Using Ontologies Providing Domain Knowledge for Data Quality Management

Stefan Brüggemann and Fabian Grüning

Abstract. Several data quality management (DQM) tasks like duplicate detection or consistency checking depend on domain specific knowledge. Many DQM approaches have potential for bringing together domain knowledge and DQM metadata. We provide an approach which uses this knowledge modeled in ontologies instead of aquiring that knowledge by cost-intensive interviews with domain-experts. These ontologies can directly be annotated with DQM specific metadata. With our approach a synergy effect can be achieved when modeling a domain ontology, e.g. for defining a shared vocabulary for improved interoperability, and performing DQM. We present five DQM applications which directly use knowledge provided by domain ontologies. These applications use the ontology structure itself to provide correction suggestions for invalid data, identify duplicates, and to store data quality annotations at schema and instance level.

1 Motivation and Goal

Data Quality Management (DQM) approaches report on the quality of data measured by defined data quality dimensions and, if desired, correct data in databases. DQM relies on domain knowledge for detecting and possibly correcting erroneous data, as data without its definition cannot be interpreted as information and is therefore meaningless. On the one hand, DQM approaches like [9] or [1] define phases where domain experts provide their knowledge for further utilization in the DQM process. On the other hand, there are domain ontologies, i.e. formal specifications of

Stefan Brüggemann
OFFIS, Germany
e-mail: brueggemann@offis.de

Fabian Grüning
University of Oldenburg, Germany
e-mail: fabian.gruening@informatik.uni-oldenburg.de

S. Schaffert et al. (Eds.): Networked Knowledge - Networked Media, SCI 221, pp. 187–203.
springerlink.com

conceptualizations of certain domains of interest, that already provide such knowledge but remain unused in the DQM context. Our contribution is to directly use the knowledge provided by domain ontologies in the DQM context in order to improve the DQM's outcome.

This contribution is structured as follows: Firstly, we will discuss work related to this topic and secondly describe our approach which directly uses the knowledge provided by domain ontologies in the context of DQM in detail by presenting five applications, namely consistency checking, proactive management of consistency constraints, duplicate detection, metadata management, and semantic domain modeling. Finally, we will draw some conclusions and point out further work on this topic.

2 Related Work

Little work has been done on the field of using ontologies for DQM. Existing approaches can be divided into two major classes:

The first application of ontologies in the case of DQM is management of data quality problems and methods. The OntoClean Framework has been introduced in [20]. It provides a template for performing data cleaning consisting of several steps like building an ontology, translating user goals for data cleaning into the ontology query language, and selecting data cleaning algorithms.

The second application of ontologies is the use of domain ontologies. They provide domain specific knowledge needed to validate and clean data. This allows for detecting data problems, which could not be found without this knowledge. To the best of our knowledge, only [15] and [13] use domain ontologies in this way.

We extend these approaches by annotating domain ontologies with DQM-specific metadata which we show in the following section by presenting five DQM-applications that further include the usage of algorithms of the data mining domain.

3 Multiple Utilizations of Domain Ontologies for DQM

To show the advantages of using ontologies in the context of DQM and emphasize the usefulness of our approach to improve the outcome of DQM we present five applications of domain ontologies in that context: consistency checking, proactive management of consistency constraints, duplicate detection, metadata management, and semantic domain modeling.

3.1 Context-Sensitive Inconsistency-Detection with Ontologies

Data cleaning is often performed when data has to be integrated into a database. Data cleaning consists of the detection and removal of errors and inconsistencies from data [16]. We use domain specific knowledge to detect inconsistencies. Consistency is defined as the abidance of semantic rules. These rules can be described with integrity

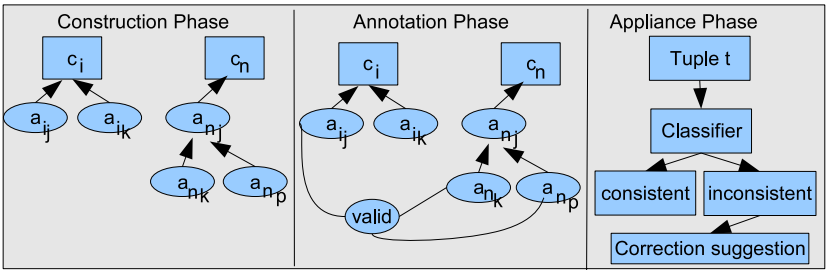

Fig. 1 Overview of an inconsistency detection algorithm using a domain ontology

constraints in relational databases for attributes on schema level. On instance level, consistency is being defined as the correct combination of attribute values. A tuple is consistent when the values from each attribute are valid in combination.

We now provide an algorithm and a data model for consistency checking.

3.1.1 Basic Idea

Figure 1 shows a graphical representation of the consistency checking algorithm. The algorithm consists of three phases. In the construction phase a domain ontology is being created. It can be learned from an existing database, created manually, or already existing ontologies can be used. The expressive power of OWL (Web Ontology Language) enables a generic semi-automatic ontology construction approach. The domain ontology can almost completely be used for DQM, only tuples have to be labeled as valid in the annotation phase. In the appliance phase tuples are being identified as being consistent or inconsistent. When an inconsistency is being detected, a correction suggestion is made. The ontology structure is used to correct invalid tuples. Other valid tuples are searched and characterized as possible corrections. The suggestions are ranked using the distance between the valid and invalid tuples. The advantage over the statistical edit/imputation-approach presented by [6] is the usage of the context of invalid attributes for correction. The statistical approach replaces invalid tuples with randomly chosen values, whilst our approach suggests context-sensitive corrections changing as few attributes as possible.

3.1.2 Data Model Used for Consistency Checking

A relation schema $R = (A_1, .., A_n)$ is defined as a list of attributes A_1, .., A_n. Each attribute A_i belongs to a domain $dom(A_i)$. Each domain $dom(A_i)$ defines a non-empty set of valid values. A relation r of R is a set of n-tuples $r = t_1, .., t_m$. Each tuple t_i is a set of values $t_i = (v_{i_1}, .., v_{i_n})$ with $v_{i_j} \in dom(A_j)$.

In the simplest case a tuple t_i is valid if and only if $\forall 1 \leq j \leq n : v_{i_j} \in dom(A_j)$. According to our definition, a tuple is consistent if it is valid and all v_{i_j} are combined correctly.

When validating a tuple t, using only the domain $dom(A_j)$ doesn't enable to identify inconsistencies because combinations cannot be checked. Therefore an ontology

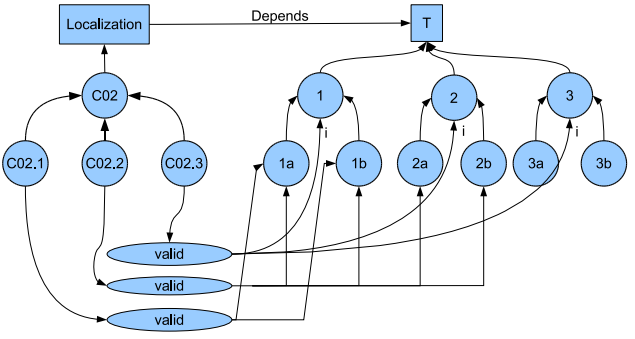

Fig. 2 Ontology containing concepts Localization and T with individuals

is being built containing all values $a_{i_k} \in dom(A_i)$ of each domain with $k = |dom(A_i)|$. Furthermore, domains often contain hierarchical, multidimensional, or other complex structures. These can be respected in an ontological structure.

An ontology consists of a concept C_i for each domain $dom(A_i)$. Attributes a_i are defined as individuals of C_i. They are arranged using "moreSpecificThan" properties to enable modeling complex structures.

Dependencies are defined between concepts. For instance, there is often no semantic dependency between the attributes "id" and "surname". Instead, in oncology, several constraints exist when combining "localization"-values and "T", "N", and "M"-values from the TNM-classification (tumour, node, metastasis) scheme [12]. Concepts have properties "valid" and "invalid" to combine attributes of different concepts and to label them as valid or invalid.

3.1.3 Example

We now provide an example from tumour classification in the cancer registry of lower saxony in Germany. Figure 2 shows an ontology containing the concept Localization, which depends on the concept T. The individuals "C02", "C02.1", "C02.2", and "C02.3" describe malignant neoplasms of the tongue, where the "C02.x" (tip, bottom, 2/3 of front) individuals are more specific than "C02". The property "moreSpecificThan" is hidden due to readability. The three "valid" nodes are introduced as blank nodes and used to describe the following three consistency rules: "C02.1" is only valid with "T"-values "1a" and "1b". "T"-values lower than 2 describe tumour-sizes lower than 2cm. "C02.2" is valid with "T"-Values "1a", "1b", "2a", and "2b". "2x"-values are sizes between 2cm and 5cm. "T"-values larger than "2" describe tumour sizes larger than 5cm. "C02.3" is valid with "T"-values "1", "2", and "3". Specifying these connections with "i" defines these as inheriting connections. These connections are inherited to the children of "1", "2", and "3". Therefore the more specific values of "1", "2", and "3", namely "1a", "1b", "2a", "2b", "3a", and "3b", are also valid with "C02.3".

For instance, the tuples $< C02.3, 1 >$, $< C02.3, 3 >$, and $< C02.1, 1a >$ with the structure $< Localization, T >$ can be resolved as valid. The tuple $< C02.1, 3 >$

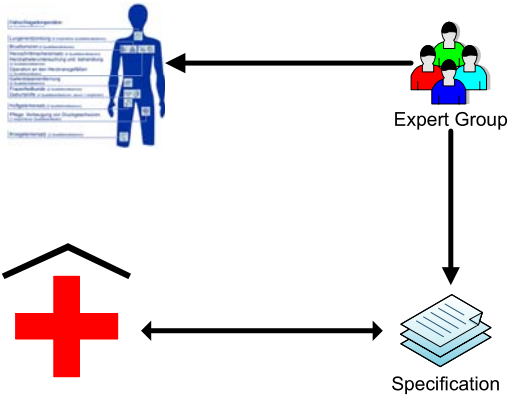

Fig. 3 Medical Documentation: Expert Group defines measures and publishes them in a specification. Hospitals have to send reports corresponding to this specification.

instead can be identified as invalid, but using the ontological structure, the tuples $< C02.1, 1a >$, $< C02.1, 1b >$, $< C02.3, 3 >$, $< C02.1, 3a >$, and $< C02.1, 3b >$ can be resolved as correction suggestions.

3.2 Proactive Management of Consistency Constraints

As we have seen in the previous section, a domain ontology can be either learned from an existing knowledge base or has to be created manually. We now focus on the latter case and introduce *ProCon*, which is an approach for proactive management of consistency constraints. It is described in detail in [5]. ProCon has been introduced in a scenario in the German public health sector. A German law defines that the medical quality in public health has to be measured and compared. Therefore the "BQS Bundesgeschäftsstelle Qualitätssicherung gGmbH"[1] was founded in 2000. The BQS manages and coordinates the comparison of german hospitals. Figure 3 shows the BQS-workflow: An expert group, consisting of several medics, defines quality measures for relevant medical sectors like cardiology or oncology. For each measure thresholds are being defined. For instance, when the goal is to ensure good medical quality the fatality rate for a specific operation must not exceed a predefined threshold. These measures are being delivered to hospitals and software vendors in a large document. This document describes the information needs of the BQS. The hospitals have to send quality reports with all requested data periodically.

When abnormalities in the delivered data are identified or thresholds are reached, like the mortality rate for a specific operation is beyond a specific treshold, a so called "structured dialogue" is being entered. In this dialogue the BQS analyzes whether bad data quality, insufficient medical quality, or poor medical documentation is responsible for violating the respective measure. In practice, often bad data

[1] http://www.bqs-online.de

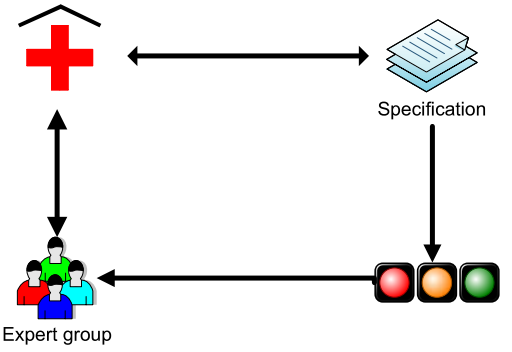

Fig. 4 BQS analyzes reported data and begins a ”structured dialogue” with hospitals when measures are not matched

quality can be made responsible. This results in time-consuming communication about detecting and removing inconsistencies. The structured dialogue is shown in figure 4: Hospitals deliver the data which is requested by the specification. This data is being analyzed for inconsistencies and violations of bounds and thresholds. Based on this analysis the expert groups starts a detailed communication with the hospital, which has delivered the data. The compared data of the hospitals is then being published in quality reports. When structured dialogues with hospitals are being performed, some of their data cannot be considered for these reports. This would result in competitive disadvantages, because the data is being used for patient guides. Patients often choose hospitals with a lot of experience in a desired operation.

The described approach has some shortcomings: Although the described approach is top-down-driven, data quality is not being modeled explicitly on a high level. Information needs are being described on a high level by domain experts. They explicitly define dimensions, measures, and all required data. These domain experts are able to distinct data in valid or invalid. Therefore in ProCon we propose the role of a data quality modeler. This modeler defines attribute value combinations in the modeled information needs as valid or invalid. When hospitals know about these quality constraints, they are able to deliver data with high quality.

Another shortcoming is the technical specification itself. The BQS delivers access databases containing the defined information needs and technical rules. These databases cannot be used directly, but have to be integrated in the information system of each hospital. In our approach we propose to deliver the modeled rules using ontologies. This has the benefit that software vendors and hospitals do not need to develop rule checking software. When data has to be exported to the BQS, it can be instantiated as an instance of the ontology and reasoners are able to check the consistency of the resulting ontology automatically (compare also section 3.5).

We now present a model-driven approach of defining consistency constraints in different scenarios. The described approach fits well in the defined BQS-context, but can be applied to other domains as well. The approach is shown in figure 5. It starts with a metamodel of the domain of discourse (part A in the figure). Our

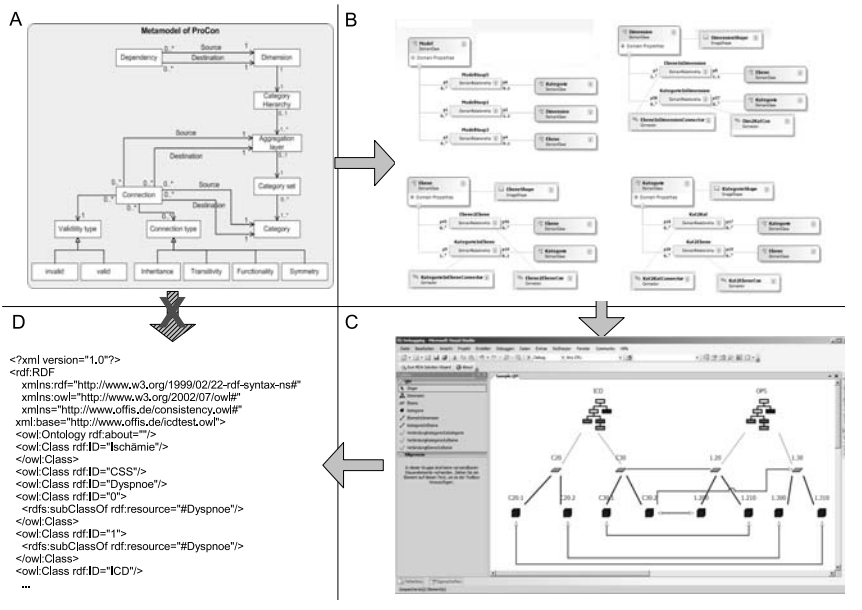

Fig. 5 Transformation steps of a modeled domain of discourse. A metamodel is being converted to a domain specific language. An editor can automatically be generated for this language. This editor can be used to create models that are then being transformed into an ontology.

scenario is based on the multidimensional data model MADEIRA [21], but other models like relational databases or XML-schemata are possible as well. Our model consists of a dimension, which contains a hierarchy of categories. These categories can be combined in aggregation layers.

This data model has been extended with the possibility of defining consistency constraints between dimensions. Therefore dependencies can be created to show where constraints are possible and where not. For instance, there is no dependency between ”age” and ”name”, but between ”localisation” and ”T”, as described in section 3.1. The individuals ”C02”, ”C02.1”, C02.2”, and ”C02.3” for localisation were defined in 3.1.3 and can be identified as categories and aggregation layer.

Connections can be defined between categories and categories, aggregation layer and aggregation layer, and categories and aggregation layer and vice versa. Each connection can be labeled as describing a valid or an invalid attribute value combination. Different connection types can be defined: An inheriting connection describes that the connection is being inherited from an aggregation layer to its children. Transitive connections can be traversed. A functional connection describes that a category or aggregation layer is only valid with the connection destination, and with no other. When such a connection is symmetric, it is modeled that both connection endings can only be connected with the respective other end.

Such a data model can simply be mapped to a domain specific language (DSL) (shown in part B). This DSL contains the same entities and relations as the metamodel. Using the Microsoft DSL tools [14], an editor can be generated automatically. This editor provides all elements defined in the metamodel (shown in part C).

This editor can be used by domain experts to model the information needs and to label combinations as valid or invalid. The graphical representation is a core benefit of this approach, because domain experts do not need to learn to use complex new tools, but can directly use the tool used for modeling the information needs.

With an appropriate transformation the modeled information needs and consistency constraints can be stored as an ontology (shown in part D). Due to the mappings the ontology is an instance of the metamodel, but must not be generated from the metamodel.

The ontology provides several benefits:

- The ontology can be used to check the consistency of the modeled constraints. It can identify contradictions in the model, which can directly be presented to the data quality modeler at design time.
- Data can be checked using this ontology. The ontology can directly be used to identify errors in data by instantiating the data which has to be checked in the ontology. A reasoner can then identify conflicts in the data.
- The ontology can be used as a technical document and can be delivered to software vendors and hospitals.

This approach can be used in situations where information needs can be previously defined on a semantically high level by domain experts. Due to the proactive creation of consistency constraints erroneous data can be avoided or repaired directly in the data sources. When this approach is being applied in the described BQS-scenario, inconsistencies can directly be avoided in hospitals. This results in a much faster publication of more and consistent data. The structured dialogue can be avoided for a couple of cases where bad data quality would initiate the structured dialogue.

3.3 Duplicate Detection

[17] presents an algorithm and its evaluation for several configurations based on [3] for detecting duplicates in databases which are multiple representations of one real world entity and therefore a major issue relevant e.g. in the scenario of integrating several databases. Figure 6 shows a graphical representation of the algorithm which uses a classification algorithm from the data mining domain and will be explained in the following.

The algorithm consists of two consecutive phases, the learning phase and the application phase. In the learning phase a classifier learns the characteristics of duplicates from labeled data, i.e. pairs of instances that are marked as duplicates or non-duplicates. The algorithm's inputs are the distances between every two of the instances' attributes and the information whether or not the instances are duplicates. The algorithm's output is a classifier that is able to distinguish between duplicates and non-duplicates by having identified the combination and grade of those

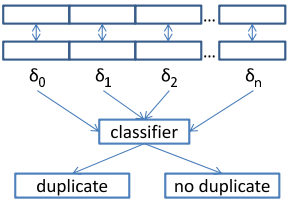

Fig. 6 A duplicate detection algorithm using a classification algorithm. In the learning phase, the algorithm's inputs are the distances between the corresponding attributes and the knowledge about whether or not the instances are duplicates, in the application phase that knowledge is deduced through the classifier that is the learning phase's output.

attributes' similarities, that are relevant for instances being duplicates. The application phase uses those classifiers for detecting duplicates in non-labeled data. The advantage over the statistical approach presented by [7] is the usage of similarity metrics, e.g. string distance metrics, to calculate the attributes' distances instead of using binary information whether two attributes have identical values or not as those metrics are more sensitive in the case of small differences.

Although the described algorithm can be used to find duplicates in any database using any data model the usage of an ontology provides a major advantage: As ontologies' concepts represent real world extracts without any normalization or considerations with respect to the performance of the database, e.g. by artificially inserting redundancy, those concepts' attributes completely describe a real world entity. Such an algorithm can therefore directly be applied to the concepts' instances as they semantically contain all information their real world counterparts are defined by. There is no "object identification problem" where real world entities are scattered around several data model elements, e.g. tables, or extended by artificial values like (primary) keys that are not relevant to the decision whether or not two instances represent the same real world entity. Therefore ontologies' conceptualizations provide an ideal basis for duplicate detection in databases. The ontology is furthermore used for the following applications: Labeling instances as correct or incorrect for using them as data that can be learned from, annotating the scales of measurement for proper preprocessing of the data, etc. Those annotations of user-defined metadata will be explained in the next section.

3.4 Metadata Annotation

Models for data quality are used to make statements about data regarding to their data quality. [2] point out that those models are a major issue for establishing a DQM approach. We show three DQM-specific metadata tasks where ontologies and especially their serializations in RDF (resource description framework) are an excellent choice for making those statements.

3.4.1 Data Provenance

Establishing a DQM approach often requires an integration of several data sources. Data provenance refers to the task of keeping track of the data's origins for correctly giving information about the data quality's state of those databases. XML Namespaces that are widely used to identify RDF's resources' origins can directly be used to point out the database the data is coming from.

To make this work data quality repositories act a little bit like data warehouses in the sense that they are used for integration of the data that's quality has to be managed. The integrated instances provide the information about their originating databases by their namespaces. These namespaces can than simply be used for compiling reports about the data quality of the respective databases by tracing back the evaluated instances to their origins.

3.4.2 Data Quality Annotations at Schema and Instance Level

Both at schema and instance level annotations are needed for DQM. At schema level DQM-algorithms might need to know the attributes' levels of measurements for proper preprocessing. At instance level several annotations like labeling for consistency checking (see section 3.1), duplicate detection (see section 3.3), or rule mining (refer to [4]) can be performed.

Again, RDF's resources provide an elegant way to make statements about data on both schema and instance level, as RDF-Schema and OWL-Ontologies are formulated as RDF-triples too. Those resources can be used as subjects in statements about data quality aspects, e.g. that a number is a nominal value (e.g. an identifier for a room) and therefore distances of two of those values cannot be calculated meaningfully. The duplicate detection algorithm must handle such information e.g. by applying "1" to the distance if those values are different and "0" otherwise.

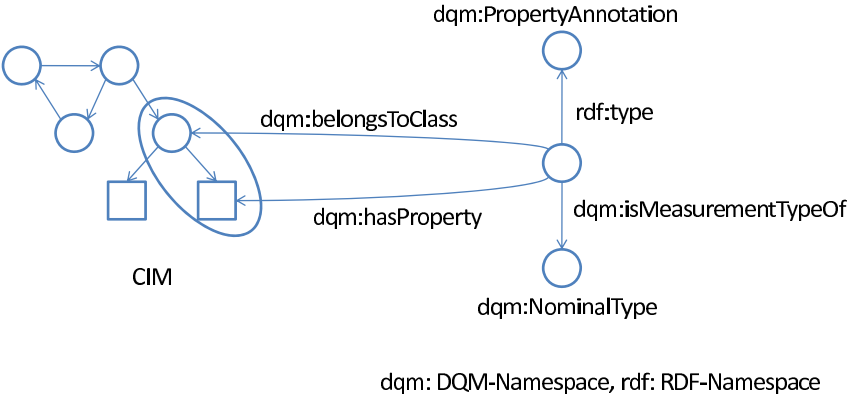

Fig. 7 Example of a metadata annotation at schema level

Such a metadata annotation is shown in figure 7. On the left hand side of the figure an ontology is shown which consists of concepts (circles) and datatype properties (squares). The statement we want to make about this ontology is that a certain concept's datatype property represents a nominal value (denoted by the ellipse). The right hand side of the figure shows a solution for making such a statement: An instance of a concept "PropertyAnnotation" is created that points out the concept ("belongsToClass") and the concept's datatype property ("hasProperty") in question. Finally, the statement about the level of measurements is made by the object "NominalValue" of the PropertyAnnotation's statement about its "isMeasurementTypeOf"-property.

Another example is shown in figure 8. In contradiction to the previous example, an annotation is made on instance level. The use case shown here is the marking of two instances that have been found suspicious to be duplicates by an algorithm like the one presented in section 3.1. Therefore an instance of the concept "DuplicateSuspicion" is created and the two instances of any (domain) ontology concept under suspicion are linked to that instance by the object properties "hasSuspicion".

3.4.3 An Ontology for the DQM-Domain

The annotations introduced in the preceding section need a vocabulary. Such a vocabulary can be provided by creating a DQM-ontology. The ontology in question has to cover concepts for the following annotations: At schema level the level of measurements have to be annotated for proper preprocessing. At instance level the preprocessed values as well as time stamps for measuring the value currencies have to be annotated. Furthermore, the already mentioned labeling of consistent tuples and labels for training data mining algorithms have to be annotated. Erroneous data has to be pointed out, specifying the reason for the suspected errors like outliers and inconsistencies (also see [19]). Marking duplicates needs a special concept as several

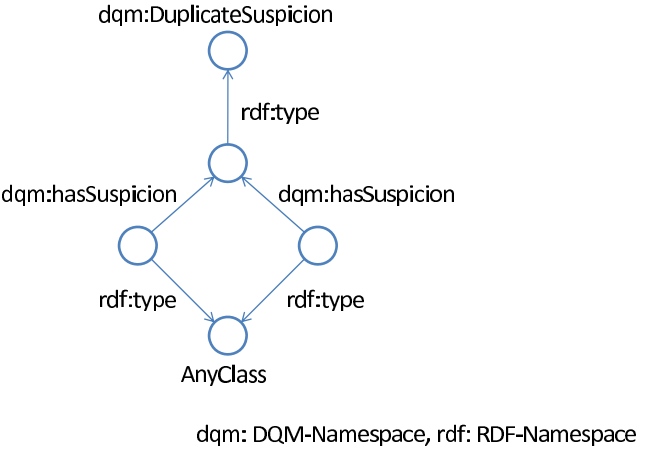

Fig. 8 Annotation of a duplicate suspicion

instances are involved like presented in the previous section. In this case further processing might e.g. involve the removal of one of those instances after confirming the multiple representation of a real world entity by those instances by a domain expert.

The concepts of figures 7 and 8 denoted by the namespace "dqm" belong to our proposed DQM-ontology. But not only is a vocabulary for making such statements defined by the DQM-ontology. It is further used for modeling DQM specific knowledge like presented in figure 9. There the relationships between three different DQM-aspects are defined: Data quality methods ("DataQualityMethod") describe algorithms for detecting possibly wrong data by e.g. applying statistical methods or data mining algorithms like the one presented in section 3.3 for identifying multiple representations of real world entities. Their results are interconnected with the data quality dimensions ("DQ-Dimension") which are used for measuring the quality of a database's data. Those dimensions can be defined and interconnected with the data quality algorithms outcomes by the user, so that the reports generated by the DQM fit the user's need for information regarding the data quality of his databases.

Figure 9 shows a predefined configuration where the information about the outcome of the algorithm for detecting duplicates is among others an input for the calculation of the dimension "Accuracy". In certain contexts it could be reasonable to define a dimension for duplicates itself, linking the new dimension with the outcome of the duplicate detection algorithm alone. Finally, the ontology defines the vocabulary that the algorithms use for marking suspicious data ("Suspicion"). Again, the usage of the concept used by the duplicate detection algorithm "DuplicateSuspicion" has already been introduced in the previous section and its application is shown in figure 8. With this information a report about the state of the

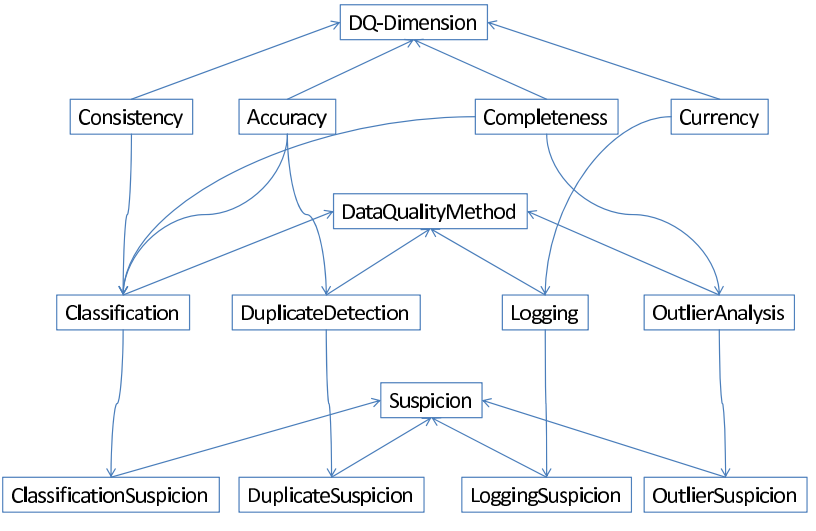

Fig. 9 Excerpt of the DQM-ontology showing interconnections between DQM-dimension, algorithms and suspicion annotations (straight lines being subclass definitions)

database's data quality can be generated as the marked suspicions can be taken into consideration for evaluating the different data quality dimensions.

3.5 Domain Ontologies as a Foundation for Correct Data

Beyond the use cases presented so far domain ontologies themselves provide semantic specifications that are useful for detecting erroneous data. We will show an example by remodeling the "Common Information Model CIM" (refer to [10]) which is an IEC standard for the utility domain by using OWL. The results presented in this section are part of a master thesis by Thomas Gebben [8] who kindly allowed us to present them here.

3.5.1 Introducing the CIM

As a domain ontology for the utility domain the CIM covers all informational aspects for that domain, like generation, protection, topology, measurements, assets, consumers, etc. The CIM itself is modeled in the Unified Modeling Language UML and is provided in several other formats like RDF by automated convert processes. As the outcome of those processes does not provide information that goes beyond that available in the source format, our approach is to build the CIM from scratch with the semantics provided by OWL.

3.5.2 Modeling the CIM as an Ontology

For the remodeling of the CIM we chose a so-called profile of the CIM which denotes a true subset of the CIM. The "Common Power System Model" CPSM [11] contains objects that are necessary to describe the topology and all assets of an electrical grid such as transformers, lines, substations, etc. A snipped of such a definition is shown in Listing 1.

```
<owl:FunctionalProperty rdf:ID="unitName">
  <rdfs:domain rdf:resource="#IdentifiedObject"/>
  <rdfs:range>
    <owl:DataRange>
      <owl:oneOf rdf:parseType="Resource">
        <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"> MW
          </rdf:first>
        <rdf:rest rdf:parseType="Resource">
          <rdf:rest rdf:parseType="Resource">
            <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
              > MVA </rdf:first>
            <rdf:rest rdf:parseType="Resource">
              <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#
                string"> Count </rdf:first>
              <rdf:rest rdf:parseType="Resource">
                <rdf:rest rdf:parseType="Resource">
                  <rdf:rest rdf:parseType="Resource">
                    <rdf:rest rdf:parseType="Resource">
                      <rdf:first rdf:datatype="http://www.w3.org/2001/
                        XMLSchema#string"> MVAr </rdf:first>
                      <rdf:rest rdf:parseType="Resource">
                        <rdf:rest rdf:parseType="Resource">
                          <rdf:rest rdf:parseType="Resource">
```

```
                        <rdf:first rdf:datatype="http://www.w3.org/2001/
                            XMLSchema#string"> SwitchPosition </rdf:first
                            >
                        <rdf:rest rdf:resource="http://www.w3.org
                            /1999/02/22-rdf-syntax-ns#nil"/>
                      </rdf:rest>
                      <rdf:first rdf:datatype="http://www.w3.org/2001/
                          XMLSchema#string"> TapPosition </rdf:first>
                    </rdf:rest>
                    <rdf:first rdf:datatype="http://www.w3.org/2001/
                        XMLSchema#string"> kV </rdf:first>
                  </rdf:rest>
                </rdf:rest>
                <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema
                    #string"> Ratio </rdf:first>
              </rdf:rest>
              <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#
                  string"> PerCent </rdf:first>
            </rdf:rest>
            <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#
                string"> Amperes </rdf:first>
          </rdf:rest>
        </rdf:rest>
      </rdf:rest>
      <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
          Degrees </rdf:first>
    </rdf:rest>
  </owl:oneOf>
 </owl:DataRange>
</rdfs:range>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
```

Listing 1. Definition of the property "unitName" of the concept "IdentifiedObject".

We define a property "unitName" which makes a statement about ("rdfs:domain") instances of the concept "IdentifiedObject" which are e.g. measurements. The range of the property is limited to one of the denoted values ("MW", "MVA", "Count", etc.). Furthermore, the property is defined as a functional property, which means that no or exactly one value per instance is allowed. By providing this information the ontology itself will point out instances that do not satisfy those conditions and therefore prevents wrong data from the beginning. Such an ontology can be used for generating messages to exchange data between different enterprises of the utility domain. In the following examples we will show the automated detection of erroneous instances that violate the stated rules. We will use Pellet [18] as a reasoner for this task which also generates the messages that will be shown.

3.5.3 Examples for Detecting Errors with a Domain Ontology

We now give two examples of wrong instances regarding the definition stated above. Figure 10 shows an instance of "MeasurementValue" that defines two different values for the property "unitName" for the same resource.

The reasoner correctly detects a violation of the functional aspect of the property. It therefore rejects the stated definition of the shown instance. Figure 11 shows an example of the violation of the "oneOf" data range restriction of the values of the property.

```
<MeasurementValue rdf:ID="MeasurementValueTESTDATAPROPERTY">
  <unitName rdf:datatype="http://www.w3.org/2001/XMLSchema#string">MW</unitName>
</MeasurementValue>

 <MeasurementValue rdf:ID="MeasurementValueTESTDATAPROPERTY">
  <unitName rdf:datatype="http://www.w3.org/2001/XMLSchema#string">MVA</unitName>
</MeasurementValue>


 WARN [main] (KnowledgeBase.java:1573) - Inconsistent ontology. Reason: Individu
al http://www.owl-ontologies.com/Ontology1209114222.owl#MeasurementValueTESTDATA
PROPERTY has more than one value for the functional property http://www.owl-onto
logies.com/Ontology1209114222.owl#unitName
```

Fig. 10 Instance of "MeasurementValue" with violation of the functional property "unitName"

```
<MeasurementValue rdf:ID="MeasutValueTESTDATAPROPERTY">
    <unitName rdf:datatype="http://www.w3.org/2001/XMLSchema#string">MVdddA</unitName>
</MeasurementValue>

WARN [main] (KnowledgeBase.java:1573) - Inconsistent ontology. Reason: Plain literal "MVdddA"^^http://www.w
3.org/2001/XMLSchema#string does not belong to datatype not(and([not(value(literal(SwitchPosition,(),http:/
/www.w3.org/2001/XMLSchema#string))),not(value(literal(TapPosition,(),http://www.w3.org/2001/XMLSchema#stri
ng))),not(value(literal(kV,(),http://www.w3.org/2001/XMLSchema#string))),not(value(literal(MVAr,(),http://w
ww.w3.org/2001/XMLSchema#string))),not(value(literal(Ratio,(),http://www.w3.org/2001/XMLSchema#string))),no
t(value(literal(MVA,(),http://www.w3.org/2001/XMLSchema#string))),not(value(literal(Count,(),http://www.w3.
org/2001/XMLSchema#string))),not(value(literal(PerCent,(),http://www.w3.org/2001/XMLSchema#string))),not(va
lue(literal(Degrees,(),http://www.w3.org/2001/XMLSchema#string))),not(value(literal(Amperes,(),http://www.w
3.org/2001/XMLSchema#string))),not(value(literal(MW,(),http://www.w3.org/2001/XMLSchema#string)))])). Liter
al value may be missing the rdf:datatype attribute.
```

Fig. 11 Instance of "MeasurementValue" with violation of the "oneOf" data range restriction of the property "unitName"

The reasoner again detects an inconsistent ontology stating that the literal "MVddA" is not defined as one of the allowed values for the property in question.

Several classes of consistency checks are performed in a similar manner like checking cardinality constraints, assuring that only concepts defined by the profile are used, etc.; we use the complete expressivity of the OWL-DL dialect to create a powerful ontology that conforms to the CPSM standard and rejects as many malformed instantiations of its concepts as possible.

4 Conclusions and Future Work

The usage of the knowledge provided by domain ontologies can be used to improve DQM's outcomes in several ways as shown by five given examples, namely consistency checking, proactive management of consistency constraints, duplicate detection, seamless possibility of metadata annotation, and semantic domain modeling. Therefore, a synergy effect from modeling a domain ontology, e.g. for defining a shared vocabulary for improved interoperability, and DQM can be achieved. The further work will include the appliance of our described approaches on enterprise scaled databases to verify their applicability. The EWE AG (please visit `www.ewe.de`) partly funds the projects the presented results originate from and also provide such data for large scale tests. As described, other test scenarios are tumour classification in cancer registries and the reports for the BQS.

References

[1] Amicis, F.D., Batini, C.: A methodology for data quality assessment on financial data. Studies in Communication Sciences 4, 115–136 (2004)
[2] Batini, C., Scannapieco, M.: Data Quality. Springer, Heidelberg (2006)
[3] Bilenko, M., Mooney, J.R.: Employing trainable string metrics for information integration. In: Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web, Acapulco, Mexico, pp. 67–72 (August 2003)
[4] Brüggemann, S.: Rule mining for automatic ontology based data cleaning. In: Zhang, Y., Yu, G., Bertino, E., Xu, G. (eds.) APWeb 2008. LNCS, vol. 4976, pp. 522–527. Springer, Heidelberg (2008)
[5] Brüggemann, S.: Proaktives Management von Konsistenzbedingungen im Analytischen Performance Management. In: Proceedings of DW 2008, Synergien durch Integration and Informationslogistik (2008)
[6] Fellegi, I.P., Holt, D.: A systematic approach to automatic edit and imputation. Journal of the American Statistcal Association 71, 17–35 (1976)
[7] Fellegi, I.P., Sunter, A.B.: A theory for record linkage. Journal of the American Statistical Association 64(328), 1183–1210 (1969)
[8] Gebben, T.: OWL-Reasoner basierte Gültigkeitsprüfung von CIM-Topologien gemäßCommon Power System Model (CPSM). Master thesis, Universität Oldenburg (to be published) (2009)
[9] Hinrichs, H.: Datenqualitätsmanagement in Data Warehouse-Systemen. PhD thesis, Universität Oldenburg (2002)
[10] IEC - International Electrotechnical Commission: IEC 61970:301: Energy management system application program interface (EMS-API) - Part 301: Common Information Model (CIM) Base. International Electrotechnical Commission (2003)
[11] IEC - International Electrotechnical Commission: IEC 61970: Energy Management System Application Program Interface (EMS-API) - Part 452: CIM Network Applications Model Exchange Specification. International Electrotechnical Commission (2006)
[12] International Union Against Cancer (UICC). TNM Classification of Malignant Tumours, 6th edn. John Wiley & Sons, New Jersey (2001)
[13] Kedad, Z., Métais, E.: Ontology-based data cleaning. In: Andersson, B., Bergholtz, M., Johannesson, P. (eds.) NLDB 2002. LNCS, vol. 2553, pp. 137–149. Springer, Heidelberg (2002)
[14] Microsoft Corporation: Domain Specific Language Tools, `http://msdn2.microsoft.com/en-us/vstudio/aa718368.aspx/` (Feburary 12, 2009)
[15] Milano, D., Scannapieco, M., Catarci, T.: Using ontologies for xml data cleaning. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM-WS 2005. LNCS, vol. 3762, pp. 562–571. Springer, Heidelberg (2005)
[16] Rahm, E., Do, H.H.: Data cleaning: Problems and current approaches. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering 23(4), 3–13 (2000)
[17] Schünemann, M.: Duplikatenerkennung in Datensätzen mithilfe selbstlernender Algorithmen. Master thesis, Universität Oldenburg (2007)
[18] Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Web Semantics: Science, Services and Agents on the World Wide Web 5(2), 51–53 (2007)

[19] Uslar, M., Grüning, F.: Zur semantischen Interoperabilität in der Energiebranche: CIM IEC 61970. Wirtschaftsinformatik 49(4), 295–303 (2007)
[20] Wang, X., Hamilton, H.J., Bither, Y.: An ontology-based approach to data cleaning. Technical report, Department of Computer Science, University of Regina (June 2005)
[21] Wietek, F.: Intelligente Analyse multidimensionaler Daten in einer visuellen Programmierumgebung und deren Anwendung in der Krebsepidemiologie. PhD thesis, Universität Oldenburg (2000)