An Ontology-based Methodology for Reusing Data Cleaning Knowledge

Ricardo Almeida[1], Paulo Maio[1,2], Paulo Oliveira[1,2] and João Barroso[3]
[1]*ISEP-IPP – School of Engineering of Polytechnic of Porto, 431 R. Dr. Bernardino de Almeida, Porto, Portugal*
[2]*GECAD – Research Group on Intelligent Engineering and Computing for Advanced Innovation and Development, Porto, Portugal*
[3]*UTAD – University of Trás-os-Montes and Alto Douro, Vila Real, Portugal*

Keywords: Data Quality, Data Cleaning, Knowledge Reuse, Vocabulary, Ontologies.

Abstract: The organizations' demand to integrate several heterogeneous data sources and an ever-increasing volume of data is revealing the presence of quality problems in data. Currently, most of the data cleaning approaches (for detection and correction of data quality problems) are tailored for data sources with the same schema and sharing the same data model (e.g., relational model). On the other hand, these approaches are highly dependent on a domain expert to specify the data cleaning operations. This paper extends a previously proposed data cleaning methodology that reuses cleaning knowledge specified for other data sources. The methodology is further detailed/refined by specifying the requirements that a data cleaning operations vocabulary must satisfy. Ontologies in RDF/OWL are proposed as the data model for an abstract representation of the data schemas, no matter which data model is used (e.g., relational; graph). Existing approaches, methods and techniques that support the implementation of the proposed methodology, in general, and specifically of the data cleaning operations vocabulary are also presented and discussed in this paper.

1 INTRODUCTION

Globalization has resulted in a high number of acquisitions, partnerships and/or fusions among organizations worldwide. This has forced some organizations to change their working methods and business models. On the other hand, globalization has also gave rise to technological advances, namely in the way organizations and people communicate with each other.

Information systems of the organizations are being constantly challenged to adapt and to integrate several and heterogeneous data sources. The heterogeneity of the data sources mainly relies on two issues: (i) the underlying data model (e.g., relational data model (Codd, 1970); object-oriented (Booch, 1993); document-oriented (Han et al., 2011); triples/graphs (Brickley and Guha, 2014)); and, (ii) the adopted schema which corresponds to a conceptualization of a given domain/application and, therefore, may differ on terminology, semantics and granularity.

In this context of change, data of a single data source is often: (i) collected under different circumstances (e.g., application/device used to collect the data; the applied business rules); and/or, (ii) the result of (one or more) data integration processes. Both situations can be the cause of quality problems in the data. In the scope of this paper, Data Quality Problems (DQPs) are problems that exist at data/instance level, such as: missing values in mandatory attributes; domain violations; uniqueness violations; business rules violations; existence of duplicates (equal; approximate; or inconsistent (Oliveira et al., 2005a; 2005b)). Data cleaning is a technique (or a process) for detection and correction of DQPs (Milano et al., 2005), which cannot be fully automatic, since it requires the intervention of an expert (Dasu et al., 2003) for specifying the detection and correction operations to be executed. After analysing the extensive literature on this subject (Fürber and Hepp, 2011; Knuth and Sack, 2014; Oliveira et al., 2009; Weis and Manolescu, 2007), three main conclusions were achieved.

First, all data cleaning methodologies/tools follow a similar process that is highly dependent on the domain expert and rely on a large number of detection and correction operations manually specified. Usually, the process is made up of the following steps: (i) ask the expert to specify the detection operations; (ii) run the specified detection operations; (iii) ask the expert to specify the correction operations to be carried on the data to solve the identified problems; and, (iv) run the specified correction operations to clean the data.

Second, most of the existing methodologies/tools are specific for a given data model (e.g., relational databases) and, therefore, exploit the particularities of that data model, preventing their applicability on data sources with another data model (Almeida et al., 2012). Even when the data model is the same, usually there are differences between the schemas of two data sources representing the same application domain. It is not possible to apply the same set of cleaning operations in two data sources if their schemas are different. Thus, these approaches are only suitable for scenarios were all data sources share, the same data model and data schema.

Third, despite some of the analysed methodologies/tools are targeted to a given application domain (e.g., health), it has been recognized that there are a significant amount of DQPs and data cleaning operations that are generic enough to be applied on different data sources (most certainly with different schemas) from distinct application domains (e.g., detection/correction of DQPs in e-mail attributes; detection/correction of DQPs in zip codes).

Nowadays, the concerns about the quality of data and DQPs are especially important, because the amount of generated and collected data has exponentially increased, as exposed by advents such as the Internet of Things (IoT) (Atzori et al., 2010), and Big Data (Hashem et al., 2015; Snijders et al., 2012). It is important to keep in mind that data by themselves are not valuable. The value is in the analysis done on the data and how the data are turned into information and, eventually, into knowledge. However, if the data is affected by quality problems, the quality of the analysis will also reflect that. This is known as the GIGO (garbage in, garbage out) principle.

Previously, we have proposed a novel and generic data cleaning methodology that intends to assist the domain expert in the specification of the cleaning operations (Almeida et al., 2015). The methodology reuses cleaning knowledge previously specified for other data sources, even if those sources have different data models and/or schemas. This paper details the methodology further by: (i) specifying the requirements that a Data Cleaning Operations Vocabulary (DCOV) must satisfy; (ii) proposing the use of ontologies in RDF/OWL as a data model for an abstract representation of any data schema, no matter the data model in which it is based (e.g., relational; graph); and, (iii) analysing the literature in order to identify approaches, methods, and techniques that support/facilitate the implementation of the proposed methodology as a whole and, specifically, of the DCOV.

The paper is organized as follows. Section 2 briefly describes and exemplifies the data cleaning methodology (Almeida et al., 2015). In Section 3, the requirements that a Data Cleaning Operations Vocabulary (DCOV) must satisfy are described in detail. The DQM ontology (Fürber and Hepp, 2011), seen as the available vocabulary closest to cover most of the DCOV requirements, is presented in Section 4. Section 5 provides a brief description about issues regarding the implementation of the mapping, transformation and data cleaning operations rewriting processes of the Bridge Layer considered in the methodology. At last, in Section 6, conclusions are presented as well as future work directions.

2 DATA CLEANING METHODOLOGY

The Data Cleaning Methodology proposed in (Almeida et al., 2015) relies and promotes the following principles: (i) the specification of Data Cleaning Operations (DCOs) for a given data source (say *DS*$_1$) should be carried on by a domain expert through an application operating as much as possible closer to the human conceptual level and (ii) to better assist the domain expert it is able to take advantage of data cleaning knowledge (including DCOs) specified previously on the context of another data source (say *DS*$_2$) whose domain partially overlaps with *DS*$_1$ domain, even when *DS*$_1$ and *DS*$_2$ have a different data schema and/or model. In the following, this methodology (depicted in Figure 1) is briefly described and complemented with a running example based on the scenario introduced in **Example 1**.

Example 1 (Scenario). Consider the scenario where an organization (say *Org1*) in result of an on-going business integration process ends up with three distinct databases (say *DB*$_1$, *DB*$_2$ and *DB*$_3$) about

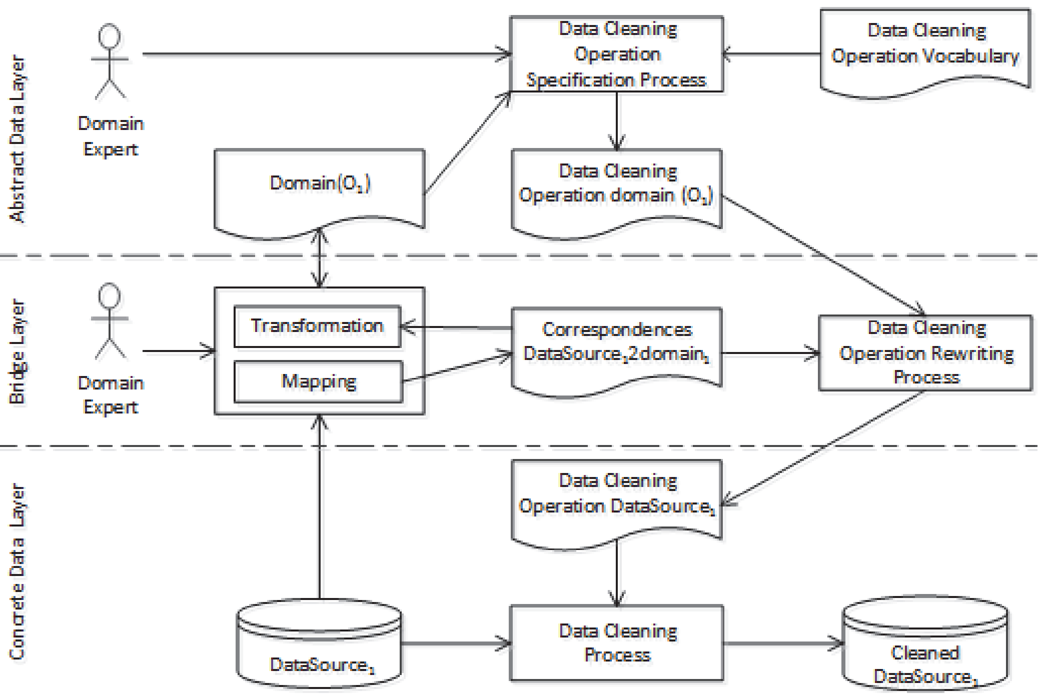
Abstract Data Layer
Domain Expert
Data Cleaning Operation Specification Process
Data Cleaning Operation Vocabulary
Domain(O1)
Data Cleaning Operation domain (O1)
Bridge Layer
Domain Expert
Transformation
Mapping
Correspondences DataSource1 2domain1
Data Cleaning Operation Rewriting Process
Concrete Data Layer
Data Cleaning Operation DataSource1
DataSource1
Data Cleaning Process
Cleaned DataSource1

Figure 1: Graphical representation of the adopted Data Cleaning Methodology.

respectively a graph-oriented and document-oriented NoSQL databases having different schemas. Yet, it was perceived that all databases have several DQPs that need to be fixed.

The Concrete Data Layer (CDL) comprehends a specific automatic Data Cleaning Process (DCP) that is able to interpret and execute a set of DCOs specified according to (i) the DCP' requirements (e.g. required DCOs' specification language) and to (ii) the particularities (e.g. schema and model) of the data source to be clean. The result is a cleaned data source that is a changed copy of the original data source, so that the detected data quality problems are solved (cf. **Example 2**). However, the domain expert only specifies DCOs at the Abstract Data Layer (ADL) supported by a Data Cleaning Operations Specification Process (DCOSP).

Example 2 (Concrete Data Layer). From the provided scenario description, one can say that *Org1* has three concrete data sources to be clean: DB_1, DB_2 and DB_3. Yet, consider that *Org1* has three distinct DCPs (say DCP_1, DCP_2 and DCP_3). DCP_1 requires DCOs to be specified as SQL queries. Conversely, DCP_2 requires DCOs to be specified as SPARQL queries while DCP_3 requires queries following a MongoDB[1] syntax.

[1] https://www.mongodb.com

Besides the domain expert, the DCOSP requires (i) a conceptualization of the domain of interest and (ii) a Data Cleaning Operation Vocabulary (DCOV). The vocabulary aims to univocally describe the structure and semantics of (outputted) DCOs for any domain of interest. Instead, the conceptualization aims to capture and describe a domain of interest in a more abstract and intuitive manner than concrete data sources. Thus, the result of the DCOSP is a set of DCOs specified at a conceptual level and, therefore, independent of any concrete data source.

In this work, the team adopts OWL-DL ontologies (McGuinness and Harmelen, 2004) to capture either the vocabulary and the domain conceptualization as ontologies are seen as the best answer for the demand of having intelligent systems operating closer to the human conceptual level (Obrst et al., 2003) and they may vary in different levels of expressiveness (e.g. *ALC* vs. *SHOIN*). Moreover, on section 4 the team proposes and describes a vocabulary fulfilling all the methodology requirements (cf. section 3).

The Bridge Layer (BL) enables the interoperability between the concrete and the abstract data layers through a set of semi-automatic processes that are able:

- To Generate a domain conceptualization from a given concrete data source (cf. **Example 3**);

Example 3 (Conceptualization). Consider that instead of adopting an existing domain conceptualization of customers and purchased products, *Org1* decides to adopt its own conceptualization grounded on the schema of DB_1 since it is seen as the most appropriate (or complete) comparatively to DB_2 and DB_3. Therefore, no difference regarding terminology, semantics and granularity would exist between DB_1 and O_D. Details on how this can be done are provided in section 5.

- To generate a set of correspondences between entities of a concrete data source and the entities of domain conceptualization (cf. **Example 4**);

Example 4 (Generating Correspondences). In our scenario, the Mapping process needs to generate three sets of correspondences: one between DB_1 and O_D (say A_{1D}); other between DB_2 and O_D (say A_{2D}); and another between DB_3 and O_D (say A_{3D}). Details on how this can be done are provided in section 5.

- To exploit previously established correspondences to rewrite the DCOs specified by the domain expert at the ADL according to the target concrete data source (cf. **Example 5** and **Example 6**).

Example 5 (Defining DCOs). Consider that *Org1* adopts the DCOV described in this work (say $DCOV_{Org1}$) such that it describes (structurally and semantically) all DCOs (say DCO_{OD}) defined at the ADL by the domain expert with the support of a given DCOSP over O_D.

Example 6 (Rewriting DCOs). At the BL, is responsibility of the rewriting process (DCORP) to translate DCO_{OD} previously specified in order to be correctly interpreted and executed by DCP_1, DCP_2 and DCP_3. For that, it takes as input the pair $<DCO_{OD}, A_{1D}>$ (or $<DCO_{OD}, A_{2D}>$ or $<DCO_{OD}, A_{3D}>$) and outputs DCO_1 (or DCO_2 or DCO_3 respectively) as being the DCOs to be executed on DB_1 (or DB_2 or DB_3 respectively) by DCP_1 (or DCP_2 or DCP_3).

At last, it is worth noticing that (i) establishing correspondences between a concrete data source and a domain conceptualization and (ii) rewriting conceptual DCOs to concrete DCOs are two mandatory tasks. On the other hand, the task of generating a domain conceptualization it is only necessary in scenarios where there is not an (available) domain conceptualization.

3 REQUIREMENTS FOR THE DATA CLEANING OPERATION VOCABULARY

This section describes the envisioned requirements that a given Data Cleaning Operation Vocabulary (DCOV) must satisfy. Such requirements are specified via a use case model and a set of competency questions. Further, in section 4, an existing ontology is described and analysed regarding the specified requirements.

3.1 Use Case Model

The DCOV is an artefact used by a DCOSP to support the specification of DCOs, namely its structure and semantics, which in turn are further rewritten by a DCORP in order to be executed. Considering this, both the DCOSP and the DCORP have influenced the derived use case model depicted in Figure 2, which is exclusively focused on Detection DCOs (i.e. Correction DCOs were not considered so far).

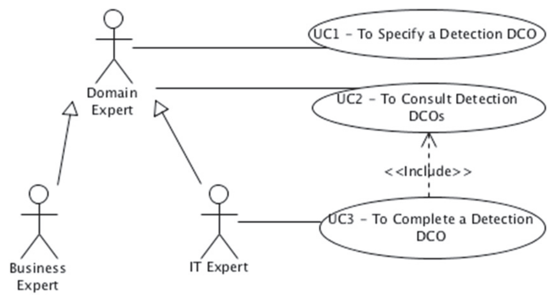
UC1 - To Specify a Detection DCO
Domain Expert
UC2 - To Consult Detection DCOs
<<Include>>
UC3 - To Complete a Detection DCO
Business Expert
IT Expert

Figure 2: Partial Use Case Diagram.

From the depicted use case model, it is perceived that the actor "domain expert" suggested in the methodology is specialised in two distinct actors:

- Business Expert: corresponds to a person that has a widely and deeply knowledge about the domain of interest in hands (e.g. customers and purchased products) and, therefore, (s)he is aware of which detection DCOs must hold. However, (s)he lacks some specific Information Technology (IT) knowledge (e.g. the ability to specify a regular expression or a functional dependency between attributes) that might be need for the specification of some kind of DCOs;
- IT Expert: corresponds to a person that has some knowledge regarding the domain of interest in hands, typically less than the one that the Business Expert has, but having all the necessary

IT knowledge for a complete DCOs specification.

In the following, each identified use case is briefly described.

UC1 - To Specify a Detection DCO: the domain expert is somehow browsing the data elements (e.g. concepts and properties) of the domain of interest and aims to specify a detection DCO for a given (set of) data element(s). The system supports the user by presenting the supported kinds of DCOs and asks for selecting one. Further, the user is guided (e.g. through a wizard) to enter/identify the information required to properly instantiate the selected kind of DCO (distinct kinds of DCO have different information demands). Additionally, information regarding the provenance or the context on which such DCO is applicable might be collected. Besides, the system must handle underspecified DCOs (i.e. DCOs having missing information).

UC2 - To Consult Detection DCOs: the domain expert aims to know and analyse the DCOs that were previously specified. The system must provide the ability to list all the existing DCOs and support filtering that list by multiple criteria such as (i) the ones that apply on a given data element; (ii) the ones that are of a given kind; (iii) the ones that are underspecified.

UC3 - To Complete a Detection DCO: the IT expert aims to complete the specification of a previously selected DCO that is underspecified. For that, the system might provide the already specified information and ask for the missing one (e.g. to detect misspelling errors the user must select the dictionary that should be used).

3.2 Competency Questions

Based on the use case model and on a literature analysis of typical data quality problems, the DCOV must also satisfy the requirements arising from the following competency questions:

CQ 1: Which kind of data cleaning operations can be specified?

CQ 2: Which information needs to be collected in order to instantiate properly a given data cleaning operation, namely a detection DCO?

CQ 3: Which kind of information is used to instantiate a detection DCO?

CQ 4: Which DCOs are underspecified, i.e. DCOs having missing information?

CQ 5: Which detection DCOs apply on instances of class C or values of property P?

CQ 6: Which DCOs were created by Person X?

3.3 Summary

In summary, the DCOV should provide a mean such that a given DCOSP is able to know (i) the different kinds of Detection DCOs that are possible to instantiate; (ii) the variety of information that is required to instantiate a given kind of Detection DCO; (iii) the specified Detection DCOs and its status (e.g. underspecified); (iv) the provenance or context on which a DCO is applicable. On the other hand, the DCOV should provide a mean such that a

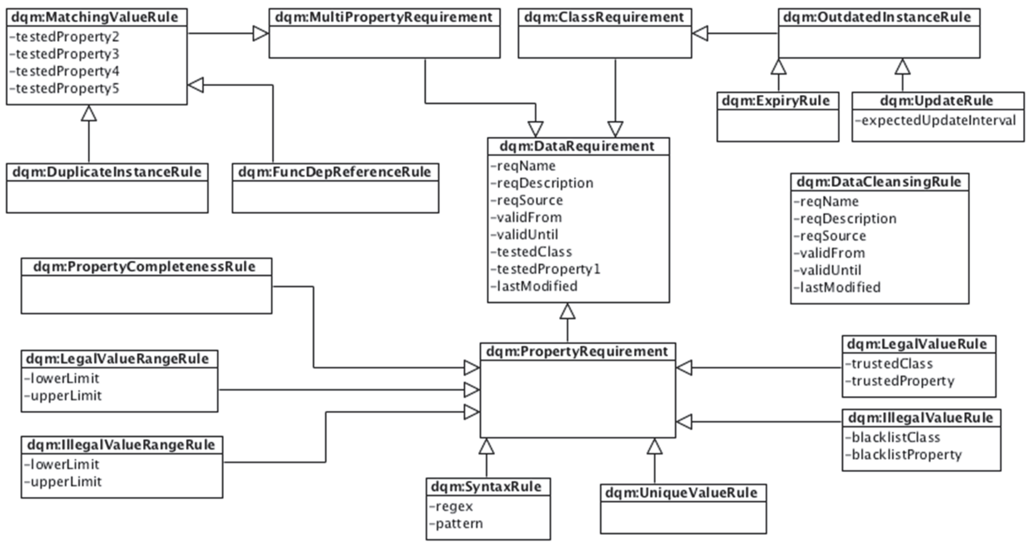

Figure 3: Partial UML representation of the DQM Ontology.

given DCORP has proper meta-data regarding the information collected during the instantiation of a DCO in order to determine the conditions under which a DCO is rewritable according to the terms of the target data cleaning process.

4 ADOPTING A DATA CLEANING OPERATION VOCABULARY

After an extensive literature review, to the best of our knowledge, the DQM ontology (Fürber and Hepp, 2011) is seen as the available vocabulary that is closest to cover most of the DCOV' requirements identified previously (cf. section 3). Thus, in the following, the main elements (classes and properties) of the DQM ontology are introduced (cf. section 4.1) and discussed in light of such requirements (cf. section 4.2). Some complementary examples are provided considering the domain conceptualization introduced in **Example 7**.

Example 7 (Customer Conceptualization). Consider that O_D of *Org1* comprehends a class customer (`ex:Customer`) and properties for capturing customers' name, address, tax number, the maximum discount allowed (in percentage) and the last time that customer information was updated (`ex:name`, `ex:address`, `ex:taxNumber`, `ex:maxDiscount` and `ex:lastUpdate` respectively).

4.1 DQM Ontology

The DQM ontology was designed to represent knowledge regarding data quality monitoring, data quality assessment and data cleaning operations in RDF/OWL. However, in the context of this work, the DQM ontology (partially depicted in Figure 3) is described and analyzed regarding the representation of DCOs only.

As previously introduced, data cleaning operations are commonly distinguished between the ones focused on detecting that some kind of quality problem holds (called detection DCOs) and the ones focused on resolving the detected quality problems (called correction DCOs). The DQM ontology makes this distinction through two main classes: `dqm:DataRequirement` (as it is online available[2] or as originally published `dqm:DataQualityRule`) and `dqm:DataCleansingRule` respectively. Each kind of DCO is captured as being a sub-class of one of these two classes, depending on its nature (i.e. detection or correction). Although, a given sub-class (e.g. `dqm:PropertyRequirement`) may represent a kind of DCO that is not worth of being instantiated since its purpose is serving as a logical arrangement which may capture a set of characteristics shared by its direct sub-classes. Given this, and seeing the hierarchy of classes as a tree, just the leaves classes (e.g. `dqm:UniqueValueRule`) capture a kind of DCO that worth of being instantiated. Currently, detection DCOs are organized in three distinct logical categories: `dqm:PropertyRequirement`, `dqm:MultiPropertyRequirement` and `dqm:ClassRequirement`.

[2] http://semwebquality.org/dqm-vocabulary/v1/dqm

The `dqm:PropertyRequirement` represents DCOs acting on a single property (`dqm:testedProperty1`) of a given class (`dqm:testedClass`) as shown in **Example 8** and **Example 9**. Such DCOs intend, for example, to check which instances of the given class do not have specified a value for such property (`dqm:PropertyCompletenessRule`), or, instead, if the property value is valid. The validity of a value may be expressed using three alternative perspectives: (i) by specifying a legal range of values (`dqm:LegalValueRangeRule` for numeric values in a given interval or `dqm:LegalValueRule` for finite enumerated values); (ii) by specifying an illegal range of values (respectively `dqm:IllegalValueRangeRule` or `dqm:IllegalValueRule`); and (iii) by checking if the property value meets a given syntax (`dqm:SyntaxRule`) expressed, for instance, as a regular expression. At last, it can be verified if all values of the given property are unique (`dqm:UniqueValueRule`).

Example 8 (PropertyCompletenessRule). Taking into consideration the known O_D of *Org1* and using the DQM ontology as DCOV, one could define a DCO to check if there is any customer whose tax number is unknown. For this, the following detection DCO could be specified as follows (using Turtle syntax):

```
[] a dqm:PropertyCompletenessRule;
    dqm:testedClass ex:Customer;
    dqm:testedProperty1 ex:taxNumber.
```

Example 1 (LegalValueRangeRule). Consider that *Org1* establish that the maximum discount that any customer might have is 20%. To check if there is any customer currently having a discount higher than 20%, one could define the following DCO:

```
[] a dqm:LegalValueRangeRule;
   dqm:testedClass ex:Customer;
   dqm:testedProperty1 ex:maxDiscount;
   dqm:upperLimit "20%".
```

The `dqm:MultiPropertyRequirement` represents DCOs acting on at least two properties (`dqm:testedProperty1` to `dqm:tested PropertyN`) of a given class (`dqm:testedClass`) as shown in **Example 10.** Such DCOs intend, for instance, to check the existence of duplicated instances (`dqm:DuplicateInstanceRule`) or to validate functional dependencies between properties (`dqm:FuncDepReferenceRule`);

Example 10 (DuplicateInstanceRule). Consider that *Org1* aims to check the existence of duplicated customers based on the customers' name and address. Thus, one could define the following DCO:

```
[] a dqm:DuplicateInstanceRule;
    dqm:testedClass ex:Customer;
    dqm:testedProperty1 ex:name;
    dqm:testedProperty2 ex:address.
```

The `dqm:ClassRequirement` represents DCOs acting on the instances of a given class as a whole (cf. **Example 11**) in order to detect the existence of instances of such class whose expiration date is overcome (`dqm:ExpiryRule`) or whose last update is too old and, therefore, needs to be updated (`dqm:UpdateRule`).

Example 11 (UpdateRule). Consider that *Org1* defines that customers information should be updated in a max interval of four years (i.e. 1641 days). Thus, one could define the following DCO:

```
[] a dqm:UpdateRule;
    dqm:testedClass ex:Customer;
    dqm:testedProperty1 ex:lastUpdate;
    dqm:expectedUpdateInterval "1461".
```

In addition, every DCO may also have a set of complementary information such as: (i) a name and a description (respectively `dqm:reqName` and `dqm:reqDescription`); (ii) a temporal definition of the DCO validity through the `dqm:validUntil` and/or `dqm:validFrom` properties; (iii) last time the DCO was revised (`dqm:lastModified`); and (iv) to keep track of the information source affording such DCO (`dqm:reqSource`). Besides, it is suggested to combine the DQM ontology with well-known vocabularies (e.g. Dublin Core Metadata[3]) to identify who is the DCO creator (`dc:creator`).

4.2 Improving DQM Ontology

According to authors of the DQM ontology, due to its early stage of development, it may have some important elements missing. Besides, it is also necessary to realize (i) which of the identified requirements are already satisfied and how; and (ii) which requirements still need to be satisfied and, therefore, demanding that the DQM ontology is improved and/or extended.

The hierarchy of sub-classes of class `dqm:Data Requirement` and `dqm:DataCleansingRule` captures the kind of data cleaning operations that can be specified (CQ 1) and allows them to be somehow grouped according to a logical perspective. Yet, accommodation of other (and probably more complex) kind of DCOs is possible by adding new sub-classes and new properties (cf. **Example 12**).

Example 12 (Extending DQM Ontology). One may require that the values of properties capturing textual descriptions are written in accordance to a particular natural language. Such kind of detection DCO is not supported yet. To support that, a new class (e.g. `new:SpellerSyntaxRule`) would be added as depicted in figure 4.

In this respect, (the usage of) some properties may also need to be slightly changed in order to allow a better categorization and/or description of its values since such information may be important for the purpose of rewriting a DCO (cf. **Example 13**).

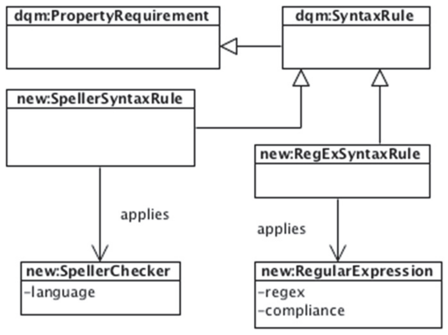

Figure 4: Extending partially the DQM Ontology.

Example 13 (Refining DQM Ontology). Currently, the range of acceptable values for the `dqm:regex` property is a `string` representing a regular expression. This does not allow capturing the compliance[4] in which regular expressions are specified. To support this, DQM Ontology could be refined as depicted in Figure 4.

By following and generalizing the approach taken in **Example 12** and **Example 13** one would also

[3] http://www.dublincore.org/2010/10/11/dcelements.rdf

[4] ISO/IEC/IEEE 9945:2009.

explicitly identify additional kind of information (CQ 3) that play a concrete role in the DCO specification (e.g. `new:SpellerChecker`).

Information regarding a given instance of a DCO is collected by means of a diverse set of available properties (e.g. `dqm:testedClass`, `dqm:testedProperty1`, `dqm:reqDescription`, `dqm:reqName`). Still, no means is provided to identify which information is mandatory and which is optional. However, a mechanism for doing that would foster the verification of if a given DCO instance is properly instantiated (CQ 2) or, instead, it is an underspecified instance (CQ 4). Regarding this issue, it is envisaged to combine the DQM ontology with an external vocabulary enabling such features.

Finally, it is worth noticing that questions similar to CQ 5 and CQ 6 are answered by means of simple SPARQL queries as the one shown in **Example 14**.

Example 14 (SPARQL Query). In order to know which detection DCOs created by "Jim Jam" apply on class `ex:Customer`, one could executed the following SPARQL query:

```
select ?dco where {
      ?dco a dqm:DataRequirement;
            dqm:testedClass ex:Customer;
            dc:creator "Jim Jam". }
```

5 BRIDGING CONCRETE AND ABSTRACT DATA LAYERS

This methodology also relies on the ability to successfully interoperate between the so-called Concrete Data Layer and the Abstract Data Layer. Thus, this section provides a brief description about issues regarding the implementation of the Mapping, Transformation and DCO rewriting processes of the Bridge Layer.

The Mapping process aims to establish, at the conceptual level, a set of correspondences between a concrete and an abstract data source. As proposed in this work, the abstract data source corresponds to a RDF/OWL ontology about the domain of interest in hands. However, the concrete data source may be any kind of repository (e.g. relational databases, NoSQL databases). Given this, in order to facilitate the process of establishing correspondences it is envisioned an *a priori* task responsible for translating in a direct way the schema of the concrete data source to a RDF/OWL ontology. Regarding this task, for relational databases it can be used the Direct Mapping approach (Arenas et al., 2012) or, alternatively, the R2RML (Das et al., 2012) approach. Both are a W3C recommendation for mapping relational databases to ontologies. For non-relational databases the adoption of existing similar approaches/tools is envisioned. Ultimately, if those approaches/tools do not exist, a two-steps approach might be adopted. First, from the non-relational model (e.g., object-oriented) to the relational model (e.g. through an ORM tool) and, further, from the relational model to the ontological model as previously described. After having the schema of the concrete data source represented in a RDF/OWL ontology several semi-automatic techniques from the schema matching (Bellahsene et al., 2011) and ontology-matching (Otero-Cerdeira et al., 2015) domains can be applied to discover and/or suggest the required correspondences.

Example 15 (Correspondences). As a result of the mapping process, one could establish the following equivalence correspondences between O_D and DB_1, O_D and DB_2, O_D and DB_3): `<ex: Customer, DB`$_1$`:Customer>`, `<ex:Customer, DB`$_2$`:Buyer>` and `<ex:Customer, DB`$_3$`: Account>`.

The Transformation process exploits the correspondences established by the Mapping process in order to transform (if necessary) the data (at the extensional level) of a concrete data source (e.g. DB_1, DB_2, DB_3) into data (i.e. instances) of the abstract data source (e.g. O_D). This process may rely, for instance, in the SPARQL-RW approach (Makris et al., 2012) or in the SBO approach (Maedche et al., 2002).

The DCO rewriting process also exploits the correspondences established by the Mapping process to rewrite the DCOs specified for a given domain/application (e.g. O_D) at the abstract data layer according to the target data source (e.g. DB_1, DB_2, DB_3). **Example 16**, **Example 17** and **Example 18** demonstrates the intended feature.

Example 16 (Rewriting a DCO to a SQL Query). Considering the equivalence correspondences `<ex:maxDiscount, DB`$_1$`:DiscMax>` and `<ex: Customer, DB`$_1$`:Customer>`, DCP_1 could rewrite the DCO specified in **Example 9** to a SQL query as follows:

```
Select * From DB1:Customer
      where DB1:DiscMax > 20%
```

Example 17 (Rewriting a DCO to a SPARQL Query). Considering the equivalence correspondences `<ex:maxDiscount, DB`$_2$`:MaxDiscPerc>` and `<ex:Customer, DB`$_2$`:Buyer>`, *DCP*$_2$ could rewrite the DCO specified in **Example 1** to a SPARQL query as follows:

```
Select ?c where {
      ?c a DB2:Buyer;
      ?c DB2:MaxDiscPerc ?desc;
      Filter (?desc > 20%)}
```

Example 18 (Rewriting a DCO to a MongoDB Query). Considering the equivalence correspondences `<ex:maxDiscount, DB3:MaxDiscount>` and `<ex:Customer, DB3:Account>` *DCP*$_3$ could rewrite the DCO specified in **Example 9** to a MongoDB query as follows:

```
Db.DB3:Account.find({
      DB3:MaxDiscount:{$gt: 20%} });
```

Yet, it is worth to underline that besides the target data source, the output of the rewriting process depends on the data cleaning process that will execute the specified DCO on the target data source. As such, DCOs may be rewritten to SQL or SPARQL or MongoDB queries (as exemplified) or to any other syntax that is required (e.g. Rule Interchange Format recommend by W3C).

6 CONCLUSIONS

We have proposed a generic methodology that aims to isolate the specification of DCOs from the particularities of (concrete) data sources (namely of their model and/or schema) and, therefore, is targeted for reusing DCOs on different and diverse data sources. For this, the proposed methodology relies on the ability:

- To abstract the model and schemas of concrete data sources into a conceptual level comprising (i) a conceptualization of a domain of interested (e.g. customers orders) for which the DCOs will be specified and (ii) a vocabulary capturing different types of DCO and representing their structure and semantics in a way that is independent of any domain of interest;
- To rewrite DCOs specified for a particular domain of interest and according to the aforementioned vocabulary to be in accordance with the model, schema and any other requirements of a concrete data source.

Regarding the domain conceptualization and DCO vocabulary, this paper proposes the adoption of RDF/OWL ontologies because: (i) ontologies are seen as the best answer for the demand for intelligent systems that operate closer to the human conceptual level (Obrst et al., 2003); and, (ii) ontologies conceptualizations may vary in different levels of expressiveness (e.g. *ALC* vs. *SHOIN*).

Yet, as discussed and exemplified in section 5, these abilities are successfully accomplished through the establishment of mappings between the conceptual elements (e.g. ontological concepts) of the conceptualization of the domain of interest and the schema elements of a concrete data source (e.g. tables of a relational database) and, therefore, giving a semantic meaning to the schema elements. Based on these mappings is then possible to instantiate the DCOs previously specified at the conceptual level and propose or suggest their execution to the domain expert at the schema level of a concrete data source.

A key point of the proposed data cleaning methodology is the need of a DCO Vocabulary (DCOV). In this respect, this paper complements and refines our previous work, first, by specifying the requirements that the DCOV should satisfy (cf. section 3) and, second, by analysing candidate vocabularies on light of the defined requirements (cf. section 4). The requirements were specified via a use case model and a set of competency questions. Having these requirements as basis, an extensive literature review was made to identify possible candidates vocabularies. Consequently, we found that the DQM ontology (Fürber and Hepp, 2011) was able to cover most of the DCOV requirements. The main elements of the DQM ontology were introduced and discussed in the paper at the light of the requirements. The purpose was to demonstrate the DQM ontology was suitable for our purpose. Due to its early stage of development we have concluded that some elements are missing in the DQM ontology. However, we also have concluded that, as shown in section 4, missing elements can easily be added by extending the DQM ontology. As so, this ontology was adopted as the DCOV to support the conceptual specification of the DCOs.

As future work, we intend to devote our attention to the correction DCOs, since we have been focused on the detection DCOs. We also plan to refine/specify further the bridging process between the abstract and concrete data layers. We have already started doing this work, as can be seen in this paper (cf. Section 5). Currently, this methodology is being implemented and tested in some case studies using real-word data.

Despite the proposed methodology is targeted for reusing DCOs, we recognized that some operations may be too specific of a data source (e.g. detection of syntax violations in the product codes of a company) and, therefore, chances of being useful in another data source are diminish, which can be seen as a serious limitation. For this kind of DCOs, the traditional approach (i.e. specifying DCOs at the schema level of the concrete data source) is still the best option. However, our practical experience of dealing with different data sources from several domains has allowed us to conclude that: (i) there is much data cleaning knowledge which can be applied to different data sources from the same domain (e.g. detection of business rules violations); (ii) there is data cleaning knowledge which is so general that can even be applied to data sources of different domains (e.g. detection of a syntax violation in an e-mail address). Hence, the proposed methodology intends to explore both realities and promote the reuse of data cleaning knowledge.

ACKNOWLEDGEMENTS

This work is supported by FEDER Funds through the "Programa Operacional Fatores de Competitividade - COMPETE" program and by National Funds through FCT "Fundação para a Ciência e Tecnologia".

REFERENCES

Almeida, R., Maio, P., Oliveira, P., João, B., 2015. Towards Reusing Data Cleaning Knowledge, in: New Contributions in Information Systems and Technologies. Springer, pp. 143–150.

Almeida, R., Oliveira, P., Braga, L., Barroso, J., 2012. Ontologies for Reusing Data Cleaning Knowledge, in: Semantic Computing (ICSC), 2012 IEEE Sixth Int. Conf. on. IEEE, pp. 238–241.

Arenas, M., Bertails, A., Prud'hommeaux, E., Sequeda, J., 2012. A direct mapping of relational data to RDF.

Atzori, L., Iera, A., Morabito, G., 2010. The internet of things: A survey. Computer networks 54, 2787–2805.

Bellahsene, Z., Bonifati, A., Rahm, E. (Eds.), 2011. Schema Matching and Mapping. Springer Berlin Heidelberg, Berlin, Heidelberg.

Booch, G., 1993. Object-Oriented Analysis and Design with Applications, 2 edition. ed. Addison-Wesley Professional, Redwood City, Calif.

Brickley, D., Guha, R.V., 2014. RDF Schema 1.1 [WWW Document]. URL http://www.w3.org/TR/2014/REC-rdf-schema-20140225/

Codd, E.F., 1970. A relational model of data for large shared data banks. Communications of the ACM 13, 377–387.

Das, S., Sundara, S., Cyganiak, R., 2012. R2RML: RDB to RDF mapping language.

Dasu, T., Vesonder, G.T., Wright, J.R., 2003. Data quality through knowledge engineering, in: Proceedings of the Ninth ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining. ACM, pp. 705–710.

Fürber, C., Hepp, M., 2011. Towards a vocabulary for data quality management in semantic web architectures, in: Proc. of the 1st Int. Workshop on Linked Web Data Management. ACM, pp. 1–8.

Han, J., Haihong, E., Le, G., Du, J., 2011. Survey on NoSQL database, in: Pervasive Computing and Applications (ICPCA), 2011 6th Int. Conf. on. IEEE, pp. 363–366.

Hashem, I.A.T., Yaqoob, I., Anuar, N.B., Mokhtar, S., Gani, A., Khan, S.U., 2015. The rise of "big data" on cloud computing: review and open research issues. Information Systems 47, 98–115.

Knuth, M., Sack, H., 2014. Data cleansing consolidation with PatchR, in: The Semantic Web: ESWC 2014 Satellite Events. Springer, pp. 231–235.

Maedche, A., Motik, B., Silva, N., Volz, R., 2002. MAFRA—A MApping FRAmework for Distributed Ontologies in the Semantic Web, in: Workshop on Knowledge Transformation for the Semantic Web (KTSW 2002), ECAI. pp. 60–68.

Makris, K., Bikakis, N., Gioldasis, N., Christodoulakis, S., 2012. SPARQL-RW: Transparent Query Access over Mapped RDF Data Sources, in: Proc. of the 15th Int. Conf.on Extending Database Technology, EDBT '12. ACM, New York, NY, USA, pp. 610–613. doi:10.1145/2247596.2247678

McGuinness, D., Harmelen, F. van, 2004. OWL Web Ontology Language Overview [WWW Document]. URL http://www.w3.org/TR/owl-features/ (accessed 6.11.15).

Milano, D., Scannapieco, M., Catarci, T., 2005. Using ontologies for xml data cleaning, in: On the Move to Meaningful Internet Systems 2005: OTM 2005 Workshops. Springer, pp. 562–571.

Obrst, L., Liu, H., Wray, R., 2003. Ontologies for corporate web applications. AI Magazine 24, 49.

Oliveira, P., Rodrigues, F., Henriques, P., 2009. SmartClean: An Incremental Data Cleaning Tool, in: Quality Software, 2009. QSIC'09. 9th Int. Conf. on. IEEE, pp. 452–457.

Oliveira, P., Rodrigues, F., Henriques, P., Galhardas, H., 2005a. A taxonomy of data quality problems, in: 2nd Int. Workshop on Data and Information Quality. pp. 219–233.

Oliveira, P., Rodrigues, F., Henriques, P.R., 2005b. A Formal Definition of Data Quality Problems., in: IQ. MIT.

Otero-Cerdeira, L., Rodríguez-Martínez, F.J., Gómez-Rodríguez, A., 2015. Ontology matching: A literature review. Expert Systems with Applications 42, 949–971. doi:10.1016/j.eswa.2014.08.032

Snijders, C., Matzat, U., Reips, U.-D., 2012. Big data: Big gaps of knowledge in the field of internet science. Int. Journal of Internet Science 7, 1–5.

Weis, M., Manolescu, I., 2007. Declarative XML data cleaning with XClean, in: Advanced Information Systems Engineering. Springer, pp. 96–110.