting-ratio dashboard.

4 PROOF OF CONCEPT

In this section we describe a proof of concept prototype for the proposed DQ management that is realized in our organization. Moreover, we shall elaborate on performance evaluation of its problem to DQ attribute mapping.

4.1 Implementation

Our realization of the proposed architecture includes problem registration, semantic field processing, problem to DQ attribute mapping, DQ attribute ranking, problem clustering, problem resolving, and problem severity measuring.

We used the Team Development environment of Oracle APEX as our ITS to enable data analysts to register the arising DQ related problems. The data log is stored in an Oracle DBMS (Database Management System). Currently, there are 334 problems registered together with their desired and momentary problem severity levels.

In order to determine the 'semantic-field processing table' for the registered problems, we use a heuristic as described below. Given a DQ attribute, the current implementation carries out two steps of (a) determining a list of the related terms for the semantic-field corresponding to the DQ attribute, and (b) syntactical decomposing of every related term to some phrases of smaller sizes that appear in problem descriptions. We assume that every phrase set $PS_{m,i,j}$ comprises at most two short phrases, i.e., $M_{m,i,j} \le 2$ in Relation (4). Therefore, we shall sometimes use the term 'phrase pair' instead of 'phrase set'.

Assume that we have some potential DQ attributes derived from literature and that we have the actual problems descriptions registered in the ITS. In the first step of the heuristic we analyze every pair of (problem description, potential DQ attribute). When a problem description is conceptually related to a DQ attribute, then the conceptual formulation of the problem description is recorded as a related term. This related term has a smaller size than the corresponding problem description size. Iteration of this step results in two columns of the 'related terms' and the corresponding 'DQ attributes' in a semantic-field processing table. Lines (5) and (7) in the pseudo code below refer to this process. In the second step, every related term is decomposed into sets of smaller phrases that syntactically appear in problem descriptions. This results in another column 'phrase pair' in the semantic-field processing table. Lines (6) and (7) in the pseudo code below refer to this process.

```
⊳ SFP is set of rows of the semantic-field processing table
⊳ rt is a related term
(1) SFP ← ∅
(2) for each problem description x do
(3)    for each potential DQ attribute dq do
(4)       if x refers to dq then
(5)       define rt as a conceptual
                 formulation of dq
(6)    decompose x into (p1, p2)
(7)    if (p1, p2, rt, dq) ∉ SFP then
       SFP ← SFP ∪ (p1, p2, rt, dq)
```

Note that here some problems cannot be readily mapped to a DQ attribute. Moreover, the related terms obtained from the first stage are natural language terms. The syntactical decomposition of such natural language terms into phrase pairs can have more than one parsing tree (Mooney, 2007). For example, related term 'missing data' can be decomposed to phrase pairs {Is, Missed}, {Are, Missed}, {Is, Missing} or {Are, Missing}.

Due to a prototype character of the current implementation, the clustering of problems and resolving problems according to their impacts / costs are currently based on a manual process. The measuring of the momentary severity level of problems is based on the described KPIs. The KPIs of complementary measurements, as defined in Relations (16) and (17), are defined in SQL terms and visualized by a dynamic PHP website. Currently, the ITS is deployed in another server and it is loosely coupled to the other components (as problem logs are downloaded as files). This slows down the communication between these two systems. In the future we intend to mitigate the communication speed of the current implementation.

4.2 Evaluation

Generic DQ management functionalities, which are identified in (Woodall et al., 2013), are also represented in the proposed DQ management in this contribution. The proof of concept system has been realized, deployed, and used in our organization since early 2014. All functionalities of the realized system work as described in this contribution.

For performance evaluation here we report on the performance of our heuristic for the problem to DQ attribute mapping as the key system component in our problem solving system. Our heuristic cannot target all problems in the ITS because we start with DQ attributes and look at the problem descriptions in the ITS to identify the semantic-field of every DQ attribute (i.e., the related terms). Based on related terms our proof of concept seeks out the phrase pairs in a problem statement. As a result, this process may overlook some problems if for them no related term can be identified, thus failing to map such problems to DQ attributes. This overlooking could be due to not exhaustively searching the space of registered problems and DQ attributes or not describing problems expressively. Our search of related terms and phrases stops at a certain point due to practical reasons, for example, after finding a certain number of phrase-pairs.

Those problems that are (not) mapped to DQ attributes are called (un)targeted problems. In order to reduce the number of untargeted problems we iterated the heuristic described above to come up with the (new) related terms corresponding to some (potential) DQ attributes. These iterations reduced the number of untargeted problems sharply, as shown in Figure 7. After a certain number of iterations, however, the number of untargeted problems did not decrease much. We suspect this is because the descriptions of the remaining problems are poorly written, which makes it difficult to associate them with any related term based on the syntax of these problem descriptions.

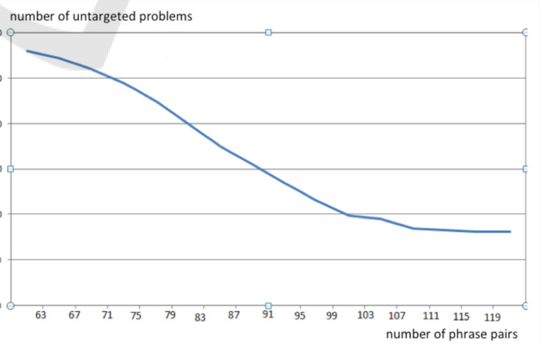

Figure 7: Number of untargeted problems (vertical) in terms of the number of related terms.

4.3 Discussion and Limitations

In this contribution we proposed to measure the severity level of the reported problems and map them to the corresponding DQ attribute levels. A way to measure the severity level of registered problems is to measure KPIs, which faces some challenges like defining effective, valid, and standardized performance indicators. For instance, a KPI based on measuring the hamming distance of 2 words can be ineffective. For instance, the words “Netherlands” and “Holland” are semantically closer than their Hamming distances when considering the cultural background of both words. Measuring semantic distances, on the other hand, is more challenging than measuring hamming distances.

An underlying assumption in our proposal is that data analysts of an organization register encountered problems in an ITS. In practice, users are not eager to register problems effectively and expressively. Organizations should encourage and train their employees to fill in such logging system so that the benefits of the proposed system can be harvested. Using tags and labels to mark DQ problems, see (Canovas Izquierdo, et al., 2015), can further be explored to this end.

We proposed a data quality management approach to utilize user-generated inputs about DQ problems to carry out DQ management. For each functional component, furthermore, we proposed some simple (and heuristic) methods to realize the component’s functionality. Due to modular property of the proposed DQ management approach, one can replace these methods by defining customized methods suitable for own organization and problem domain.

5 CONCLUSIONS

In this contribution we presented the formal description and the system architecture of an integrated system for resolving the problems observed in datasets based on DQ management. The proposed architecture, moreover, results in a dynamic DQ management system, which relies on user generated data (i.e., data users/analysts who describe the DQ related problems they encounter in their daily practice). By managing DQ related problems encountered in an organization at an operational level, our proposal manages also the organization’s DQ issues (i.e., realizes DQ management). To this end, we semantically and dynamically map the descriptions of DQ related problems to DQ attributes. The mapping provides a quantitative and dynamic means to determine the relevant DQ attributes and the level of their relevancy, given the operational setting (i.e., the desired and momentary problem severity levels).

The realization of the proposed DQ management in our organization has given us insightful feedback on its advantages and limitations. As we envisioned, the solution bridged successfully the gap between the operational level (e.g., data analysts) and strategic level (e.g., managers) DQ stakeholders within our organization. To fully benefit from the potentials of the proposed architecture, however, it is necessary to encourage the users of datasets (i.e., data analysts) to provide their inputs about the DQ related problems that they encounter proactively and expressively. Through improving the problem registration process one can reduce the number of untargeted problems and guarantee their influence on dataset problem resolution and DQ management processes. It is for our future research to explore, for example, user awareness and training solutions, and to develop objective KPIs and problem resolving techniques (e.g., to determine the capabilities and costs of candidate solutions).

REFERENCES

AHIMA, 2012. Data Quality Management Model (Updated). *In Journal of American Health Information Management Association: AHIMA*. Vol. 83, No.7, 62-67.

Bargh, M. S., Choenni, S., Meijer, R., 2015a. Privacy and Information Sharing in a Judicial Setting: A Wicked Problem. In Proceedings of DG.O, 97-106, ACM.

Bargh, M. S., van Dijk, J., Choenni, S., 2015b. Dynamic data quality management using issue tracking systems. *In the IADIS International Journal on Computer Science and Information Systems (IJCSIS, ISSN: 1646-3692)*, ed. P. Isaias and M. Paprzycki, Vol. 10, No. 2, pp. 32-51.

Bargh, M. S., Mbgong, F., Dijk, J. van, Choenni, S., 2015c. A framework for Dynamic Data Quality Management. In *Proceedings of ISPCM,* Las Palmas, de Gran Canaria, Spain.

Batini C, Cappiello C, Francalanci C, Maurino A., 2009. Methodologies for Data Quality Assessment and Improvement. *ACM Computing Surveys*, Vol. 41, No. 3, Article 16, ACM.

Birman, K. P., 2012. Consistency in Distributed Systems. Book Chapter in *Guide to Reliable Distributed Systems*, 457-470.

Bugzilla Website, 2015. https://www.bugzilla.org (retrieved on 31/10/2015).

Choenni, S., Leertouwer, E., 2010. Public Safety Mashups to Support Policy Makers. *In Electronic Government and the Information Systems Perspective (EGOVIS)*, Bilbao, 234-248, Springer.

Canovas Izquierdo, J. L., Cosentino, V., Rolandi, B., Bergel, A., Cabot, J., 2015. GiLA: GitHub Label Analyzer. *In IEEE 22nd International Conference on*

Software Analysis, Evolution and Reengineering (SANER), 479-483. Montreal, Canada.

Davenport, T. H., Glaser, J., 2002. Just-in-time delivery comes to knowledge management. *Harvard business review*, 80(7), 107-11.

Dijk, J. van, Choenni, R., Leertouwer, E., Spruit, Brinkkemper, S., 2013. A Data Space System for the Criminal Justice Chain. *In Proceedings of ODBASE*, Graz, Austria, Springer, 755-763.

EPA, 2006. Environmental Protection Agency. Data Quality Assessment: A Reviewer's Guide, Technical Report EPA/240/B-06/002, EPA QA/G-9R.

Eppler, M. J., Wittig, D., 2000. Conceptualizing Information Quality: A Review of Information Quality Frameworks from the Last Ten Years. In *Proceedings of the Conference on Info Quality,* 83-96.

H2desk Website, 2015. https://www.h2desk.com (retrieved on 31/10/2015).

Jiang, L., Barone, D., Borgida, A., Mylopoulos, J. 2009. Measuring and Comparing Effectiveness of Data Quality Techniques. van Eck, P., Gordijn, J., Wieringa, R. (Eds*.), International Conference on Advanced Information Systems Engineering (CAiSE)*, LNCS 5565,171–185, Springer-Verlag Berlin Heidelberg.

JIRA Software Website, 2015. https://www. atlassian.com/software/jira (retrieved on 31/10/2015).

Knowledgent 2015. White Paper Series: Building a Successful Data Quality Management Program, http://knowledgent.com/whitepaper/building-successful-data-quality-management-program/ (retrieved on 31/10/2015).

Kornai, A. 2010. The Algebra of Lexical Semantics. In: Mathematics of Language, 174-199, Springer.Mooney R. J., 2007. Learning for Semantic Parsing. In *Proceedings of Computational Linguistics and Intelligent Text Processing*, Mexico City (invited paper), A. Gelbukh (Ed.), 311-324, Springer.

Netten, N., van den Braak, S., Choenni, S., Leertouwer, E., 2014. Elapsed Times in Criminal Justice Systems. *In Proceedings of ICEGOV*, 99-108, ACM.

Pipino, L. L. et al., 2012. Data Quality Assessment. In: Communications of the ACM. Vol. 45, No. 4, 211-218.

Price, R., Shanks, G., 2004. A Semiotic Information Quality Framework. *In Proceedings of International Conference on Decision Support Systems (DSS),* 658-672.

TOPdesk Website, 2015. http://www.topdesk.nl (retrieved on 31/10/2015).

Wand, Y., Wang, R. Y., 1996. Anchoring Data Quality Dimensions in Ontological Foundations. In *Communications of the ACM*, Vol. 39, No. 11, 86-95.

Wang, R. Y., Strong, D. M. 1996. Beyond Accuracy: What Data Quality Means to Data Consumers. In: *Journal of Management Information Systems*. Vol. 12, No. 4, 5-33.

Woodall, P., Borek, A., Parlikad, A. K., 2013. Data Quality Assessment: The Hybrid Approach. *In Information & Management*, Vol. 50.