Detecting Quality Problems in Data Models by Clustering Heterogeneous Data Values

Viola Wenz
Philipps-Universität Marburg
viola.wenz@uni-marburg.de

Arno Kesper
Philipps-Universität Marburg
arno.kesper@uni-marburg.de

Gabriele Taentzer
Philipps-Universität Marburg
taentzer@uni-marburg.de

***Abstract*—Data is of high quality if it is fit for its intended use. The quality of data is influenced by the underlying data model and its quality. One major quality problem is the heterogeneity of data as quality aspects such as understandability and interoperability are impaired. This heterogeneity may be caused by quality problems in the data model. Data heterogeneity can occur in particular when the information given is not structured enough and just captured in data values, often due to missing or non-suitable structure in the underlying data model. We propose a bottom-up approach to detecting quality problems in data models that manifest in heterogeneous data values. It supports an explorative analysis of the existing data and can be configured by domain experts according to their domain knowledge. All values of a selected data field are clustered by syntactic similarity. Thereby an overview of the data values' diversity in syntax is provided. It shall help domain experts to understand how the data model is used in practice and to derive potential quality problems of the data model. We outline a proof-of-concept implementation and evaluate our approach using cultural heritage data.**

I. Introduction

The digital transformation of our society is an ongoing challenge and nearly every action in the digital world creates data. To effectively use the collected data, it has to be of high quality. Data quality, often defined as "fitness for use" [1], comprises various dimensions, such as completeness, accuracy, and timeliness. Quality problems can be manifold and may affect one or more quality dimensions. The first essential step to better data quality is to identify existing quality problems in data and investigate their causes.

Heterogeneity, also referred to as representational inconsistency, is considered a data quality problem [2] in general. We focus on heterogeneity of data values stored in the same data field. Its cause may be the underlying data model as long as it allows such a heterogeneity. Data fields that contain heterogeneous data values further may point to a lack of structure in the related data model. Since "data quality techniques become increasingly complex as data looses structure" [3], a technique for systematically analysing data fields in this respect is needed as an essential step towards data model improvement.

Considering cultural heritage data, for example, domain experts constantly try to improve the quality of their data and data models. Nevertheless, data values, such as artist names, titles and creation dates of cultural objects, can be very heterogeneous in form and content. In this domain, data is typically collected manually and data models often allow wide ranges of data. Data transformations introduce further heterogeneity. Let's consider the width and height information of cultural objects, for example. Stored as semi-structured data, there may occur entries such as `101.2 cm`; `2 m`; `3.5 x 4.5 cm`; `100 m?` and even `–`. They show that the content of this data field is underspecified. It is used not only to store one value but may contain also information about the measurement unit as well as several values. Even meta information such as uncertainty (expressed with `?`) and lack of knowledge (denoted by `–`) are stored in the same data field. This heterogeneity may indicate quality problems of the underlying data model, in particular a lack of structure to represent such additional information. Similar problems may occur also in other domains where data is often semi-structured and entered manually, such as the domain of biodiversity [4].

Heterogeneity of data stored in the same field is a quality issue as it is more difficult to process such unstructured data than clearly structured data. The causes for heterogeneous data in single data fields can be manifold: The acquisition of data is not as accurate as it should be, the data management software and the underlying data model are not adequate for the kind of data acquired or the transformation that produced that data is faulty. We focus on the quality of the underlying data model here as it is the core of data management and therefore, plays a critical role. Acquisition software and actual data acquisitions as well as data transformations all depend on data models.

The importance of data model quality was discovered early. A conceptual basis for data model quality management was laid by Moody and Shanks in [5]. They developed a framework with quality factors, such as completeness and understandability, quality metrics, and improvement strategies; they evaluated it in [6]. One of their findings was that metrics are of limited use for analysing data model quality as research participants rated qualitative descriptions of quality problems more useful than quantitative ones. This has motivated us to develop a qualitative analysis of data model quality.

Based on the observation that heterogeneous data is difficult to process, we present a *bottom-up approach that clusters values of selected data fields such that domain experts can explore unknown quality problems and requirements of data models.* For example, uncertain knowledge that is implicitly expressed and non-expected information given may indicate problems of the data model. The approach can be configured according to domain knowledge. To achieve meaningful clusterings, multiple iterations with modified configurations may

be needed. Finally, domain experts interpret the clustering concerning quality problems in the data model.

Motivated by data quality analysis, we have developed this clustering approach for identifying quality problems in data models. As it just takes a list of data values as input, it can also be used to identify quality problems of other kinds of models such as conceptual models [7] and meta-models [8], [9]. While quality properties of meta-models are typically concerned with the form of existing structures, our clustering approach can find out missing structure in meta-models. Models-at-runtime (cf. [10]), for example, that store and process data from system logs may show more heterogeneity than intended and may also profit from our clustering approach.

We start our presentation with motivating examples in Sec. II, introduce the concepts of our clustering approach in Sec. III, give an overview of available tool support in Sec. IV, and present an initial evaluation performed in Sec. V. Finally, we discuss related work in Sec. VI and conclude in Sec. VII.

II. Motivational Examples

To motivate our approach with examples, we consider two local databases on cultural heritage objects, such as paintings and buildings. They use the data models MIDAS [11] and LIDO v1.0 [12]. MIDAS is described in a manual but not realised as an XML schema. The MIDAS data we consider was created by domain experts manually. LIDO is a CIDOC-CRM [13] application and XML schema for harvesting and exchanging metadata of collectibles. The LIDO data was created via a data transformation applied to data in MIDAS.

For our running example, we focus on measurement information about cultural heritage objects. In LIDO, measurements of objects are expressed with the element `measurementsSet`. It contains the following three elements: `measurementUnit`, `measurementValue` and `measurementType`. The element `measurementUnit` serves as our running example. According to the LIDO schema, it may contain an arbitrary string value. The LIDO documentation describes this element as follows. *"Definition: The unit of the measurement. How to record: E.g. cm, mm, m, g, kg, kb, Mb or Gb. Repeat this element only for language variants."* [12]

We chose a database in LIDO that contains 87,042 `measurementUnit` elements with 179 distinct string values overall. Based on the LIDO documentation, we expect simple indications of the measurement unit only. However, a diversity of values was found. Examples are `-`; `-10.5 cm`; `x 55 cm`; `cm / 120 cm` and `? cm`. (Note that in the original data a comma is used as the decimal separator as it is in German.)

As a second example, we investigated a MIDAS database on cultural heritage objects that includes 118,032 distinct values in the field `artist name`. The MIDAS manual [11] contains verbal descriptions of several rules on how certain information should be expressed in this field. Basically, the first, last and middle name(s) of an artist should be given. The field is used as an identifier for artists, for example, to relate them to an object. Thus, the manual lists additional information that can be appended to ensure the uniqueness of the entry. Furthermore, the manual explains how to encode different types of uncertainty concerning the artist or artist name. As the acquisition software based on MIDAS does not ensure these rules, the data found in this field is pretty heterogeneous. Examples include `Munch, Edvard`; `B., I. C.`; `Zindel, Peter (1841)`; `Lay?, ? de`; `Walt ..., R.` and `Grass, A. / Graß, Adolf / Grohs, A.`.

Furthermore, the database contains 52,523 distinct values in the field for dating objects. According to the MIDAS manual, values may contain the indication of a year and, if known, also the indication of a month and a day. The manual further explains how time spans and different types of uncertainty should be encoded. Again, as those rules are not ensured automatically, the data of this field is pretty heterogeneous as well. Examples (adapted to English) include `1895/1902`; `1686.10.24`; `x`; `ca. 1781`; `after 174ante`; `since 1927`; `unknown`; `1900-around 1991`; `1847 (Original 1846)` and `Beginning 20th century`.

As the examples show, data values of a field may be quite heterogeneous, often due to missing structure in the data model. The heterogeneity implies that the data and the data model may be of low quality. For example, the understandability and comparability of this data may suffer. Thus, *quality assurance* is necessary. Analysing data values manually, however, is a time-consuming task. Regular expressions can be used to implement all the rules given in the manuals, to find data with quality issues. But further heterogeneity may be present in the data values, often due to the need to store additional information. Thus, an explorative approach is needed that supports domain experts in gaining an overview of the data values given and adapting the data model accordingly.

III. Approach

We present a bottom-up approach for supporting domain experts in detecting quality problems in data models. The idea is to cluster all data values of a field of interest by syntactic similarity. The degree to which certain syntactic features influence the similarity between data values depends on the field analysed. Thus, domain experts can configure the clustering process according to their domain knowledge. The resulting clustering provides an overview of the data values' diversity in syntax. By interpreting it, domain experts may identify quality problems in the data model. In particular, the clustering can reveal the encoding of diverse information in a field through specific syntax. This is often caused by missing or non-suitable structure in the data model.

The *interactive workflow* of our approach is visualised in Fig. 1. It is intended to be usable by *domain experts* who are interested in analysing and improving the quality of their data model. The *inputs* to the workflow are a database and the underlying data model. The *output* of the workflow is a list of quality problems of the data model. The workflow is *iterative* as it may be necessary to adjust the configuration and re-execute the clustering algorithm several times to achieve a useful clustering. The steps of the workflow are presented in more detail below using the running example.

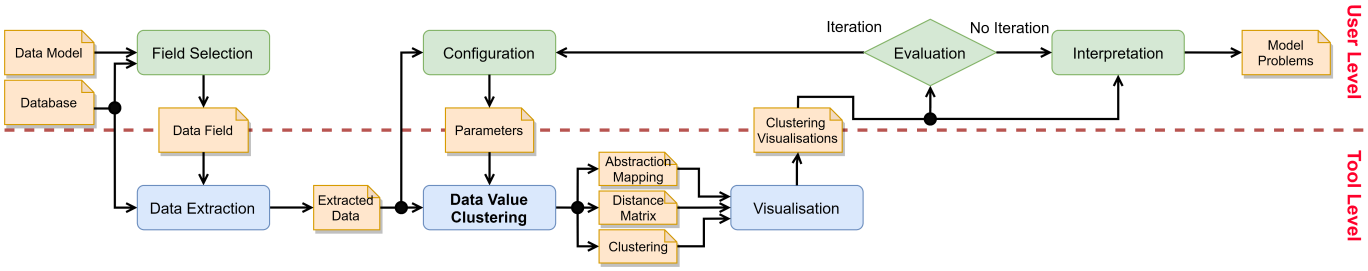

Fig. 1. Workflow of the approach to detecting quality problems in data models by clustering heterogeneous data values.

1) Field Selection: Given a database and the underlying data model, domain experts select a data field to be analysed. For the running example, we used a database on cultural heritage data based on LIDO. We selected the field `measurementUnit`.

2) Data Extraction: Next, all data values of the selected field are extracted from the database automatically. The result is a list of data values; note that the remaining part of the workflow is independent of the database technology used.

3) Configuration: Based on the observation that "incorporating domain expert input often improves clustering performance" [14], the data value clustering process is configured by domain experts. For this purpose, their domain knowledge has to be mapped to parameter values. Considering the running example on measurement units, the domain knowledge includes, for example, the fact that small sequences of letters are expected. The configuration facilities are further explained in the following paragraph.

4) Data Value Clustering: The data value clustering is performed in three steps as outlined in the following. Details and examples are explained in Section C of the appendix. Since our goal is to provide an overview of significant differences in the syntax of the data values, the first step is an *abstraction*. Thereby, syntactic features that, according to the configuration by domain experts, are irrelevant for clustering are removed. For example, we can abstract from concrete characters of a specific group of similar characters, such as letters, or from the length of certain character sequences. For the measurement units, for example, we abstract from the length of digit sequences. The configuration depends on expectations about the syntax of the data values and the kinds of syntax variations of the values that cause significant variations in their meaning. Ultimately, the original data values are mapped to a smaller set of shorter values via a set of abstraction rules determined by the configuration. The result is the abstraction mapping. The abstraction step is a first grouping of similar values since each abstracted value represents a set of original values. In the running example, 179 values given originally are abstracted to 22 values. The coloured boxes in Fig. 2 represent groups of original values that were mapped to the same abstracted value.

If the abstraction does not produce a manageable amount of groups, our approach suggests clustering the abstracted values. As a prerequisite, pairwise *distances* (i.e., dissimilarities) between abstracted data values have to be computed. Having applied the approach to cultural heritage data, we achieved reasonable results with the basic *edit distance* that allows insertions and deletions of string characters only, and the Levenshtein distance [15], which additionally allows substitutions. The dissimilarity between two values depends on the data field analysed. Therefore, we use configurable weights for each edit operation (namely insertion, deletion and substitution) applied to each possible character. Domain experts configure the weights based on their domain knowledge. In general, edit operations of unexpected characters and operations that may have a significant impact on the values' meaning should be weighted high as they may indicate quality problems. For the measurement units, we therefore chose the weight for inserting letters lower than those for digits and special characters.

Finally, the abstracted values are *clustered* by syntactic similarity based on the calculated distance matrix. Only clustering algorithms that can operate on string distances can be applied, such as hierarchical clustering [16], k-medoids [17] and DBSCAN [18]. Domain experts have to select an algorithm and configure its parameters dependent on the data field analysed. We provide a setting that allows experimenting with a variety of clustering algorithms. For the running example, we chose hierarchical clustering. It is up to future research to investigate which clustering algorithms and parameter settings are most suitable for detecting quality problems in data models and how this depends on the kind of data considered.

5) Visualisation: For an overview, the clustering of the abstracted data values is presented. Per abstracted value, a corresponding original value is shown as a representative. Fig. 2 shows an excerpt of this representation for the running example. Each column of that table represents a cluster. Six out of ten clusters are shown. For further exploring the clustering, the mapping between original and representative values is shown, as by the coloured boxes in Fig. 2.

Moreover, we apply multidimensional scaling [19] to present the data values in a two-dimensional Cartesian space based on their distances. A corresponding scatter plot of the running example is shown in Fig. 3. Each dot is labelled with a representative value. Each cluster is represented by a different colour. This visualisation allows domain experts to get a quick impression of key properties of a clustering: a clustering is *compact* if there is a high similarity within clusters and it is

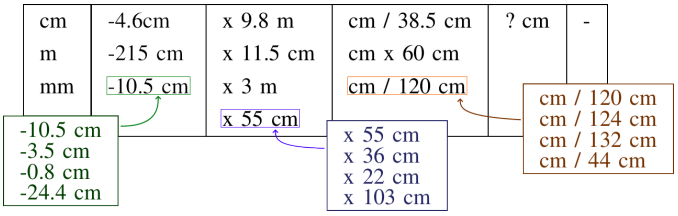

Fig. 2. Excerpt from clustering of measurement unit values showing representatives. Excerpt of represented original values shown in boxes.

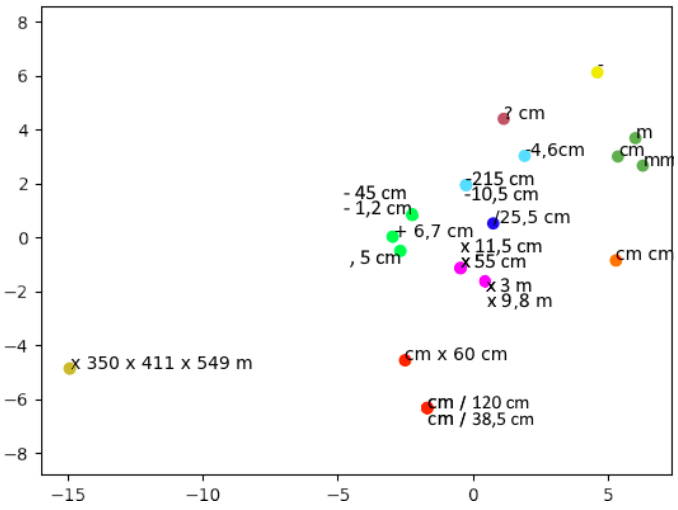

Fig. 3. Scatter plot of clustered measurement unit values resulting from multidimensional scaling. Note that a comma is used as the decimal separator.

separate if there is a low similarity between clusters [20].

6) Evaluation: To achieve a useful clustering that enables the detection of quality problems, it may be necessary to perform several iterations with modified configurations. Thus, the quality of the clustering must be analysed to decide whether another iteration is necessary. In general, the quality of clusterings can be evaluated by *internal* and *external* clustering validation [20]. Internal validation means evaluating the quality of the clustering without the use of any external information. It does not evaluate the quality of the clustering in the intended usage scenario. Therefore, external clustering validation takes external information into account [20]. In our case, this means that domain experts evaluate the clustering based on their domain knowledge. For each cluster, they should assess what kinds of values it includes, whether the grouping of these values makes sense, and how useful it is for the detection of quality problems. If the clustering does not bring new insights, the experts' domain knowledge may not have been adequately translated into parameters. The result of the evaluation is the decision whether another iteration is necessary. Note that an additional iteration does not always improve the clustering. If most clusters are considered useful and just a few seem internally heterogeneous, it may be profitable to cluster the values contained in the problematic clusters separately with a modified configuration. If ultimately the domain knowledge is appropriately translated into parameter values but the clustering still does not reveal significant differences in the data values, they may be homogeneous.

In the following, we sketch an external evaluation of the clustering in our *running example* (Fig. 2) based on several interviews with domain experts. The first cluster contains the expected values. The minus in the values of the second cluster indicates measurement values in form of intervals in the original MIDAS data. The "x" in the values of the third cluster indicates measurements of multiple dimensions. The values in the fourth cluster imply measurements of multiple dimensions or alternative measurements of the same dimension that are taken from different sources. Cluster five and six represent values that encode different types of uncertainty namely doubtful or missing information. The domain experts considered the clustering useful in general since it gives a systematic overview of the groups of values that differ in syntax, semantics and cause.

But as indicated by Fig. 3, cluster four (which is coloured red) is not very compact and the included characters "x" and slash may have significantly different meanings. Hence, we might consider a clustering more useful where cluster four is split up correspondingly. To achieve this, we would need another iteration with a modified configuration. Admittedly, additional iterations are more profitable if the induced change in the clustering is of greater magnitude.

7) Interpretation: Ultimately, the domain experts interpret the clustering concerning data model quality. They should evaluate what quality problems of the data values the clustering reveals and what may be their causes in the data model.

In the following, we outline the interpretation of our *example clustering* (Fig. 2) by domain experts. They recognised the problem that, besides the expected units, the data values also include measurement values and special characters. They inferred that values of an enumeration only should be allowed. Based on the second cluster, the experts identified the need to reconsider the representation of intervals as measurement values in LIDO. Clusters three and four imply that support for measurements of multiple dimensions and alternative measurements of the same dimension should be investigated. Also, the documentation and the data transformation must be checked correspondingly. Clusters five and six indicate the need to support uncertain information explicitly. In sum, the clustering helped the experts to identify several quality problems in LIDO.

IV. Tool Support

In the following, we report on a proof-of-concept implementation of our approach [21]. Fig. 4 shows the tool architecture. The `Clustering-Based Analyser` realises the workflow presented in Sec. III. It controls the data transfer between the other components. The data values are queried from a database in the component `Data Extraction`. The component `Export` allows saving all parameters and the clustering in JSON format and creates a representation of the clustering as an Excel file. The components `Data Value Clustering` and `GUI` are discussed in the following subsections.

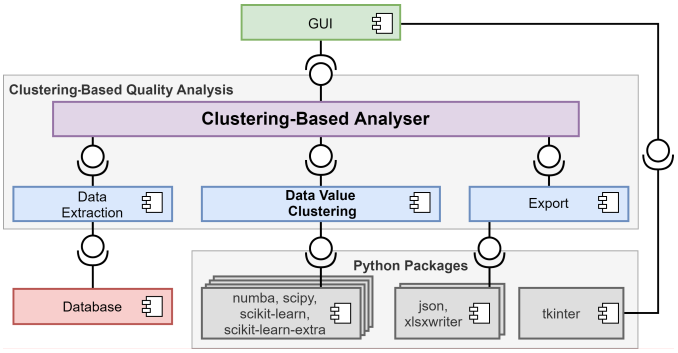

Fig. 4. Component diagram of the implementation.

A. Data Value Clustering

The component `Data Value Clustering` is a Python implementation of the algorithm presented in Section III-4 and allows performing data value clustering via an API.

For supporting the abstraction step, we implemented a set of abstraction rules and a mechanism for applying those rules to data values. It can be extended by further rules in the future.

For the calculation of the basic edit distance and the Levenshtein distance, we implemented a parallelisable version of the Wagner–Fischer algorithm [22]. For improving the performance of distance calculations for large sets of potentially long data values, we use Numba [23], a compiler that translates a subset of Python code into faster machine code.

We use implementations of clustering algorithms offered in the packages scikit-learn [24], scikit-learn-extra [25] and scipy [26].

B. GUI

To facilitate the usage of our tool, we provide a graphical user interface. It is still under development and not yet empirically evaluated. We give a short overview of the GUI developed so far. Details are provided in Section B of the appendix. The implementation of our GUI is based on the package tkinter [27].

The GUI should enable domain experts to configure the tool such that all relevant *domain knowledge* is taken into account. For the configuration of the *abstraction*, we developed a binary response questionnaire. The questions aim at the expert's assessment of the importance of certain syntactical features. The answers are translated to valid combinations of abstraction rules. Abstracted data values resulting from the application of these rules are shown exemplarily and updated dynamically.

Currently, the weights for the *Levenshtein distance* can be specified directly as numbers or by placing graphical objects, which represent groups of characters, on a 2D canvas, whereby their distances define the weights for corresponding substitutions. We plan to investigate how far the distance configuration process can be simplified, for example by deriving substitution weights automatically to decrease the number of parameters that must be specified.

For *selecting a clustering algorithm and configuring its parameters*, our tool provides a graphical interface based on standard widgets. How experts could be supported in appropriately selecting and configuring a clustering algorithm based on domain knowledge and experience is still subject to future research.

Finally, *calculated clusterings are visualised* using tables and scatter plots (as presented in Sec. III-5).

V. Initial Evaluation

We investigate the following research questions:

RQ1: How far does the approach support the detection of quality problems in data models?

- **RQ1.1:** Does the approach support the detection of quality problems in data models?
- **RQ1.2:** What kinds of quality problems in data models can be revealed with the approach?

In the following, the setup of our evaluation is presented. Findings and threats to validity are discussed thereafter.

A. Setup

We selected four diverse data fields from two XML databases with cultural heritage data using the data models LIDO [12] and MIDAS [11]. The data fields are briefly presented in Table I and Sec. II. We refer to them by the descriptive names given in the first column. The fourth column shows the number of distinct values in the corresponding database. Altogether, the selected fields cover both numerical information (dating) and textual information (artist name, measurement unit and attribution qualifier). Furthermore, we selected fields that expect just a few different values (measurement unit and attribution qualifier) and fields where data may occur in very diverse manifestations (artist name and dating).

For the evaluation of our approach, we considered clusterings of the values of the selected data fields. To answer the research questions, the quality of the clusterings must be evaluated in the intended usage scenario. Consequently, we applied *external* clustering validation (see Sec. III-6): For the ultimate *evaluation and interpretation of the clusterings* we recruited four domain experts on cultural heritage data. Their expertise covers the following task areas: data model development, usage and development of acquisition software, data acquisition, editing of data, and definition of data transformations. All experts know both data models.

Since the GUI of the tool is still under development, it was not feasible yet to let domain experts go through the whole workflow. To get the clusterings, we configured the algorithm based on our own domain knowledge, therefore. Our domain knowledge is based on a greater experience in the field of cultural heritage data of more than a year, in particular, good insight into the data models MIDAS and LIDO. Hence, we assumed that domain experts could have arrived at similar configurations. We assessed the quality of the initial clustering manually and in some cases modified the configuration slightly a few times until we yielded a clustering that made sense from our point of view. The configurations are explained in Section A of the appendix.

TABLE I
OVERVIEW OF DATA FIELDS USED FOR THE EVALUATION

Field	Model	Explanation	#Values	Example values (adapted to English)
Artist name	MIDAS	Name of an artist	118,032	"Martin, A. (junior)", "Raffael?", "Walt ..., R."
Dating	MIDAS	Dating of an object, such as completion or destruction	52,523	"ca. 1840-1850", "1895/1902", "x"
Measurement unit	LIDO	Unit of a measurement of an object	179	"-10.5 cm", "x 55 cm", "? cm"
Attribution qualifier	LIDO	Characterisation of the attribution of an actor to an event	56	"attributed?", "studio / successor"

We presented the four clusterings to each of the experts independently using Excel files, each including three sheets. The first sheet shows a manually compiled excerpt of the corresponding data model documentation. The other sheets were created by our tool and present the clustering of the abstracted values by means of original values as representatives (cf. Fig. 2), and the clustering of the original values. The Excel files are provided in [28].

For each of the four clusterings, we firstly asked for an *evaluation* of the clustering with the following questions: (1) Which kinds of values are included in each cluster? (2) Does the clustering make sense? (3) Does the clustering help in detecting quality problems in the set of data values? (4) Does the clustering bring new insights about the data considered?

Secondly, we asked for an *interpretation* of the clustering concerning data model quality using the following questions: (5) Which quality problems of the data values does the clustering reveal? (6) Does it reveal quality problems of the data values you would hardly detect otherwise? (7) What are causes for these problems in the data model? (8) Which quality improvements of the data model would you suggest?

At the end of each session, we asked the expert for a final *conclusion* using the following questions: (9) Do the clusterings support the detection of quality problems in data models? (10) What kinds of quality problems in data models can be revealed by clusterings of data values?

B. Findings

Per clustering, we summarise the answers to Questions (1) and (2) in the *evaluation* paragraph and the answers to (5), (7) and (8) in the *interpretation* paragraph. The answers to (3), (4) and (6) were similar for all clusterings and thus, are discussed together with those to (9) and (10) in Sec. V-B5.

1) Artist Name:

a) Evaluation of the Clustering: The domain experts agreed that the clustering makes sense. They intuitively assigned a specific meaning to most of the clusters. For example, one cluster contains values with additional information appended in brackets. The experts explained that this is often done to ensure the uniqueness of entries as artist names are used as identifiers in MIDAS. Another cluster comprises values that contain question marks at different positions. According to the experts, they encode different kinds of uncertainty, for example, concerning the first name or the relation between an object and the artist. Further clusters represent further uncertainties encoded implicitly using other special characters. The values found in the database do not necessarily conform to the syntax rules described in the MIDAS manual (see Sec. II). For example, the value `Heitz, Heinrich ?` is not covered by any of the rules. Thus, the set of data values is even more heterogeneous than the manual implies.

b) Interpretation of the Clustering: The experts identified an encoding of additional information, which is given implicitly and shows some heterogeneity. They considered it as problematic and largely attributed it to the fact that the artist names are used as identifiers and therefore need to be unique. The experts suggested using numbers that are automatically generated as identifiers instead. Different kinds of uncertainty in the data field were considered as a further problem. The experts identified the requirement to support the different kinds of uncertainty explicitly by using additional structure in the data model to increase understandability. Furthermore, they questioned whether it makes sense to accept arbitrary string values as artist names in MIDAS. Some experts mentioned a splitting of the name into several fields.

2) Dating:

a) Evaluation of the Clustering: Overall, the experts considered the clustering meaningful. The discussions of the following clusters were most interesting: There are several value clusters that contain specific textual indications of uncertainty or imprecision as a prefix to a numerical date, such as `around`. Often, several (uncertain) dates are given in one value, separated by a minus or slash, such as `after 1831/before 1852`. The experts stated that entries which include a minus or slash probably represent a period of time and an unknown point in time within the interval, respectively. Other clusters comprise entirely textual values indicating some lack of knowledge, such as `without year`. Again, the set of data values considered contains more implicit encodings than mentioned in the MIDAS manual (see Sec. II).

b) Interpretation of the Clustering: The implicit and heterogeneous encoding of uncertainty, imprecision and missing knowledge was identified as the main quality problem of the data values. The experts attributed the values' heterogeneity to the fact that they are not checked against syntax rules. Furthermore, they expressed the need to separate numerical dates (conforming to ISO 8601 [29]) from special characters and textual modifiers that encode uncertainty. The requirement to support the expression of uncertainty in a homogeneous and clear way was also mentioned. Also, the meaning of the separators should be differentiated by additional structure in the data model. Regarding this, two experts stated the need to support both certain and uncertain boundaries of intervals.

3) Measurement Unit: The evaluation and interpretation of the clustering are discussed in Sec. III-6 and Sec. III-7.

4) Attribution Qualifier:

a) Evaluation of the Clustering: The experts considered the clustering as useful and assigned a potential meaning to each cluster. For example, one cluster contains single words followed by a question mark, which indicates uncertainty. Furthermore, there are two clusters of values containing multiple statements separated by an ampersand or a slash. The experts read them as AND and XOR relations, respectively.

b) Interpretation of the Clustering: The experts identified an implicit encoding of multiple statements and uncertainty in the data values as problematic. To reduce the heterogeneity, they suggested to limit the set of allowed values to a controlled vocabulary. Furthermore, the meaning of the separators ampersand and slash should be distinguished explicitly by the data model via structure. The concatenation of multiple statements also raised the question whether the LIDO documentation could be clarified regarding the repetition of this field. Potential problems in the transformation by which the data was created, were also mentioned. Besides, the experts mentioned the requirement that LIDO should support the documentation of uncertainty in another field. Some experts suggested the introduction of an additional field to distinguish between the following information: the type of the attribution (e.g. `attributed` vs. `alternative attribution`) and the relation between the person described with the data and the person attributed to the event (e.g. `school` vs. `successor`).

5) Conclusions by Domain Experts: Concerning Question (3), the experts stated that the clusterings give good overviews of the heterogeneous syntax of the data values. This allowed the experts to derive quality problems in the set of data values.

When considering Question (4), they agreed that all four clusterings bring new insights. The clusterings reveal how the data model is used in practice as opposed to how it is expected to be used. Deviations indicate quality problems of the data model, often related to previously unknown requirements. Further, quality problems of the data transformation can be revealed. Even when being aware of the problems previously, the experts found the clustering useful for understanding the problem in depth and for considering quality improvements.

To Question (6), they reported that, for data quality assurance, they typically use a list of all data values of a field of interest sorted alphabetically or by the number of occurrences. They agreed that clusterings provide a much clearer and more systematic overview than their previous practice. They stated the following advantages of clusterings: they allow the detection of a wider range of quality problems in data, the detection is faster, and the backtracking to quality problems in the data model is easier. The experts argued that the more data values there are in a field, the greater the benefit of clustering.

Concerning Question (9) and thus RQ1.1, the experts agreed that the clusterings provide an overview of the data values' heterogeneity in syntax, which supports the detection of quality problems in the data model, in part by revealing previously unknown requirements.

Concerning Question (10) and thus RQ1.2, the experts observed that the clusterings primarily reveal encodings of multiple information in a single value using specific syntax. Often, uncertainty about the actual information is encoded implicitly. According to the experts, this occurs either due to misuse of the data model, potentially caused by documentation issues, or because the data model does not support expressing the information explicitly. Often, the values' heterogeneity also indicates the use of wrong data types or a lack of syntax constraints. Note that some of the mentioned problems of LIDO v1.0 were addressed in LIDO v1.1 Public Beta [30].

C. Conclusion

Concerning RQ1.1, we found that, in the chosen setting, the approach supports the detection of quality problems in data models. With regard to RQ.1.2, the findings imply that especially missing structure but also inappropriate data types, lack of syntax constraints and problems in the documentation can be exposed. The clusterings further helped the experts to come up with ideas for improvements.

The presented evaluation was performed in the domain of cultural heritage, where data is often semi-structured and largely created manually. This may lead to a high amount of heterogeneity of data values, especially to implicit encodings of uncertainty. How the approach performs on significantly different kinds of data is subject to future research.

D. Threats to Validity

The main threat to *construct validity* is that the configurations were not performed by external domain experts but by ourselves. But as argued in Sec. V-A, we have experience in the domain and thus assume that external experts could have arrived at similar configurations. Once the GUI is completed, we will conduct a thorough empirical evaluation where the configuration will be performed by domain experts.

Threats to *external validity* could be the selection of the databases, fields and domain experts. The two data models we selected are diverse: LIDO is event-oriented, intended for data exchange and developed by a national working group, whereas MIDAS is object-oriented, intended for data acquisition and developed locally in a cultural heritage institution. As explained in Sec. V-A, the selected data fields are also quite diverse. To counter the relatively low number of experts, we selected them carefully to cover a broad range of tasks related to data acquisition, data models and data transformations. All experts provided valuable input to all questions.

VI. Related Work

Our approach brings together several different research activities: *clustering of data to detect data quality problems*, *approaches to homogenise data* and *quality assurance of models*. Hence, we consider related work in these directions.

A. Detecting Problems in Data Quality with Clustering

A variety of approaches apply clustering based on edit distances to detect *minor inconsistencies in textual data values*.

For example, such clustering is applied to detect *misspellings, typos and abbreviations* in textual geographical data [31], correct misspelled data values without external reference data [32], support the creation of authority files [33] and detect *duplicates* in medical records [34]. Our approach provides an overview of more significant differences in the syntax of data values to reveal problems in the underlying data model. Thus, we do not calculate the dissimilarity between data values based on individual characters but on interesting syntactical features determined by domain experts (see Sec. III-4).

Dai et al. [35] present a quantitative measure for data field heterogeneity based on cluster entropy and soft clustering. They focus on semantically different types of information given in the same column. They do not consider further forms of heterogeneity such as implicit encodings of additional information. Our qualitative approach allows finding quality issues of the underlying data model.

B. Approaches to Homogenising Data

A *pattern-based approach to homogenising data values* was proposed in [36]. It provides an overview of the data values' syntax similar to ours. It requires users to iteratively design patterns (consisting of regular expressions) manually. Instead, we use clustering to analyse the data in a bottom-up manner. This approach aims to unify inconsistent data values that represent the *same* type of information, whereas ours aims at detecting quality problems in data models, in particular *different* types of information being encoded in the same field.

Approaches to *clustering semi-structured data*, such as XML data [37], head towards the homogenisation of data structures. The underlying similarity measures focus on structural aspects. For measuring the similarity between data values, values are interpreted as multisets of words and token-based measures are applied. The authors of [37] identified the need to integrate domain knowledge into such clustering techniques. In summary, those approaches do not focus on domain-specific similarities of data values as we do and they do not aim to analyse the quality of data models.

C. Identifying Quality Problems in Data Models and Meta-models

A framework for *data model* quality management was proposed in [5]. It comprises quality factors of varying importance, quality metrics, and improvement strategies.

In a similar vein, a collection of quality attributes for *meta-models* was presented in [8]. Based on those quality attributes, an empirical study on the perception of meta-model quality was performed [38]. It showed that "the perceived quality was mainly driven by the meta-models' completeness, correctness and modularity" [38]. The authors noted that, in general, completeness and correctness are very hard to measure.

The detection and resolution of meta-model smells is presented in [39], an approach that is based on quality assurance for models in general [40]. Furthermore, there is an approach to meta-model testing via unit test suites and domain-specific expected properties with metaBest as well as an example-based construction of meta-models with metaBup [9], [41], [42].

While analysis approaches based on metrics [5], smells [39] and expected domain-specific properties of models [41] are concerned with the quality of existing structures, our bottom-up approach can find out *missing structure* and allows investigating previously *unknown requirements*. Due to this complementary nature of our approach, we are convinced that it is worthwhile to investigate the relevance of data value clustering also for analysing meta-model quality, especially if unstructured attribute values are allowed.

Models-at-runtime (cf. [10]), for example, may be used to store structural information about system logs. Applying our approach to models-at-runtime may reveal problems in their meta-models due to some heterogeneity in system logs.

VII. Conclusion

We present a bottom-up approach to detecting quality problems in data models via clusterings of data values along syntactic similarity. The approach is generic in that it is independent of the database technology used and can be adapted to different domains via the configuration. The investigation of clusterings can provide new insights concerning the actual usage of a data model, from which domain experts can derive unknown quality problems and requirements of the data model. We provide a proof-of-concept implementation that allows experimenting with different configurations. Our evaluation in the domain of cultural heritage data showed that the approach supports the detection of quality problems in data models, especially missing structures.

In future work, we will further investigate how domain knowledge can be mapped to configurations of data value clustering straightforwardly without technical knowledge. We will also investigate facilities to support experts in deciding whether further iterations are needed and in modifying a configuration suitably. Furthermore, experts shall be supported in categorising detected problems and improving the data model (e.g. similar to class model smells and refactorings [40]) as well as adapting the data to the changes. Ultimately, the whole workflow and the GUI will be evaluated empirically.

Our approach opens up new lines of research: So far, we have applied it to detect quality problems in data models. It may also be useful to find out missing structure and to investigate previously unknown requirements in conceptual models and meta-models as long as heterogeneous data values are concerned. In future, we also want to experiment with the clustering of values of several related data fields (identified, for example, with association rule mining [43]). And last but not least, observations by the interviewed domain experts imply that our approach may also be useful to explore the quality of data transformations from the bottom-up since heterogeneity of data values may also indicate quality issues in data transformations. We expect that quality assurance of model transformations would profit from those findings.

ACKNOWLEDGMENT

This work is partly funded by the German Federal Ministry of Education and Research (Grant No.: 16QK06A). We would like to thank Markus Matoni, Oguzhan Balandi, Martha Rosenkötter and Regine Stein for their valuable comments.

References

[1] G. K. Tayi and D. P. Ballou, "Examining data quality," *Communications of the ACM*, vol. 41, no. 2, p. 54–57, Feb. 1998. [Online]. Available: https://doi.org/10.1145/269012.269021

[2] A. Zaveri, A. Rula, A. Maurino, R. Pietrobon, J. Lehmann, and S. Auer, "Quality assessment for linked data: A survey," *Semantic Web*, vol. 7, no. 1, pp. 63–93, 2016. [Online]. Available: https://doi.org/10.3233/SW-150175

[3] C. Batini, C. Cappiello, C. Francalanci, and A. Maurino, "Methodologies for data quality assessment and improvement," *ACM Computing Surveys*, vol. 41, no. 3, Jul. 2009. [Online]. Available: https://doi.org/10.1145/1541880.1541883

[4] M. Diepenbroek, F. O. Glöckner, P. Grobe, A. Güntsch, R. Huber, B. König-Ries, I. Kostadinov, J. Nieschulze, B. Seeger, R. Tolksdorf, and D. Triebel, "Towards an integrated biodiversity and ecological research data management and archiving platform: The german federation for the curation of biological data (gfbio)," in *44. Jahrestagung der Gesellschaft für Informatik, Big Data - Komplexität meistern, INFORMATIK 2014, Stuttgart, Germany, September 22-26, 2014*, ser. LNI, E. Plödereder, L. Grunske, E. Schneider, and D. Ull, Eds., vol. P-232. Gesellschaft für Informatik e.V., 2014, pp. 1711–1721. [Online]. Available: https://dl.gi.de/20.500.12116/2782

[5] D. L. Moody and G. G. Shanks, "What makes a good data model? evaluating the quality of entity relationship models," in *Proceedings of The13th International Conference on the Entity-Relationship Approach*, ser. ER '94. Berlin, Heidelberg: Springer, 1994, p. 94–111.

[6] ——, "Improving the quality of data models: Empirical validation of a quality management framework," *Information Systems*, vol. 28, no. 6, p. 619–650, Sep. 2003. [Online]. Available: https://doi.org/10.1016/S0306-4379(02)00043-1

[7] D. L. Moody, "Theoretical and practical issues in evaluating the quality of conceptual models: Current state and future directions," *Data and Knowledge Engineering*, vol. 55, no. 3, p. 243–276, Dec. 2005. [Online]. Available: https://doi.org/10.1016/j.datak.2004.12.005

[8] M. F. Bertoa and A. Vallecillo, "Quality attributes for software meta models," University of Malaga, Spain, Tech. Rep., 2010.

[9] J. J. López-Fernández, E. Guerra, and J. de Lara, "Assessing the quality of meta-models," in *Proceedings of the 11th Workshop on Model-Driven Engineering/ Verification and Validation co-located with 17th International Conference on Model Driven Engineering Languages and Systems, MoDeVVa@MODELS 2014, Valencia, Spain, September 30, 2014*, ser. CEUR Workshop Proceedings, F. Boulanger, M. Famelis, and D. Ratiu, Eds., vol. 1235. CEUR Workshop Proceedings, 2014, pp. 3–12. [Online]. Available: http://ceur-ws.org/Vol-1235/paper-02.pdf

[10] M. Szvetits and U. Zdun, "Systematic literature review of the objectives, techniques, kinds, and architectures of models at runtime," *Software and Systems Modeling*, vol. 15, no. 1, p. 31–69, Feb. 2016. [Online]. Available: https://doi.org/10.1007/s10270-013-0394-9

[11] J. Bove, L. Heusinger, and A. Kailus, *Marburger Informations-, Dokumentations- und Administrations-System (MIDAS): Handbuch und CD (Literatur und Archiv; 4). - 4. überarbeitete Auflage*. K. G. Saur Verlag, 2001. [Online]. Available: https://archiv.ub.uni-heidelberg.de/artdok/3770/

[12] E. Coburn, R. Light, G. McKenna, R. Stein, and A. Vitzthum, "LIDO v1.0 (Lightweight Information Describing Objects)." [Online]. Available: http://www.lido-schema.org/schema/v1.0/lido-v1.0.xsd/

[13] C. Bekiari, G. Bruseker, M. Doerr, C.-E. Ore, S. Stead, and A. Velios, "Cidoc crm." [Online]. Available: http://www.cidoc-crm.org/

[14] Y. Chang, J. Chen, M. H. Cho, P. J. Castaldi, E. K. Silverman, and J. G. Dy, "Clustering with domain-specific usefulness scores," in *Proceedings of the 2017 SIAM International Conference on Data Mining, Houston, Texas, USA, April 27-29, 2017*, N. V. Chawla and W. Wang, Eds. Society for Industrial and Applied Mathematics, 2017, pp. 207–215. [Online]. Available: https://doi.org/10.1137/1.9781611974973.24

[15] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, vol. 10, no. 8. Soviet Union, 1966, pp. 707–710.

[16] D. Müllner, "Modern hierarchical, agglomerative clustering algorithms," *Computing Research Repository (CoRR)*, vol. abs/1109.2378, 2011. [Online]. Available: http://arxiv.org/abs/1109.2378

[17] F. E. Maranzana, "On the location of supply points to minimize transportation costs," *IBM Systems Journal*, vol. 2, no. 2, pp. 129–135, 1963. [Online]. Available: https://doi.org/10.1147/sj.22.0129

[18] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA*, E. Simoudis, J. Han, and U. M. Fayyad, Eds. Association for the Advancement of Artificial Intelligence, 1996, pp. 226–231. [Online]. Available: http://www.aaai.org/Library/KDD/1996/kdd96-037.php

[19] J. B. Kruskal, "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis," *Psychometrika*, vol. 29, no. 1, pp. 1–27, 1964.

[20] Y. Liu, Z. Li, H. Xiong, X. Gao, and J. Wu, "Understanding of internal clustering validation measures," in *ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December 2010*, G. I. Webb, B. Liu, C. Zhang, D. Gunopulos, and X. Wu, Eds. Institute of Electrical and Electronics Engineers, 2010, pp. 911–916. [Online]. Available: https://doi.org/10.1109/ICDM.2010.35

[21] V. Wenz, A. Kesper, and G. Taentzer, "Tool implementation," https://github.com/Project-KONDA/data-value-clustering.

[22] R. A. Wagner and M. J. Fischer, "The string-to-string correction problem," *Journal of the ACM*, vol. 21, no. 1, pp. 168–173, 1974. [Online]. Available: https://doi.org/10.1145/321796.321811

[23] S. K. Lam, A. Pitrou, and S. Seibert, "Numba: A llvm-based python jit compiler," in *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, ser. LLVM '15. New York, NY, USA: Association for Computing Machinery, 2015. [Online]. Available: https://doi.org/10.1145/2833157.2833162

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[25] "scikit-learn-extra documentation — scikit-learn-extra 0.1.0b2 documentation." [Online]. Available: https://scikit-learn-extra.readthedocs.io/en/latest/index.html

[26] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental algorithms for scientific computing in python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[27] "tkinter — Python interface to Tcl/Tk — Python 3.9.2 documentation." [Online]. Available: https://docs.python.org/3/library/tkinter.html

[28] V. Wenz, A. Kesper, and G. Taentzer, "Excel sheets containing clusterings used for the evaluation," https://github.com/Project-KONDA/data-value-clustering/tree/mde-intelligence/DataValueClustering/experiments/exports/study.

[29] ISO 8601-1:2019, "Date and time — representations for information interchange," International Organization for Standardization, Geneva, CH, Standard, Feb. 2019.

[30] E. Coburn, R. Light, G. McKenna, R. Stein, and A. Vitzthum, "LIDO v1.1 Public Beta (Lightweight Information Describing Objects)." [Online]. Available: http://www.lido-schema.org/schema/v1.1/lido-v1.1-public-beta.xsd

[31] M. A. Pellegrino, L. Postiglione, and V. Scarano, "Detecting data accuracy issues in textual geographical data by a clustering-based approach," in *CODS-COMAD 2021: 8th ACM IKDD CODS and 26th COMAD, Virtual Event, Bangalore, India, January 2-4, 2021*, J. R. Haritsa, S. Roy, M. Gupta, S. Mehrotra, B. V. Srinivasan, and Y. Simmhan, Eds. Association for Computing Machinery, 2021, pp. 208–212. [Online]. Available: https://doi.org/10.1145/3430984.3431031

[32] L. Ciszak, "Application of clustering and association methods in data cleaning," in *2008 International Multiconference on Computer Science and Information Technology*, 2008, pp. 97–103.

[33] J. C. French, A. L. Powell, and E. Schulman, "Using clustering strategies for creating authority files," *Journal of the Association for Information Science and Technology*, vol. 51, no. 8, pp. 774–786, 2000.

[34] E. Sauleau, J. Paumier, and A. Buemi, "Medical record linkage in health information systems by approximate string matching and clustering," *BMC Medical Informatics and Decision Making*, vol. 5, p. 32, 2005. [Online]. Available: https://doi.org/10.1186/1472-6947-5-32

[35] B. T. Dai, N. Koudas, B. C. Ooi, D. Srivastava, and S. Venkatasubramanian, "Column heterogeneity as a measure of data quality," in *Proceedings of the First Int'l VLDB Workshop on Clean Databases, CleanDB 2006, September 11, 2006, Seoul, Korea (Co-located with VLDB 2006)*, 2006. [Online]. Available: http://pike.psu.edu/cleandb06/papers/CameraReady_111.pdf

[36] B. Yi, W. Hua, and S. Sadiq, "A pattern-based framework for addressing data representational inconsistency," in *Databases Theory and Applications*, M. A. Cheema, W. Zhang, and L. Chang, Eds. Cham: Springer, 2016, pp. 395–406.

[37] A. Algergawy, M. Mesiti, R. Nayak, and G. Saake, "Xml data clustering: An overview," *ACM Computing Surveys*, vol. 43, no. 4, Oct. 2011. [Online]. Available: https://doi.org/10.1145/1978802.1978804

[38] G. Hinkel, M. E. Kramer, E. Burger, M. Strittmatter, and L. Happe, "An empirical study on the perception of metamodel quality," in *MODELSWARD 2016 - Proceedings of the 4rd International Conference on Model-Driven Engineering and Software Development, Rome, Italy, 19-21 February, 2016*, S. Hammoudi, L. F. Pires, B. Selic, and P. Desfray, Eds. Science and Technology Publications, 2016, pp. 145–152. [Online]. Available: https://doi.org/10.5220/0005632001450152

[39] L. Bettini, D. Di Ruscio, L. Iovino, and A. Pierantonio, "Quality-driven detection and resolution of metamodel smells," *IEEE Access*, vol. 7, pp. 16 364–16 376, 2019.

[40] T. Arendt and G. Taentzer, "A tool environment for quality assurance based on the eclipse modeling framework," *Automated Software Engineering*, vol. 20, no. 2, pp. 141–184, 2013. [Online]. Available: https://doi.org/10.1007/s10515-012-0114-7

[41] J. J. López-Fernández, E. Guerra, and J. de Lara, "Meta-model validation and verification with metabest," in *ACM/IEEE International Conference on Automated Software Engineering, ASE '14, Vasteras, Sweden - September 15 - 19, 2014*, I. Crnkovic, M. Chechik, and P. Grünbacher, Eds. Association for Computing Machinery, 2014, pp. 831–834. [Online]. Available: https://doi.org/10.1145/2642937.2648617

[42] J. J. López-Fernández, J. S. Cuadrado, E. Guerra, and J. de Lara, "Example-driven meta-model development," *Software and Systems Modeling*, vol. 14, no. 4, pp. 1323–1347, 2015. [Online]. Available: https://doi.org/10.1007/s10270-013-0392-y

[43] R. Agrawal, T. Imielinski, and A. N. Swami, "Mining association rules between sets of items in large databases," in *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, DC, USA, May 26-28, 1993*, P. Buneman and S. Jajodia, Eds. Association for Computing Machinery, 1993, pp. 207–216. [Online]. Available: https://doi.org/10.1145/170035.170072

[44] V. Wenz, A. Kesper, and G. Taentzer, "Configurations used for the evaluation," https://github.com/Project-KONDA/data-value-clustering/tree/mde-intelligence/DataValueClustering/experiments/evaluation.

[45] G. Navarro, "A guided tour to approximate string matching," *ACM Computing Surveys*, vol. 33, no. 1, pp. 31–88, 2001. [Online]. Available: https://doi.org/10.1145/375360.375365

[46] M. Ankerst, M. M. Breunig, H. Kriegel, and J. Sander, "OPTICS: ordering points to identify the clustering structure," in *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA*, A. Delis, C. Faloutsos, and S. Ghandeharizadeh, Eds. Association for Computing Machinery, 1999, pp. 49–60. [Online]. Available: https://doi.org/10.1145/304182.304187

[47] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, 2007. [Online]. Available: https://science.sciencemag.org/content/315/5814/972

[48] J. Shi and J. Malik, "Normalized cuts and image segmentation," in *1997 Conference on Computer Vision and Pattern Recognition (CVPR '97), June 17-19, 1997, San Juan, Puerto Rico*. Institute of Electrical and Electronics Engineers, 1997, pp. 731–737. [Online]. Available: https://doi.org/10.1109/CVPR.1997.609407

TABLE II
BASIC EDIT DISTANCE WEIGHT MATRIX OF ARTIST NAMES

	Letters	- and '	Digits	Space	Special	Comma	Other
-	1	1	10	15	100	200	1

APPENDIX A
EVALUATION

In the following, we present configuration details of the evaluation of our approach presented in Sec. V as well as the Excel sheets we used to present clusterings to domain experts.

A. Configurations

The configuration settings that were used to calculate the four clusterings underlying our evaluation are specified at [44] and outlined in the following. For each clustering, we outline the data value abstraction, the weights used for calculating the edit distance and the configured clustering algorithm applied.

1) Artist Name: The documentation of this field is outlined in Sec. II.

a) Data Value Abstraction: Sequences of upper or lower case letters separated by space were mapped to the same letter as they probably all represent sequences of names, such as first name followed by middle names. All digits were mapped to the same digit as the concrete digits do not alter the values' meaning significantly but the length of digit sequences may hint at the kind of information that is encoded. Special characters were preserved as they often encode special meaning, which we are interested in. Redundant abstracted values were removed.

b) Distance Calculation: The basic edit distance was used with the insertion and deletion weights given in Table II. Letters (representing sequences of upper or lower case letters separated by space), hyphens and apostrophes are expected in the abstracted artist name values, thus weighted low. As digits and additional blank spaces cause some variation in meaning, they are weighted higher. We actually do not expect other special characters in artist names, but if they occur, they often encode special meaning, which we are interested in revealing. Therefore, they are weighted much higher. We expect that, analogous to typical spelling variants, for some artist names first, middle and last names are given in this order while for others the last name may be given first, followed by a comma followed by first and middle name. Commas are highest since we wanted to separate these variants in the clustering. The column "other" includes mostly individual letters not included in the 26 letters of the basic Latin alphabet, such as "Ł", thus the weight is low.

c) Clustering: We used hierarchical clustering with complete linkage and applied a distance threshold of 700.

2) Dating: The documentation of this field is outlined in Sec. II.

a) Data Value Abstraction: All upper case letters were mapped to their lower case equivalent as capitalisation does not seem to significantly impact the meaning of the dating values. As we do not expect a wide variety in the textual components of the values, we do not abstract from concrete letters or the length of letter sequences. All sequences of digits were mapped to the same digit as they represent similar meanings. Special characters were preserved as they often encode special meaning, which we are interested in. Redundant abstracted values were removed.

TABLE III
LEVENSHTEIN DISTANCE WEIGHT MATRIX OF DATINGS

	-	Digits	Other
-	0	1	4
Digits	1	2	4
Other	4	4	4

TABLE IV
LEVENSHTEIN DISTANCE WEIGHT MATRIX OF MEASUREMENT UNITS

	-	Digits	Letters	Special
-	-	2	1	2
Digits	2	0	3	4
Letters	1	3	1	3
Special	2	4	3	2

b) Distance Calculation: The Levenshtein distance was used with the weights given in Table III. Mainly, digits (representing sequences of digits) are expected and considered quite similar to each other. Other characters, i.e. letters and special characters, are weighted higher as they are expected to cause much greater dissimilarity in the meaning of the dating values. Because we do not see a difference between the insertion or deletion and substitution of characters, we unified the weights all to 4. In contrast to the basic edit distance, this refinement reduces the dissimilarity between values of the same length. For example, this reduces the dissimilarity between values like `x` and `y`, which both were used in this data field as placeholder for unknown dates. It further reduces the dissimilarity between `1895/1902` and `1908–1909`, where `/` represents a point in time between two dates and `–` is used to represent periods of time. Due to other more decisive anomalies expected in this data field, we decided that such differences in values should be weighted lower.

c) Clustering: We used hierarchical clustering with complete linkage and applied a maximum cluster number of 25.

3) Measurement Unit: The documentation of this field is outlined in Sec. II.

a) Data Value Abstraction: Letters were preserved. All sequences of digits were mapped to the same digit. Those separated by a comma were all mapped to another digit. Special characters were preserved as they often encode special meaning, which we are interested in. Redundant abstracted values were removed. Details are explained in Sec. C-1.

b) Distance Calculation: The Levenshtein distance was used with the weights given in Table IV. Details are explained in Sec. C-2.

TABLE V
BASIC EDIT DISTANCE WEIGHT MATRIX OF ATTRIBUTION QUALIFIERS

	Letters	Space	Digit	Special
-	1	20	30	100

c) Clustering: We used hierarchical clustering with complete linkage and applied a distance threshold of 3.5.

4) Attribution Qualifier: The LIDO documentation describes the field as follows. *"Definition: A qualifier used when the attribution is uncertain, is in dispute, when there is more than one actor, when there is a former attribution, or when the attribution otherwise requires explanation. How to record: Example values: attributed to, studio of, workshop of, atelier of, office of, assistant of, associate of, pupil of, follower of, school of, circle of, style of, after copyist of, manner of..."*

a) Data Value Abstraction: All lower and upper case letters were mapped to the same letter as we expect a variety of letters in these values, but based on the documentation we expect entries of similar length. All digits were mapped to the same digit as the concrete digit does not matter in this context. Special characters were preserved as they often encode special meaning, which we are interested in. Redundant abstracted values were removed.

b) Distance Calculation: The basic edit distance was used with the insertion and deletion weights given in Table V. Mainly, letters are expected, thus weighted low. Since blank spaces may imply interesting extensive explanations of the attribution, they are weighted higher. As digits are not expected at all, they are weighted even higher. Since special characters are often used to encode special meaning, such as uncertainty, they are weighted the highest.

c) Clustering: We used hierarchical clustering with complete linkage and applied a distance threshold of 100.

B. Calculated Clusterings

Fig. 5 shows an excerpt of an Excel sheet used to present result clusterings to domain experts. More precisely, it is an excerpt of the Excel sheet showing the clustering of abstracted measurement unit values via representatives. In the second row the number of original values in each cluster is indicated. The number of abstracted values per cluster is indicated in the third row. Next to each representative value, the number of original values it represents is shown.

All four Excel files can be found at [28].

APPENDIX B
TOOL SUPPORT

In this section, we present some of the views of our graphical user interface concerned with configuring the data value clustering process. Our goal is to provide an interface with which the numerous parameters can be configured intuitively and easily by domain experts based on their domain knowledge. The interface shall require as little technical understanding of the clustering process as possible. The views are briefly described in Sec. IV-B, details are provided in the following.

A. Configuring the Abstraction of Data Values

Fig. 6 shows the interactive view for configuring the abstraction step. On the left-hand side, it shows a binary response questionnaire on the importance of several syntactical features of the data values. These features are related to the abstraction rules discussed in Sec. C-1. Per option, a tool tip showing an explanation and an example is provided. For demonstrating the configuration, the abstraction of the first 100 original values is dynamically visualised on the right-hand side. Per abstracted value, the corresponding original values are listed. The screenshot shows the configuration of the abstraction for the measurement unit values, discussed in Sec. C-1.

B. Configuring the Calculation of Distances

For specifying the weights of the weighted Levenshtein distance we provide two input methods. The first one requires the domain expert to input the weights directly as numbers. The corresponding view is presented in Fig. 7. It shows the configuration for the running example on measurement units explained in Sec. C-2. The view is structured analogously to the distance weight matrices presented, for example in Table VI. On the left-hand side, the characters belonging to the same group are enumerated in a row. The first column and row contain the weights of deleting and adding characters. The last column and row represent all other characters not listed before. The matrix must be symmetric to fulfil the symmetry axiom for metrics. Thus, the entry in a field is automatically copied to the symmetrically corresponding field.

An alternative is to input distance weights via a graphical view, which we call *blob view*. An example blob view is presented in Fig. 8. It configures distance weights of the running example on measurement units. Groups of characters are presented as graphical objects, called *blobs*. They correspond to the columns and rows of the matrix in Fig. 7. The values in the matrix are formed as follows: The graphical distance between two blobs is interpreted as the weight for substituting characters of corresponding groups. To configure additions and deletions we use an additional blob, the small blue blob labelled with an X. For configuring the weights, the user can move the blobs on the 2D canvas using drag and drop. The weight for substitutions within a group is represented by the size of the corresponding blob. The user can modify the size using the mouse wheel while hovering over the blob.

We are aware that this input does not give the full expressiveness of the matrix input as it would require $n - 1$ dimensions where n is the number of blobs. In contrast, this view is limited to a 2D canvas. Our intention is to provide an intuitive way to configuring distance weights. It is a potentially more relatable visualisation compared to entering numerical values into a $n * n$ matrix. A comprehensive user study is pending.

	Cluster 1		Cluster 2		Cluster 3		Cluster 4		Cluster 5		Cluster 6		Cluster 7		Cluster 8		Cluster 9		Cluster 10	
#original	64470		21925		604		18		14		6		2		1		1		1	
#abstracted	3		1		3		4		4		3		1		1		1		1	
	cm	63482	-	21925	-10,5 cm	406	, 5 cm	13	x 55 cm	5	cm / 120 cm	4	? cm	2	x 350 x 411 x	1	/25,5 cm	1	cm cm	1
	m	943			-215 cm	196	- 45 cm	2	x 11,5 cm	4	cm x 60 cm	1								
	mm	45			-4,6cm	2	+ 6,7 cm	2	x 3 m	3	cm / 38,5 cm	1								
							- 1,2 cm	1	x 9,8 m	2										

Fig. 5. Excerpt of an Excel sheet showing a clustering of abstracted measurement unit values via representatives.

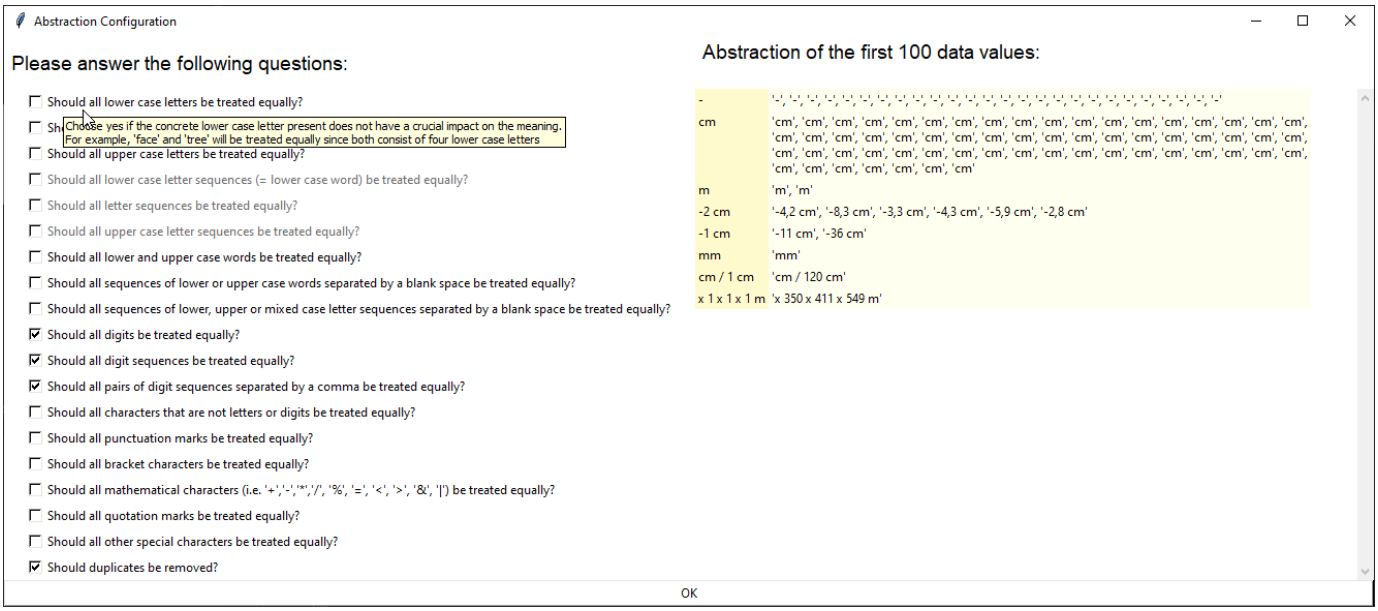

Fig. 6. Configuration of the data value abstraction.

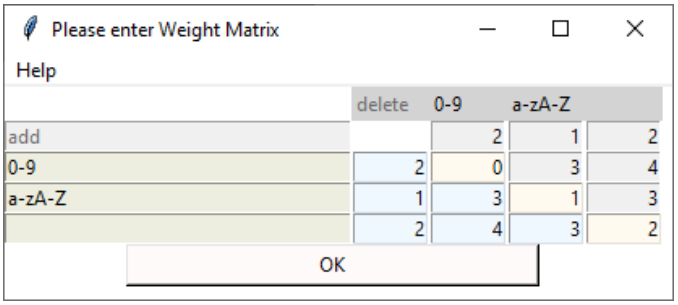

Fig. 7. Configuration of the distance weights via the matrix view.

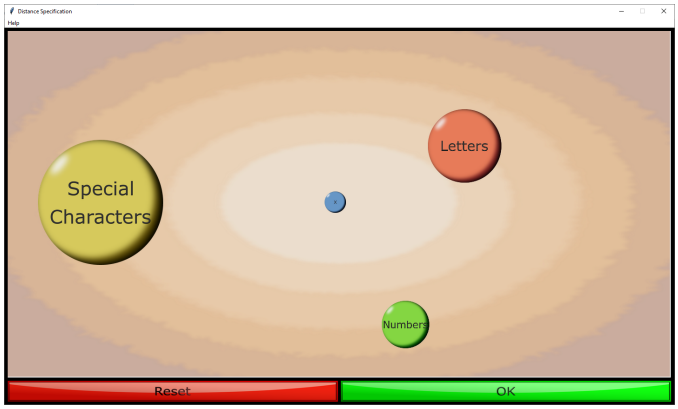

Fig. 8. Configuration of the distance weights via the blob view.

C. Configuring the Clustering Algorithm

All the clustering algorithms come with many parameters. We implemented a modularized view to configure these parameters as shown in Fig. 9 for the hierarchical clustering of abstracted measurement unit values. Each module contains the name of a parameter and an explanation. We distinguish between three kinds of parameters: boolean ones, numerical ones and enumerations. Each module contains a checkbox, a slider or an enumeration with radio buttons, respectively. Appropriate minimum and maximum values are automatically assigned to the sliders for numerical parameters. For enumerations, tooltips are provided for each option.

Because we present the parameters of the clustering algorithms to the interface, the modules also mimic their dependencies. For example, between `n_clusters` and `distance_threshold` in Fig. 9, there is an alternating dependency: only one of the parameters is required. The parameter `depth` is enabled only if the option "inconsistent" is chosen for the parameter `criterion`.

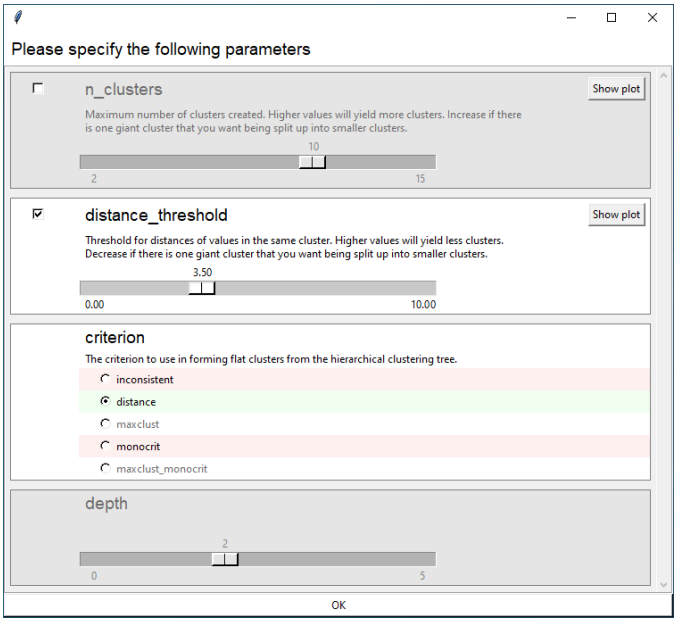

Fig. 9. Configuration of hierarchical clustering.

APPENDIX C
DATA VALUE CLUSTERING

In this section, we present the core algorithm of our approach in detail. It is visualised at the bottom of Fig. 10. The inputs of this algorithm are a set of data values extracted from a database and a configuration of each of the algorithm steps. The algorithm for data value clustering consists of the following steps: (1) an abstraction from the original data values, (2) a calculation of a distance matrix containing pairwise numerical distances between the abstracted data values and (3) a clustering of the abstracted values based on the calculated distances. The main output of the algorithm is a clustering of the abstracted data values. Additionally, the mapping between original and abstracted values and the distance matrix are outputs. In the following, these three steps and their configuration are discussed in detail. For each step, we also explain the configuration for the running example, which we specified based on our domain knowledge as outlined in Sec. V-A.

1) Data Value Abstraction: The first step is an abstraction from the original data values. Only syntactical features that, according to domain experts, are of interest for clustering are maintained. The inputs of this step are a set of data values and a configuration of the abstraction function. The output is a mapping between the original and the abstracted values.

In contrast to existing approaches to string clustering that detect, for example, misspellings, typos, spelling variants and abbreviations [31], [32], [33], we do not intend to detect minor variations in the syntax of data values that have exactly the same meaning. Instead, the goal is to provide an *overview of significant differences in the syntax* of all original data values as syntactical heterogeneity may indicate quality problems of the data model. Thus, based on domain knowledge, we abstract from the original values to remove irrelevant syntactical details, such as the length of digit sequences.

In our realisation of the abstraction step, the original data values are mapped to a smaller set of shorter values. This has a positive impact on the performance of the distance calculation and the clustering performed subsequently. An abstraction is defined by a *set of rules dependent on the configuration*. These rules are based on two observations concerning data values. First, due to their significant difference in meaning, three groups of characters can be distinguished on the top level, namely letters, digits and special characters. Second, there are three interesting levels of abstraction from concrete data values: (1) abstracting from a concrete character of a specific group (considering, e.g., “a” equivalent to “b” and “1” to “2”), (2) abstracting from the length of a sequence of characters of a specific group (considering, e.g., “a” equivalent to “painting” and “1” to “245”), and (3) abstracting from the length of a sequence of characters of a specific group containing some separators (considering, e.g., “a” equivalent to “the last supper” and “1” to “23.7”). These rules roughly correspond to expected data formats in data models while allowing finer distinctions. For example, integers and floats can be mapped to (2) and (3) directly. In other cases, the rules correspond to data formats with additional constraints. Selected abstraction rules are applied to each of the original data values. They replace each match of a specific regular expression by some character. There are additional rules for replacing an upper case letter by the equivalent lower case letter and for removing duplicate values after all other rules have been applied. Further abstraction rules corresponding to typical data formats such as dates may be added in the future.

It depends on the data field analysed which syntactical features are of interest. That’s why the abstraction must be *configured based on domain knowledge*. The configuration typically depends on expectations about the syntax of the data values. It further depends on the kinds of syntax variations of the values that cause significant variations in their meaning. Ultimately, the configuration determines which abstraction rules are applied.

For the *running example*, we configured the abstraction as follows: We expect that a measurement unit value consists of a few letters representing an abbreviation of a measurement unit, such as `cm`. To identify quality problems in the data model LIDO, we need to find out what kinds of values are included in the data that do not suit our expectation and alter the values’ meaning significantly. Hence, interesting features for clustering are especially non-letter characters (i.e. special characters and digits), longer sequences of letters and unexpected combinations of letters. The kinds of digits and the length of digit sequences (possibly separated by a decimal separator), however, are not considered decisive for the meaning of the values. Thus, sequences of digits are transformed into a single digit, sequences of digits containing a decimal separator are transformed into another single digit, while letters and special characters are preserved. When applied to measurement unit values, 179 values given originally are abstracted to 22 values.

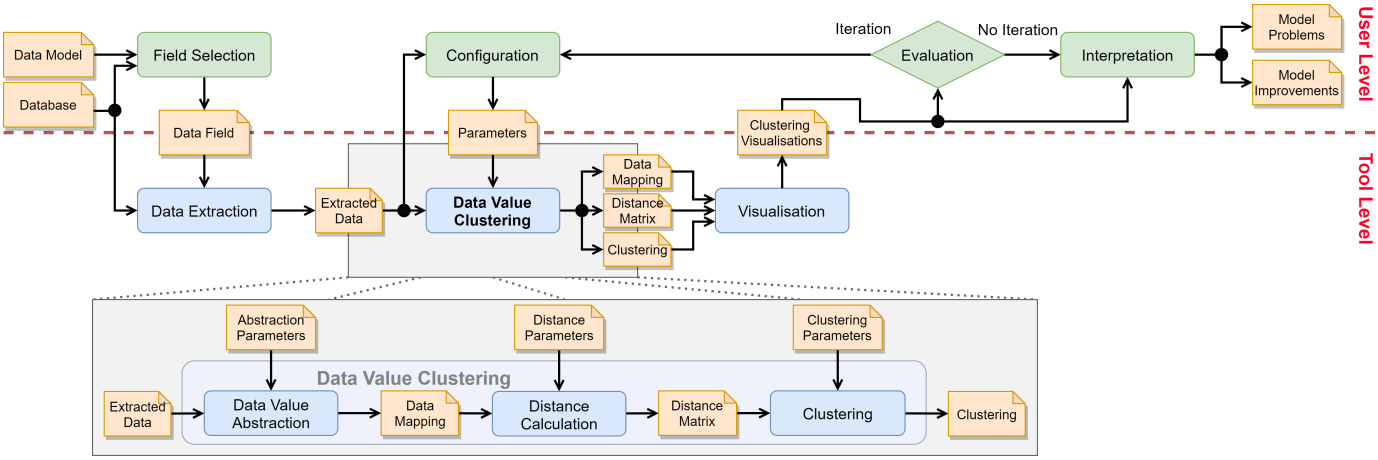

Fig. 10. Detailed workflow of the approach to detecting quality problems in data models by clustering heterogeneous data values.

The columns of Fig. 2 represent several groups of original values that were mapped to the same abstracted value.

If the abstraction produces a manageable amount of values, this may be sufficient to provide an overview of the values' diverse syntax. Otherwise, our approach suggests clustering the abstracted values to produce a manageable number of clusters containing similar abstracted values. As a prerequisite, distances between abstracted data values have to be computed.

2) Distance Calculation: The next step is the calculation of a distance matrix containing pairwise distances (i.e., dissimilarities) between all the abstracted values, again depending on domain knowledge. The inputs of the distance calculation are a set of abstracted data values and a configuration of the distance function. Its output is a distance matrix. Note that the distances between all original values that were mapped to the same abstracted value are considered to be zero.

Since our ultimate goal is to group all the abstracted values by syntactic similarity, the amount of similarity must be quantified first. For example, special characters are often used to encode special meaning. Hence, they serve as operators, and should cause high dissimilarity.

String dissimilarity is typically measured via *edit distances* allowing different kinds of string operations [45]. Having applied the approach to cultural heritage data, we achieved reasonable results with the following edit distances: the basic edit distance that allows insertions and deletions of string characters only, and the Levenshtein distance [15], which additionally allows substitutions. They may be accomplished with further distances when needed.

The dissimilarity between two values depends on the data field analysed. In a field representing the name of a person, for example, the insertion of an additional letter does not really cause dissimilarity whereas for a field representing the height of an object it does. Therefore, we use *configurable weights* for each edit operation (namely insertion, deletion and substitution) applied to each possible character. Bear in mind that a character of an abstracted value, depending on the abstraction rules applied, may represent a character of a specific group or a sequence of certain characters. Note that only the ratios of weights are relevant.

Domain experts must configure the weights based on their *domain knowledge*. Typically, vague relations between weights can be derived from domain knowledge. In general, edit operations of unexpected characters should be weighted higher than those of expected ones as they may indicate quality problems in the data model. Additionally, the more influence edit operations of certain characters have on the meaning of data values, the higher those weights should be. However, there remains some leeway in determining concrete values that satisfy these relations. Therefore, we support multiple iterations to experiment with different configurations.

The basic edit distance corresponds to the Levenshtein distance if the weights of each substitution equal the sum of the weights of the corresponding deletion and insertion. In this case, only the weights for deletions and insertions must be specified. Hence, it can serve as a starting point. For further refinement, lower weights for character substitutions may be specified if different characters are considered quite similar but not equivalent (e.g. different kinds of quotation marks) and the length of values is considered decisive for their meaning. Then substitutions of those characters have less impact on the meaning than their deletions and insertions in other contexts.

The chosen weights for the *running example* are presented in Table VI. The first column and row contain the weights for character deletions and insertions, respectively. The other cells show the weights for substitutions of corresponding characters.

We chose the weights for deleting and inserting characters according to our expectation concerning the syntax of the values. Digits and special characters are unexpected, while letters are expected. Therefore, the weight for letters is lower than those for digits and special characters.

The numbers that are not on the diagonal represent substitutions of characters of different types. We have no reason to assume that such substitutions have less impact on the meaning than deletions and insertions of corresponding characters in other contexts. Thus, we specified the weights of such substi-

TABLE VI
DISTANCE WEIGHT MATRIX FOR MEASUREMENT UNITS

	-	Digits	Letters	Special
-	-	2	1	2
Digits	2	0	3	4
Letters	1	3	1	3
Special	2	4	3	2

tutions as the sum of deleting and inserting the characters.

After the abstraction, only two different digits are left, which represent integers and decimal numbers. Since any numbers can be considered equivalent here, we set the weight for substituting one digit by the other to zero. We consider different letters as non-equivalent but as quite similar to each. Further, we expect abbreviations of measurement units, thus values of similar (short) length. Hence, substitutions of letters have less impact on the meaning than deletions and insertions of letters in other contexts. So we chose the weight of substituting one letter by another equal to the weights of deleting and inserting a letter, respectively. Thus, for example, the values `cm`, `mm` and `m` are equally dissimilar to each other. The same applies to special characters. The distances calculated with this configuration are visualised in Fig. 3.

3) Clustering: The final step of the algorithm consists in clustering a set of abstracted data values. The inputs of this step are a set of abstract data values, a distance matrix and a configured clustering algorithm. Its output is a clustering of the abstracted data values.

We can only use clustering algorithms that operate on string distances. The set of suitable algorithms includes hierarchical clustering [16], k-medoids [17], DBSCAN [18], OPTICS [46], affinity propagation [47], and spectral clustering [48].

The domain expert has to select one of these algorithms and *configure* its parameters dependent on the data field analysed. Aspects that may influence the selection of an appropriate clustering algorithm and the parameter configuration are: determinism of the computed clustering, expected heterogeneity of cluster density, expected cluster shapes (such as chains and spheres), expected heterogeneity of cluster sizes, robustness to outliers and desired speed of execution. The mapping between domain knowledge and these aspects is complex. We provide a setting that allows experimenting with a variety of clustering algorithms. It is up to future research to investigate which clustering algorithms and parameter settings are most suitable for detecting quality problems in data models and how this depends on the kind of data considered.

For our *running example*, we chose hierarchical clustering with complete linkage to cluster abstracted measurement units and set the distance threshold to 3.5. This yielded 10 clusters, which are visualised in Fig. 3 and partly presented in Fig. 2.