Maturing Pay-as-you-go Data Quality Management: Towards Decision Support for Paying the Larger Bills

Jan van Dijk[1(✉)], Mortaza S. Bargh[1], Sunil Choenni[1,2], and Marco Spruit[3]

[1] Research and Documentation Centre, Ministry of Security and Justice, The Hague, The Netherlands
{j.j.van.dijk, m.shoe.bargh, r.choenni}@minvenj.nl
[2] Research Centre Creating 010, Rotterdam University of Technology, Rotterdam, The Netherlands
r.choenni@hr.nl
[3] Department of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands
m.r.spruit@uu.nl

Abstract. Data quality management is a great challenge in today's world due to increasing proliferation of abundant and heterogeneous datasets. All organizations that realize and maintain data intensive advanced applications should deal with data quality related problems on a daily basis. In these organization data quality related problems are registered in natural languages and subsequently the organizations 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. Another challenge in data quality management is to choose appropriate solutions for addressing data quality problems due to lack of insight in the long-term or broader effects of candidate solutions. This difficulty becomes particularly prominent in flexible architectures where loosely linked data are integrated (e.g., data spaces or in open data settings). We present also a decision support framework for the solution choosing process to evaluate cost-benefit values of candidate solutions. The paper reports on a proof of concept tool of the proposed architecture and its evaluation.

Keywords: Data quality issues · Data quality management · Knowledge mapping · User generated inputs · Solution management

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

C. Francalanci and M. Helfert (Eds.): DATA 2016, CCIS 737, pp. 102–124, 2017.
DOI: 10.1007/978-3-319-62911-7_6

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 [1] and Elapsed Time Monitoring System of Criminal Cases [2]. 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 [3, 4]. 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 [5].

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 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 [6] 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. In [7–9] we presented a formal description and system architecture for an integrated system for resolving the problems observed in datasets based on DQ management principles. We evaluated the proposed architecture functionally and practically, the latter by design and realization of a proof-of-concept. The current work extends [9] with an additional framework for DQ solution management.

The paper starts with providing some background about DQ management and the related work in Sect. 2. Subsequently the motivations for and principles of our problem solving architecture are presented in Sect. 3 formally. The proposed architecture is validated by a proof-of-concept, as described in Sect. 4, where also some performance aspects are evaluated. Our conclusions are drawn and the future research is sketched in Sect. 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 [10].

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 [11]. 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 [12]. As mentioned in [13]: "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.

Flexible architectures and dynamic environments, e.g. data space architectures and open or linked data environments, are strongly user-oriented and are characterized by a pay-as-you-go data management approach [14]. Hence, DQ management in these situations is often performed from a local viewpoint. The "enterprise" as mentioned in [13] can in these architectures/environments be seen as a collaboration of organizations or data customers (e.g. data scientists), where the composition of the collaboration can vary, depending on the stakeholders of specific DQ related problems.

DQ Assessment. This component deals with determining which DQ attributes are relevant and the degree of their relevancy for an organization. As shown in Fig. 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 [15, 16], 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 [15].

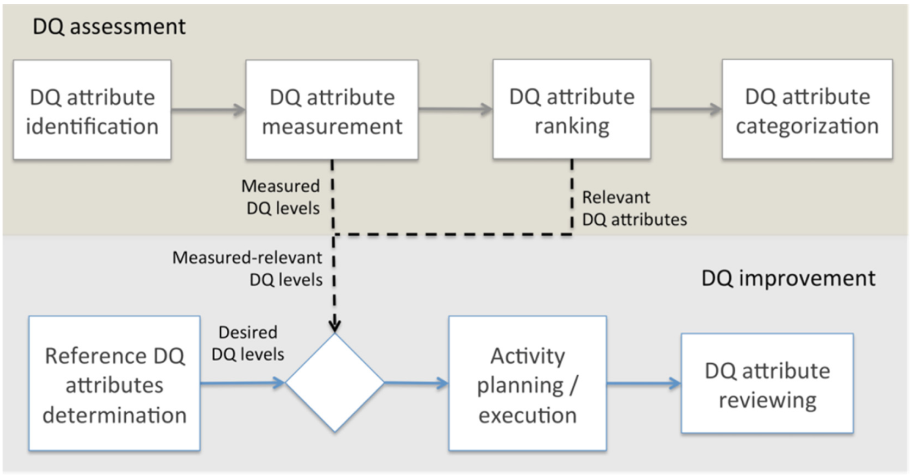

Fig. 1. Functional components of DQ management.

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 Fig. 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 [15]). '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 [15].

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 (Table 1). The WODC systematically collects, stores and enhances the Dutch judicial information directly or via its partner organizations [18]. 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.

Table 1. Seven typical problems registered in our ITS and their descriptions.

Problem	Description
1	The column with community codes is missing in the table
2	The columns with community codes are missing in all tables
3	The column with community codes must be added
4	The column with community codes cannot be found in the table
5	The column with community codes is not filled
6	The columns with community codes are not filled
7	The community codes have been deleted

Data analysts write down an encountered problem P_n by a number of parameters denoted by $P_n(X_n, DS_n, MS_n, PU_n)$; $n : 1...N$. 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 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 [19]. 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:

$$\{P_n(X_n, DS_n, MS_n, PU_n) | 0 \leq DS_n \leq MS_n \leq 1\} \text{ where } n: 1 \ldots N \tag{1}$$

Figure 2 shows the functional components of a typical problem resolving system, status of which can be maintained in an ITS. Technical staffs - data quality managers or data analysts themselves, analyze the causes of a problem and its candidate 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.

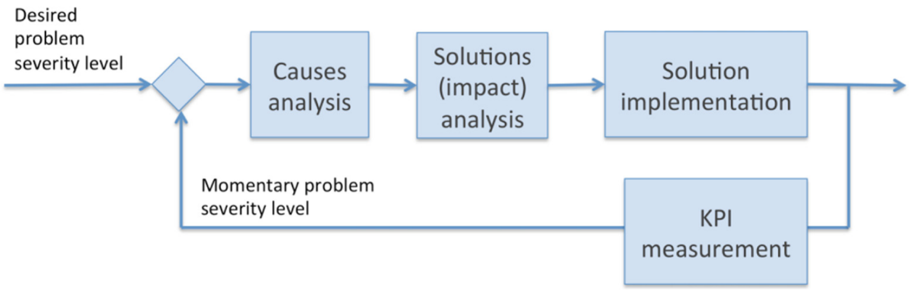
Desired problem severity level
Causes analysis
Solutions (impact) analysis
Solution implementation
Momentary problem severity level
KPI measurement

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

Furthermore, data analysts solve problems within certain boundaries (i.e. based on a certain priority or organizational limitations). Often, problems arise when data customers are working with the data. In these cases, a urgent solution is required for the dataset that is relevant for these data customers. Depending on how DQmanagement is organized, it is foreseeable that the budget for implementing the solution may come from the data customers that face the problem. This is especially the case in environments in which data management is not centrally organized, like data space architectures [3] and open data based applications/systems. We observe that, as a result of more user-oriented or pay-as-you-go data management, it becomes likely that problems are solved in a more local manner. Local KPIs ensure that the problem is solved in this particular context, but when the same problem occurs in a different context, it will be registered and handled as a new problem.

2.3 Related Work

As mentioned in Subsect. 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 [20] and JIRA [21], customer issues in customer support callcenters/helpdesks like H2desk [22], and assets in asset management companies like TOPdesk [23]. 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 [24], 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 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 [15] 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. [17] 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 [25] 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 [25] does. Moreover, unlike [25] we don't rely on statistical methods exclusively and incorporate also the domain knowledge of data analysts. Pipino et al. [26] 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 [26] 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 [26] does. Eppler and Witting [27] use the EPA methods and adds some extra attributes to evaluate how pragmatic every DQ attribute can be realized. Unlike [27] we do not use any additional attribute to determine how pragmatic the DQ attributes are.

Possible ways of resolving data quality related problems are bound by several aspects. In the process of defining the data quality objectives, developed by EPA [25], every step describes some sort of boundary. Which solution method is applied depends on the organizational scope, budget, planning, etc. For instance, when a project experiences some problems with the timeliness of a specific dataset – i.e. the project needs up-to-date data – and no other project in the organization has the need or resources to invest in this problem, the problem might be fixed within project scope. From a strict data quality management perspective, in which high data quality is achieved when the data fit its intended use [15], the problem is solved. But in long term other projects might experience the same problem, and then it is inefficient to continue fixing the problem within project scope. Data quality managers must have the knowledge that this problem occurred before and at strategic level managers have to decide if this problem has to be dealt with in a more centralized way. This exceeds the scope of a single data quality problem or cluster of current problems, because it also involves those already solved problems. Lee et al. [28] mentioned in 2003 already that it is essential for improving data quality problem resolving to register how problems in the past were solved. However, to determine which problems are most urgent and which solutions are most appropriate, a lot of knowledge and expertise are required. The more this knowledge is put into operational use, the more the maturity of data quality management in an organization grows [29]. Mostly, this is done by implementing a data quality division or team [28] that can combine technical as well as organizational insights into solving data quality related problems. For instance, such a team shall remember those imperfect solutions that lead to recurring similar problems later on, so eventually a more sustainable solution can be sought. We extend our DQ management architecture in [9] with a knowledge-based framework that helps evaluating chosen solution methods and eventually assisting the choice of the (best) solution method for new problems. This way, also more flexible architectures like those for data spaces and open data communities can mature in their DQ management.

3 Proposed Approach

Figure 3 shows our proposed system architecture for resolving data quality related problems, which is described formally in [9]. We describe the key functional building blocks of this architecture, those marked with a *, in the following subsections.

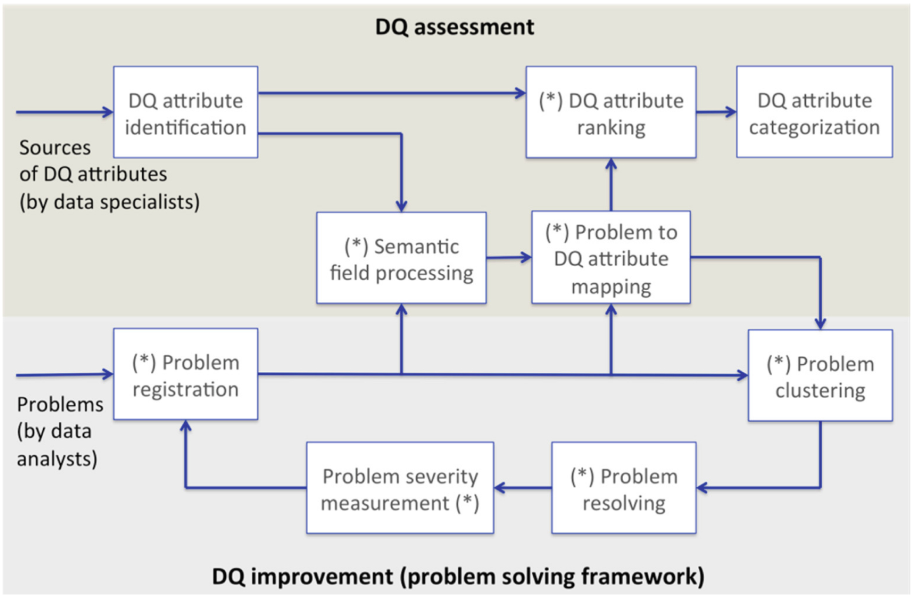
DQ assessment
DQ attribute identification
(*) DQ attribute ranking
DQ attribute categorization
Sources of DQ attributes (by data specialists)
(*) Semantic field processing
(*) Problem to DQ attribute mapping
(*) Problem registration
(*) Problem clustering
Problems (by data analysts)
Problem severity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

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

In the last subsection we elaborate more on the solution choosing and propose a separate framework for solution management, mainly to support the Problem Clustering and Problem Resolving components. This framework allows us to guide the process of solution choosing, which in turn can lead to decision support on how to solve DQ related problems.

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.

Semantic Field Processing. A semantic-field is a set of conceptually related terms [30]. 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)' (Table 2). Formally, every DQ attribute DQ_m (where $m : 1\ldots M$) 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,

$$DQ_m \equiv S_m = \left\{ RT_{m,i} \middle| i : 1\ldots M_m \right\} \tag{2}$$

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

$$RT_{m,i} = \left\{ PS_{m,i,j} \middle| j : 1\ldots M_{m,i} \right\} \tag{3}$$

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

$$PS_{m,i,j} = \left\{ PH_{m,i,j,k} \middle| k : 1\ldots M_{m,i,j} \right\} \tag{4}$$

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 Fig. 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

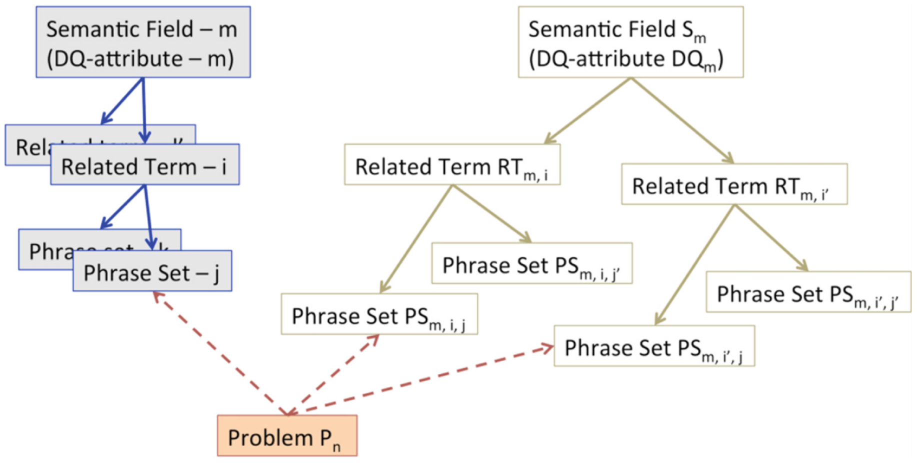

Fig. 4. An illustration of the hierarchical structure of semantic fields, related terms and phrase sets; and their relation to problems (the texts in grey blocks are intentionally abbreviated).

are common among semantic-fields/DQ attributes, and there are no phrase sets that are common among related terms.

$$\begin{array}{ll} RT_{m,i} \neq RT_{m',i'} & \forall m \neq m' \text{ or } i \neq i' \\ PS_{m,i,j} \neq PS_{m',i',j'} & \forall m \neq m' \text{ or } i \neq i' \text{ or } j \neq j' \end{array} \tag{5}$$

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

Table 2. Example of a semantic field-processing table (over the DQ attribute "completeness")

Phrase_1[a]	Phrase_2[a]	Related terms	DQ attribute (semantic field)
Is	Missed	Missing data	Completeness
Are	Missed	Missing data	Completeness
Be	Added	Adding data	Completeness
Is	Deleted	Lost data	Completeness
Are	Deleted	Lost data	Completeness

[a]*Derived from problem description. {Phrase_1, Phrase_2} is called a Phrase Set*

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 Fig. 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

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

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

$$A = \left[a_{n,m}\right]_{N \times M} \text{where } a_{n,m} = \begin{cases} 1 & \text{if} S_m \in S(P_n) \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Note that if $a_{n,m} = 0$ for all $m : 1 \ldots M$, 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

$$A_w = \left[aw_{n,m}\right]_{N\times M}\text{where } aw_{n,m} = a_{n,m}\cdot(MS_n - DS_n) \tag{8}$$

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 $PU_n.(MS_n{-}DS_n)$ to obtain the *extended weighed association matrix* as

$$A_{ew} = \left[aew_{n,m}\right]_{N\times M}\text{where } aew_{n,m} = a_{n,m}\cdot PU_n\cdot(MS_n{-}DS_n) \tag{9}$$

Note that the problems in the ITS are registered by data analysts, and therefore PU_n denotes the urgency of the problems from the viewpoint of the data analyst. This is a local worldview, because it is perceived from the viewpoint of a specific problem as observed by a specific data analyst in the field.

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 $m:1\ldots M$ is defined as:

$$R^d_m = \frac{\sum_{n=1}^{N} aw_{n,m}}{\sum_{n=1}^{N}\sum_{m=1}^{M} aw_{n,m}}\text{ or }\frac{\sum_{n=1}^{N} aew_{n,m}}{\sum_{n=1}^{N}\sum_{m=1}^{M} aew_{n,m}} \tag{10}$$

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 Relation (7) for $m:1\ldots M$ by:

$$R^s_m = \frac{\sum_{n=1}^{N} a_{n,m}}{\sum_{n=1}^{N}\sum_{m=1}^{M} a_{n,m}} \tag{11}$$

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 Fig. 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 Fig. 3, which are elaborated upon in the following.

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. Data consistency problems for instance can be resolved by adopting a centralized architecture. Both data consistency and data completeness problems can be resolved by improving registration protocols or by implementing constraints at the physical database level (i.e. integrity and value-required ("not null") constraints for data consistency and data completeness problems, respectively).

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:

$$(P_n, DQ_m) = \begin{cases} a_{n,m} & \text{see } (7) \\ aw_{n,m} = a_{n,m} \cdot (MS_n - DS_n) & \text{see } (8) \\ aew_{n,m} = a_{n,m} \cdot PU_n \cdot (MS_n - DS_n) & \text{see } (9) \end{cases} \qquad (12)$$

We specify every problem P_n by the vector $((P_n, DQ_1), (P_n, DQ_2), \cdots, (P_n, DQ_M))$ 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 Fig. 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 behaviors 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 behavioral 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.

A solution may address multiple registered problems all together. When this occurs, applying the solution affects all corresponding MS_n and even the DS_n. In practice applying a solution may change the DS_n, which was initially inserted by a data analyst. For example, when implementing the solution it may turn out that the problem is (partial) infeasible to fix. In the following, we propose a method for choosing appropriate solutions, which resembles such a classification case.

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 $S_k = \left(s_{k,1}, \cdots, s_{k,m}, \cdots, s_{k,M}\right)$ where for $m : 1\ldots,$M we have

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

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 [19] 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:

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

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 normalized 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

$$CB_k = SC_k - BF_k \text{ for } k : 1\ldots\text{K} \tag{15}$$

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). We elaborate further on choosing the best solution in Subsect. 3.3.

Problem Severity Measurement. KPIs can be defined and used to measure the momentary severity of problems. As shown in Fig. 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 Fig. 5. For our DQ management one can normalize the difference time series to derive problem severity level at a given moment t by

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

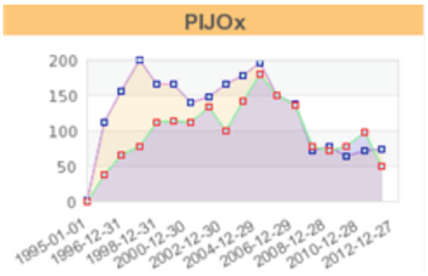

Fig. 5. Visualizations of two time series.

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

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

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 Fig. 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

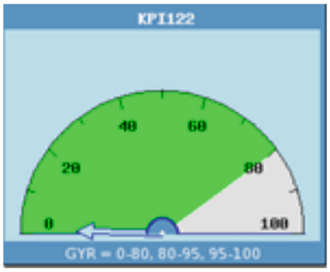

Fig. 6. Visualizations of the resulting-ratio dashboard.

subjective measurement can be useful when, for example, combining multiple and heterogeneous measures as defined in Relations (16) and (17).

3.3 A Framework for Data Quality Solutions Management

The data quality management architecture as described in [9] and in the previous subsections is sound for resolving DQ related problems. In this subsection we will elaborate more on the solution choosing, and propose a separate framework for solution management, mainly to support the Problem Clustering and Problem Resolving components. This framework allows us to formalize the process of solution choosing, which in turn can lead to decision support on how to solve DQ related problems.First, we will discuss for every step in the DQ improvement layerof our DQ management architecturethe impact on solution choosing.

- During Problem Registration a specific problem is registered in the ITS. In this stage only the local scope and urgency is known. Maybe solutions are proposed, but this will be also be done from a local perspective. For example, when data contain impossible values (i.e. outside the domain) the user might suggest setting these values to "unknown" as a temporary (local) solution.
- During Problem Clustering DQ analystsanalyze registered problems in order to cluster similar problems or classify them in terms of existing solutions. Due to the relation with DQ attributes (see Relation (13)), each problem is related to candidate solutions, i.e. the solution space. Problem Clustering defines the *theoretical* solution space, where the boundaries of the space are defined by the solutions that are applicable.
- During Problem Resolving, the boundaries of the solution space are narrowed down by the context of the problem. This is achieved by determining those solutions that are feasible given the current circumstances and by choosing the solution that has the best cost-benefit value.
- During Problem Severity Management some KPIs are developed to measure the momentary severity levels of resolved problems and compare them to the desired severity levels if possible. Note that these KPIs measure the effect of a chosen solution to a specific problem, and they do not take in account the possible effect of other solutions, nor the effect of the solution to other (not measured) problems.

Determining the cost-benefit valueof a solution requires a lot of knowledge about the domain and the context in which the solution has to be applied. This knowledge is often implicit and is hard to describe in a quantified cost-benefit value. In this paper, we propose an approach that gives an insight in the boundaries that influence chosen solutions, i.e. constrain the solution space. We distinguish the following boundaries, each of which cover several dimensions of the solution space:

1. Operational boundaries, such as resources (budget, people, software) and time frame. For instance, when you have a problem that should be solved by an expert, but there is no expert available and it is too costly or it takes too long to hire one, then this restriction forces to choose another – less optimal – solution.
2. Strategical boundaries, such as long-term business priorities. For instance, an organization has a certain budget for information management. It is a strategical choice how this budget is spent, e.g. focusing more on documentation or improving the data itself.
3. Organizational boundaries. An organization always has a certain role or scope regarding the data that are processed. For instance, an organization that is not involved in the registration of the data (e.g. a external research organization) will not be able to perform solutions that improve the registration process.
4. Domain-specific boundaries. Sometimes the domain in which data quality problems occur invoke limitations on the possibilities to improve data quality. For instance, in the criminal justice domain comparing police statistics to prosecution statistics (which is the next phase in the criminal justice chain) is challenging because exact matching of datasets is impossible due to a lack of common keys among datasets. This excludes several solutions for improving data quality at a record level.

Data analysts working in a specific domain and for a specific organization will have a good feeling for these boundaries when exploring the solution space for candidate solutions. However, a lot of these boundaries are flexible and should also be evaluated in a cost-benefit analysis. When the organization changes its strategy on information management, this can have immediate effect on the solutions. More interestingly, when the management of the organization has enough insight in the consequences of their strategical choices, they might change their strategy.

Data quality problems and their solutions, which are determined by the boundaries of the current solution space, can be seen as a solution model. Different solution models can be obtained by changing the boundaries of the solution space. The solution model with the best cost-benefit value might differ from the current one. In this way, i.e. by choosing an appropriate solution model, changing the DQ improvement strategy becomes part of the general information management strategy, which eventually leads to a more mature organization.

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., $M_{m,i,j} \leq 2$ 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 ta-
ble
▹ 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 [31]. For example, related term 'missing data' can be decomposed to phrase pairs {Is, Missed}, {Are, Missed} 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 [17], 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 Fig. 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.

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

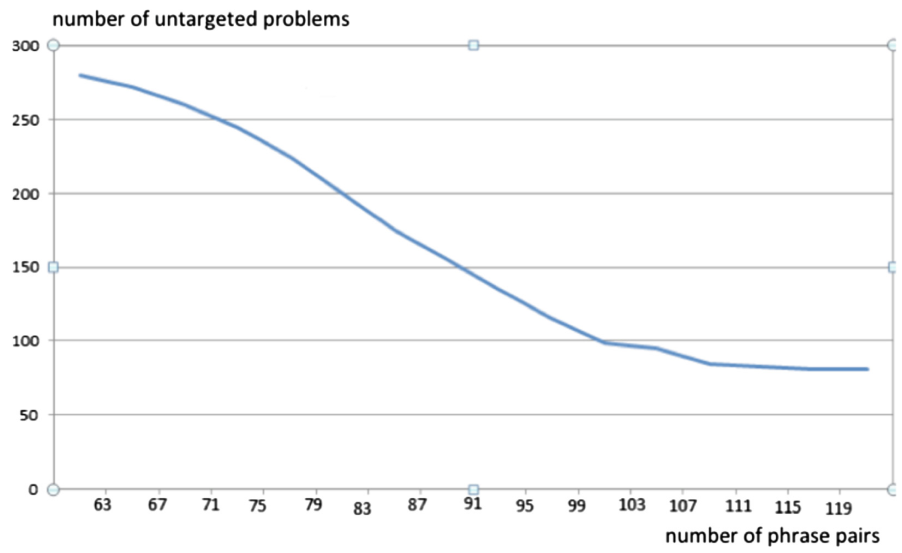

Fig. 7. Number of untargeted problems (vertical) in terms of the number of related terms.

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, [24] 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 Conclusion and Further Research

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.

An important aspect of problem resolving in DQ management is to determine the capabilities and costs of candidate solutions. In this contribution we presented a framework for decision support in the solution choosing process by guiding the choice of the cost-benefit values of candidate solutions. In the future, we intend to formalize the proposed framework for DQ solution management and to develop a proof of concept in order to research how the framework can contribute to maturing DQ management in organizations with data intensive applications, and in collaboration with our system architecture for resolving DQ related problems.

Acknowledgements. Partial results of this work were presented earlier in [9]. Tables, figures and equations have their origin in this paper, unless stated otherwise.

References

1. Choenni, S., Leertouwer, E.: Public safety mashups to support policy makers. In: Andersen, K.N., Francesconi, E., Grönlund, Å., Engers, T.M. (eds.) EGOVIS 2010. LNCS, vol. 6267, pp. 234–248. Springer, Heidelberg (2010). doi:10.1007/978-3-642-15172-9_22
2. Netten, N., van den Braak, S., Choenni, S., Leertouwer, E.: Elapsed times in criminal justice systems. In: Proceedings of the 8th International Conference on Theory and Practice of Electronic Governance (ICEGOV), pp. 99–108. ACM (2014)
3. van Dijk, J., Kalidien, S., Choenni, S.: Smart monitoring of the criminal justice system. Government Information Quarterly. Elsevier (2016). doi:10.1016/j.giq.2015.11.005
4. Christoulakis, M., Spruit, M., van Dijk, J.: Data quality management in the public domain: a case study within the Dutch justice system. Int. J. Inf. Qual. **4**(1), 1–17 (2015)
5. Birman, K.P.: Consistency in distributed systems. In: Guide to Reliable Distributed Systems, pp. 457–470. Springer, Heidelberg (2012)
6. Davenport, T.H., Glaser, J.: Just-in-time delivery comes to knowledge management. Harvard Bus. Rev. **80**(7), 107–111 (2002)

7. Bargh, M.S., van Dijk, J., Choenni, S.: Dynamic data quality management using issue tracking systems. IADIS Int. J. Comput. Sci. Inf. Syst. **10**(2), 32–51 (2015). Isaias, P., Paprzycki, M. (eds.)
8. Bargh, M.S., Mbgong, F., Dijk, J. van, Choenni, S.: A framework for dynamic data quality management. In: Proceedings of the IADIS International Conference Information Systems Post-Implementation and Change Management, pp. 134–142 (2015)
9. Bargh, M., van Dijk, J., Choenni, S.: Management of data quality related problems - exploiting operational knowledge. In: Proceedings of the 5th International Conference on Data Management Technologies and Applications (DATA), pp. 31–42. SciTePress (2016)
10. Batini, C., Cappiello, C., Francalanci, C., Maurino, A.: Methodologies for data quality assessment and improvement. ACM Comput. Surv. **41**(3), 16–52 (2009). Article no. 16
11. Wand, Y., Wang, R.Y.: Anchoring data quality dimensions in ontological foundations. Commun. ACM **39**(11), 86–95 (1996). ACM
12. Davoudi, S., Dooling, J.A., Glondys, B., Jones, T.D., Kadlec, L., Overgaard, S.M., Ruben, K., Wendicke, A.: Data quality management model (2015 Update). J. AHIMA **86**(10), 62–65 (2015). expanded web version
13. Knowledgent: White Paper Series: Building a Successful Data Quality Management Program. http://knowledgent.com/whitepaper/building-successful-data-quality-management-program/. Accessed 31 Oct 2015
14. Halevy, A., Rajaraman, A., Ordille, J.: Data integration: the teenage years. In: Proceedings of the 32nd International Conference on Very Large Data Bases, pp. 9–16. VLDB Endowment (2006)
15. Wang, R.Y., Strong, D.M.: Beyond accuracy: what data quality means to data consumers. J. Manage. Inf. Syst. **12**(4), 5–33 (1996)
16. Price, R., Shanks, G.: A semiotic information quality framework. In: Proceedings of International Conference on Decision Support Systems (DSS), pp. 658–672 (2004)
17. Woodall, P., Borek, A., Parlikad, A.K.: Data quality assessment: the hybrid approach. Inf. Manage. **50**, 369–382 (2013)
18. Bargh, M.S., Choenni, S., Meijer, R.: Privacy and information sharing in a judicial setting: a wicked problem. In: Proceedings of the 16th Annual International Conference on Digital Government Research, pp. 97–106. ACM, New York (2015)
19. Jiang, L., Barone, D., Borgida, A., Mylopoulos, J.: Measuring and comparing effectiveness of data quality techniques. In: Eck, P., Gordijn, J., Wieringa, R. (eds.) CAiSE 2009. LNCS, vol. 5565, pp. 171–185. Springer, Heidelberg (2009). doi:10.1007/978-3-642-02144-2_17
20. Bugzilla Website. https://www.bugzilla.org. Accessed 31 Oct 2015
21. JIRA Software Website. https://www.atlassian.com/software/jira. Accessed 31 Oct 2015
22. H2desk Website, https://www.h2desk.com. Accessed 31 Oct 2015
23. TOPdesk Website. http://www.topdesk.nl. Accessed 31 Oct 2015
24. Canovas Izquierdo, J.L., Cosentino, V., Rolandi, B., Bergel, A., Cabot, J.: GiLA: GitHub label analyzer. In: IEEE 22nd International Conference on Software Analysis, Evolution and Reengineering, pp. 479–483, Montreal, Canada (2015)
25. Environmental protection agency: data quality assessment: a reviewer's guide, Technical report EPA/240/B-06/002, EPA QA/G-9R (2006)
26. Pipino, L.L., Lee, Y.W., Wang, R.Y.: Data quality assessment. Commun. ACM **45**(4), 211–218 (2012). ACM
27. Eppler, M.J., Wittig, D.: Conceptualizing information quality: a review of information quality frameworks from the last ten years. In: Proceedings of the Conference on Info Quality, pp. 83–96 (2000)
28. Lee, Y.: Crafting rules: context-reflective data quality problem solving. J. Manage. Inf. Syst. **20**(3), 93–119 (2003)

29. Ryu, K.S., Park, J.S., Park, J.H.: A data quality management maturity model. ETRI J. **28**(2), 191–204 (2006)
30. Kornai, A.: The algebra of lexical semantics. In: Mathematics of Language, pp. 174–199. Springer, Heidelberg (2010)
31. Mooney, R.J.: Learning for semantic parsing. In: Gelbukh, A. (ed.) CICLing 2007. LNCS, vol. 4394, pp. 311–324. Springer, Heidelberg (2007). doi:10.1007/978-3-540-70939-8_28