

# Towards a Vocabulary for Data Quality Management in Semantic Web Architectures

Christian Fürber

Universitaet der Bundeswehr Muenchen  
Werner-Heisenberg-Weg 39  
85577 Neubiberg  
+49 89 6004 4218

christian@fuerber.com

Martin Hepp

Universitaet der Bundeswehr Muenchen  
Werner-Heisenberg-Weg 39  
85577 Neubiberg  
+49 89 6004 4217

mhepp@computer.org

## ABSTRACT

Reliable decision-making and reliable information based on Semantic Web data requires methodologies and techniques for managing the quality of the published data. To make things more complicated, the judgment of what is “good” data will often depend on the task at hand or the subjective requirements of data owners or data consumers. Some data quality requirements can be modeled using data quality rules, i.e. executable definitions that allow the identification and measurement of data quality problems. In this paper, we provide a conceptual model that allows the representation of such rules and other quality-related knowledge using the Resource Description Framework (RDF) and the Web Ontology Language (OWL). Based on our model, it is possible to monitor and assess the quality of data sources and to automate data cleansing tasks. The use of a generic conceptual model based on Semantic Web formalisms supports the definition of reusable, broadly applicable SPARQL queries and portable applications for data quality management (DQM). Furthermore, the explicit representation of rules in RDF/OWL facilitates rule management tasks, e.g. for analyzing consistency among the rules, and allows to collaborate and create a shared understanding.

## Categories and Subject Descriptors

I.2.4 [Artificial Intelligence]: Knowledge Representation Formalisms and Methods – *Semantic networks*; E.m [Data]: Miscellaneous; J.1 [Computer Applications] Administrative Data Processing – *Business*; K.6.4 [Computing Milieux]: System Management – *Quality assurance*

## General Terms

Management, Measurement, Documentation, Verification,

## Keywords

Linked Data Management, Data Quality Management, Information Quality, Ontology, SPARQL, Semantic Web, Knowledge Representation, Trust

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

LWDM 2011, March 25, 2011, Uppsala, Sweden.

Copyright 2011 ACM 978-1-4503-0608-9/11/03 ...\$10.00.

## 1. INTRODUCTION

The management of data quality plays a crucial role in every information system. Poor data may lead to poor decisions and operational disturbances or even system failures [cf. 1]. Recently, researchers and businesses have started to create, publish, and interlink a lot of data on the World Wide Web as part of the Semantic Web initiative. The amount of available data on Web scale thereby promises a higher degree in automation, better integration, and higher reusability of data [cf. 2]. Due to the openness of the Web, however, anyone can publish and retrieve data. While this creates tremendous opportunities for future applications, it also increases the need for tools and standards that support the automation of data quality management (DQM) for such Semantic Web data. In particular, we need standards that facilitate (1) the identification of data quality problems, (2) the assessment of data quality for a given task or context, and (3) the correction of data quality problems. Unfortunately, these challenges have not yet been sufficiently addressed by Semantic Web research. In this paper, we propose a domain-independent, machine-readable conceptual model for DQM in the form of an ontology, i.e. a formal conceptualization of a domain, that allows the standardized formulation of data quality and data cleansing rules, the classification of data quality problems, and the computation of data quality scores for Semantic Web data sources. The standardized formulation of data quality rules in RDF/OWL thereby increases the transparency of quality requirements, facilitates consistency checks among multiple data quality requirements, simplifies the reuse of those rules at Web Scale, and eases the identification of shared quality requirements. Moreover, the ontology enables quality checks with a limited set of queries, since they only use terms from the DQM vocabulary. The classification part of the DQM ontology helps to keep track of previously identified quality problems. The assessment of data quality scores may help with the selection of the most appropriate data sources in a distributed setting, and with the generation of trust in the resulting data or decisions. Furthermore, our approach also allows defining data cleansing rules that help automate data cleansing tasks.

In the following sections, we will define the requirements for such an ontology, and explain its core conceptual elements. Finally, we present a preliminary evaluation of the ontology by showing how we can use the represented knowledge for DQM purposes.

## 2. Defining Data Quality

Data quality is often defined as data's "fitness for use" [1, 3]. Accordingly, the quality requirements of data typically depend on the intended usage of the data. Semantic Web data can serve various purposes, which are often unknown to the data publisher. This also means that the quality requirements may be very heterogeneous, which stresses the importance to expose data quality requirements in order to understand the underlying assumptions, e.g. when presenting data quality assessment results. Moreover, it is important to be aware of the type and nature of the data item that is subject to the quality evaluation, because different components in a semantic system, such as data values, data models, queries, reasoners, and user interfaces, may all influence the perception of quality. In this paper, we concentrate on the quality of data, i.e. the quality of instances and data values. Throughout this paper, we will use the term "instance" to refer to a data representation of an entity including the states of its properties, and the term "value" to refer to the state of a property value itself.

## 3. METHOD

The development of our DQM ontology is based on the ontology engineering methodology as proposed by *Uschold and Gruninger* [4]. This section describes the requirements for the DQM ontology via motivating scenarios and competency questions; two standard steps in ontology construction. In section 4, we will define the terms required to formulate the answers to our competency questions. In section 5, we describe how the ontology was formalized and which existing ontology elements have been adopted. In section 6, we present first results of our evaluation.

### 3.1 Motivating Scenarios

In the following, we describe typical scenarios for the application of a standard ontology for DQM:

**Scenario 1:** Data owners can collect, maintain, and populate data quality rules for their data in a structured way, so that queries can use them to derive data quality monitoring and data quality assessment reports. The data quality monitoring reports show the instances that violate the data quality rules. The data quality assessment reports give data quality scores based on the specified data quality rules. Moreover, they can represent data cleansing rules in a structured way, so that update queries can execute them to correct data defects automatically.

**Scenario 2:** Data consumers can select data quality rules of a specific person or for a specific task and add their own data quality rules via the DQM vocabulary to evaluate the quality state of a Semantic Web data source.

**Scenario 3:** Data owners and data consumers have an overview of all data quality rules specified for a data source. Furthermore, they can check the consistency among the specified rules and select rules according to certain rule properties.

**Scenario 4:** Data consumers can retrieve only such data that meets their individual quality requirements as specified using the DQM vocabulary.

Summarizing the motivating scenarios above, we need a means to represent quality-related knowledge in a way so that it can be processed by automated queries for (1) data quality monitoring, (2) data quality assessment, (3) data cleansing, and (4)

information filtering, while the applied quality requirements remain transparent to the user at all times and while their consistency can be checked easily.

### 3.2 Competency Questions

Based on the motivating scenarios and on a literature analysis of typical data quality problems [5-8], we specify the requirements for a DQM ontology via the following competency questions:

**CQ1:** Which instances of a data source suffer from data quality problems according to predefined data quality requirements?

**CQ1.1:** Which instances of a given class contain values for a given property that match given illegal values?

**CQ1.2:** Which instances of a given class contain values for a given property that do not match given legal values for that property?

**CQ1.3:** Which instances of a given class with value combinations in two or more given properties do not match the legal value combinations for given tuples of properties of another given class?

**CQ1.4:** Which instances of a given class have numeric values in a given property that fall outside a given closed or open interval of legal values?

**CQ1.5:** Which instances of a given class have values in a given property that contain illegal characters or illegal syntactical patterns?

**CQ1.6:** Which instances of a given class can be considered duplicates based on a given identity metric?

**CQ1.7:** Which instances of a given class have non-unique values in a given property that must only contain unique values?

**CQ1.8:** Which instances of a given class lack a value for a given mandatory property (e.g. empty string or null values)?

**CQ1.9:** Which instances of a given class lack a given mandatory property?

**CQ1.10:** Which instances of a given class are outdated?

**CQ2:** What is the data quality state of a selected data source according to predefined data quality requirements?

**CQ2.1:** How complete is the data for a given property in instances of a given class?

**CQ2.1.1:** How many empty literals for a given property exist in instances of a given class?

**CQ2.1.2:** How many instances within a given class exist that lack a given property?

**CQ2.1.3:** How many instances must have a literal value for a given property?

**CQ2.2:** How semantically accurate are the values of a given property in instances of a given class?

**CQ2.2.1:** How many instances of a given class have semantically incorrect values for a given property?

**CQ2.2.2:** How many instances of a given class are required to have semantically correct values for a given property?

**CQ2.3:** How syntactically accurate are the values of a given property in instances of a given class?

**CQ2.3.1:** How many instances of a given class have syntactically incorrect values for a given property?

**CQ2.3.2:** How many instances of a given class must have syntactically correct values for a given property?

**CQ2.4:** How current are the instances of a given class?

**CQ2.4.1:** How many instances of a given class are outdated?

**CQ2.4.2:** How many instances of a given class must be current?

**CQ2.5:** How unique are the values of a given property in instances of a given class?

**CQ2.5.1:** How many instances of a given class have non-unique values for a given property?

**CQ2.5.2:** How many instances exist in a given class that must have unique values for a given property?

**CQ3:** For which time-frame is the data quality rule valid?

**CQ4:** Which rules are created by Person X?

**CQ5:** Which rules are based on source Y?

**CQ6:** Which rules have a confidence level above XY?

**CQ7:** Which rules are task-dependent?

**CQ8:** Which rules apply for task A?

**CQ9:** When was the rule last modified?

**CQ10:** Which data quality problems affect instances of class B and/or values of property X?

**CQ11:** Which data quality rules can be used to filter data that meets the specified quality requirements?

**CQ12:** Which rules can be applied for data cleansing?

## 4. DOMAIN CAPTURE

Based on the requirements specified in the previous section, we describe the conceptual elements for the representation of DQM-relevant knowledge. Figure 1 shows an UML Class diagram that provides an overview of the conceptual elements. However, due to the early stage of the work, additional vocabulary elements, e.g. for the description of additional data quality rules, data cleansing rules, data quality problem types, or data quality dimensions, may be specified and added to the vocabulary as part of our future work.

### 4.1 Classes for Data Quality Rules

In this section, we describe the terms required to represent certain kinds of data quality rules. We identified the following data quality rule classes by analyzing typical data quality problem types on instance level as specified in [5-8].

**Data Quality Rule:** A data quality rule is an externally given directive or a consensual agreement that defines the content and/or structure that constitute high quality data instances and values.

**Duplicate Instance Rule:** A duplicate instance rule is a data quality rule that specifies the properties which (in combination) uniquely identify an entity. I.e. if the properties of two or more different instances represent the same state, then the instances represent the same entity. Thus, the instances are considered to be duplicates.

**Legal Value Rule:** A legal value rule is a data quality rule that specifies all values that a certain property is allowed to obtain. Legal value rules, therefore, refer to reference properties of classes that hold instances with all allowed values.

**Functional Dependency Rule:** A functional dependency rule is a data quality rule that specifies legal value combinations for two or more properties that are allowed to occur within the same instance. Functional dependency rules refer to reference properties of classes that hold instances with all allowed value combinations.

**Legal Value Range Rule:** A legal value range rule is a data quality rule that specifies the upper and/or lower boundary of numeric values that a certain property is allowed to obtain.

**Illegal Value Rule:** An illegal value rule is a data quality rule that specifies the values that a certain property must not contain. Illegal value rules, therefore, refer to reference properties that hold all disallowed values.

**Illegal Value Range Rule:** An illegal value range rule is a data quality rule that specifies the upper and/or lower boundary of valid numeric values.

**Outdated Instance Rule:** An outdated instance rule is a data quality rule that specifies the point in time when an instance is no longer current.

**Expiry Rule:** An expiry rule is an outdated instance rule that specifies classes which hold instances with an expiration date that must not exceed the current date and time.

**Update Rule:** An update rule is an outdated instance rule that specifies the maximum timespan in which an instance has to be updated.

**Property Completeness Rule:** A property completeness rule is an abstract data quality rule class for data quality rules that check the completeness of instances of a certain class regarding their property values.

**Missing Property Rule:** A missing property rule is a property completeness rule that specifies a certain property that must exist in instances of a certain class.

**Missing Literal Rule:** A missing literal rule is a property completeness rule that specifies a certain property that must have a literal value in instances of a certain class.

**Conditional Property Rule:** A conditional property rule is a property completeness rule that specifies a certain property that must exist in a certain subset of instances of a certain class.

**Conditional Literal Rule:** A conditional literal rule is a property completeness rule that specifies a certain property that must have literal values in a certain subset of instances of a certain class.

**Unique Value Rule:** A unique value rule is a data quality rule that specifies that each value of a certain property must be unique in instances of a certain class.

**Syntax Rule:** A syntax rule is a data quality rule that specifies the allowed characters and/or patterns for values for a certain property of instances of a certain class.

### 4.2 Properties of Data Quality Rules

In the previous section, we described the different classes of data quality rules. The properties for these classes can be roughly

separated into global properties, i.e. properties that are inherent to every data quality rule, and specific properties, i.e. properties that belong to a subset of data quality rules.

### 4.2.1 Global Properties

In the following, we describe the properties that are applicable for every data quality rule type.

**Creator of the Rule:** Every data quality rule is created by at least one agent. For provenance reasons, it is important to attach information about the rule's creator.

**Source of the Rule:** Data quality rules are usually based on information sources, such as real world perceptions, domain knowledge, policies, data standards, laws, personal or consensual decisions, etc. For provenance reasons, it is important to attach information about the rule's source.

**Tested Class:** The tested class property specifies the class that holds the instances that shall be tested for data quality problems.

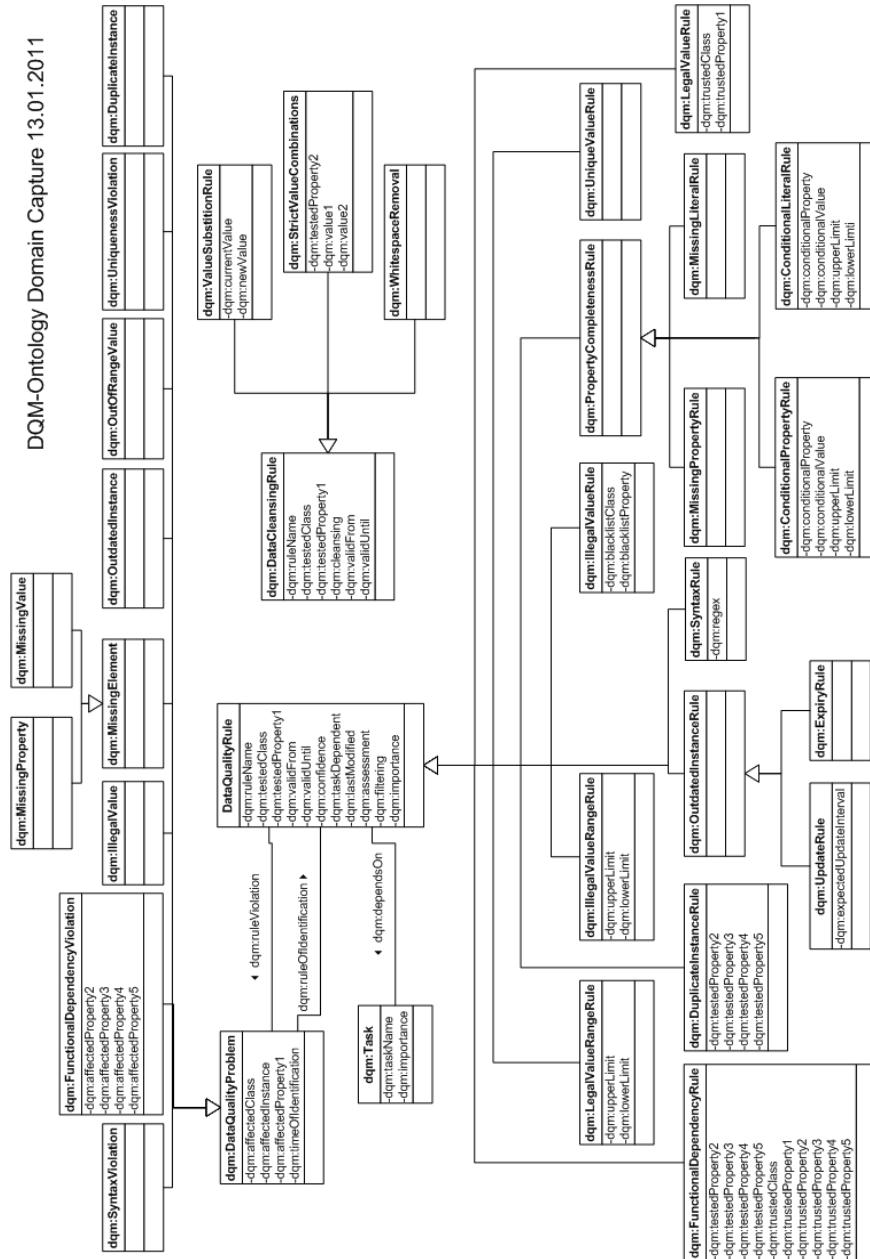


Figure 1. Overview about the DQM ontology<sup>1</sup>

<sup>1</sup> The class `dqm:DataQualityScores` and its elements are not shown in the diagram due to space limitations

**Tested Property:** The property that holds the values to be tested for data quality problems is called “Tested Property”.

**Validity:** Sometimes data quality rules may be valid only for a limited period of time. Thus, it must be possible to define a start and end time for the validity period of the rule.

**Confidence:** Sometimes rule creators are not sure about the correctness and completeness of data quality rules, while it must still be possible to express rules with uncertainty about their completeness and correctness. Thus, it should be possible to define a confidence level for each data quality rule indicating the confidence of the rule creator in the correctness and completeness of the rule.

**Last Modification:** Since data quality rules may change over time, it shall be possible to identify the time of the last modification.

**Relevancy for Assessment:** Not every data quality rule will be eligible for data quality assessment, e.g. due to immaturity, low precision, or lack of consensual understanding. Thus, it shall be possible to flag all rules whether they are relevant for data quality assessment.

**Relevancy for Information Filtering:** Some data quality rules may be used to retrieve only data that meets the previously specified quality requirements. Thus, a property shall specify whether the data quality rule shall be used for information filtering.

#### 4.2.2 *Specific Properties*

In the following, we describe the properties that belong to a subset of data quality rule classes:

**Trusted Class:** Legal value rules and functional dependency rules require the specification of the trusted class as a reference that holds instances with legal values / legal value combinations.

**Trusted Property:** A trusted property holds the values that serve as a trusted reference in legal value rules / functional dependency rules, e.g. to define legal values for a tested property.

**Blacklist Class:** A blacklist class holds instances with values that are disallowed for a certain data set. Thus, illegal value rules refer to such classes over the blacklist class property.

**Blacklist Property:** A blacklist property holds the values that are disallowed for a certain data set.

**Regular Expression:** Regular expressions are a common means to express character ranges and other syntactic patterns. Thus, they can be used to express syntax rules and are, therefore, an important property of the syntax rules.

**Expected Update Interval:** Update rules require the specification of an expected update interval which indicates the maximum duration between two data updates for instances of a class in order to be current.

**Conditional Property:** Conditional property / literal rules only apply to a subset of instances of a specific class. To localize the relevant subset of the rule, it is necessary to specify the conditional property that holds the values that can be used to filter the relevant instances for the rule. This property is called a conditional property.

**Conditional Value:** The value which can be used to select the relevant instances for conditional property / literal rules is specified via the conditional value property.

**Upper / Lower Limit:** Legal / illegal value ranges can be determined via an upper and/or lower boundary to check whether the value lies inside or outside of the specified interval. Moreover, upper and lower limits may be used to determine a subset of all instances of a specific class via the values of a conditional property in cases of conditional completeness rules.

### 4.3 Data Quality Problem Classification

In the following, we describe the classes that can be used to classify data quality problems that can be discovered by data quality rules specified in the previous section.

**Data Quality Problem:** A data quality problem occurs when a data value or a data instance does not meet the quality requirements.

**Syntax Violation:** A syntax violation is a data quality problem that occurs when a data value contains disallowed characters or does not match a predefined pattern.

**Functional Dependency Violation:** A functional dependency violation is a combination of different property values within the same instance that must not occur together. E.g. an instance describing the man “Peter Miller” has the value “Mr.” as its salutation property, but the value “female” as its gender.

**Illegal Value:** An illegal value is a data value that must not be used for a property.

**Missing Element:** A missing element is a data quality problem that occurs when schema elements, instances, or data values are missing, while required.

**Missing Property:** A property is missing when an instance does not contain a specific property that is required.

**Missing Value:** A missing value occurs when a property of an instance does not hold a value. This is very common in Semantic Web data that was automatically generated from existing sources in a template-based approach.

**Outdated Instance:** An instance is outdated when it represents an outdated state of its corresponding real-world entity.

**Out Of Range Value:** A value is out of range when it is not part of the legal value range or when it is part of the illegal value range.

**Duplicate Instance:** Two or more instances are duplicates when they represent the same real world entity.

**Uniqueness Violation:** A uniqueness violation occurs when two or more identical values are assigned to a property that requires unique values.

### 4.4 Properties of Data Quality Problems

In the following, we describe the properties required for the description of data quality problems.

**Affected Instance:** An affected instance is the data instance that contains one or more data quality problems.

**Affected Class:** An affected class holds one or more affected instances.

**Affected Property:** An affected property specifies the property of a data instance that contains one or more data quality problems. Some data quality problems, such as functional dependency violations, have more than one affected property.

**Time of Identification:** The time of identification specifies the date and time when the data quality problem was identified.

## 4.5 Classes for Data Quality Assessment

Data quality assessment in the understanding of this paper is the “process of assigning numerical and categorical values to information quality (IQ) dimensions” [9]. IQ dimensions in our understanding are thereby data quality dimensions that can be measured using data quality rules. Based on a summary by *Batini et al.* [10], we identified five basic dimensions that can be used to classify data quality scores which are assessable using the data quality rules specified above.

**Data Quality Assessment:** Data quality assessment is an abstract class for data quality dimensions that can be used to structure scores that indicate the quality state of classes and/or properties.

**Completeness:** Completeness is the “extent to which data are of sufficient breadth, depth, and scope for the task at hand” [3]. In terms of the Semantic Web we distinguish between property completeness and population completeness.

**Property Completeness:** Property completeness is the degree to which values for a specific property or the property itself are not missing in entities of a specific class [cf. 11, 12].

**Population Completeness:** Population completeness is the degree to which all objects of a certain reference are represented in a specific class [cf. 11, 12].

**Accuracy:** Accuracy in the understanding of this paper is the degree to which a data value represents the desired state regarding syntax and semantics.

**Syntactic Accuracy:** Syntactic accuracy is the degree to which data values of a property represent legal values and are free from syntax violations.

**Semantic Accuracy:** Semantic accuracy is the degree to which the data values of an instance represent the correct state of an entity’s property.

**Uniqueness:** Uniqueness is the degree to which properties and classes are free from duplicate values and instances.

**Entity Uniqueness:** Entity uniqueness is the degree to which entities (that must be uniquely represented within a certain class) are unique.

**Property Uniqueness:** Property uniqueness is the degree to which the values of a property (that must only contain unique values within instances of a certain class) are unique.

**Timeliness:** Timeliness in the understanding of this paper is the degree to which instances of a specific class (1) are updated within an expected time or (2) have not exceeded their expiration date.

## 4.6 Properties for Data Quality Assessment

In the following, we describe the properties of data quality scores.

**Assessed Class:** The assessed class holds the instances that have been analyzed to compute the assessment score.

**Assessed Property:** The assessed property represents all properties that have been tested for rule violations during the computation of the assessment score.

**Plain Score:** The plain score represents a data quality dimension score that was assessed by using a certain formula that accounts for the violations of data quality rules related to the size of the relevant data set. It can be computed by using the simple ratio which is calculated by subtracting the ratio between the total number of rule violations and the total number of relevant instances from one. The result ranges from 0 to 1 with 0 representing the worst quality state and 1 representing a perfect quality state of the investigated data items [12].

**Weighted Score:** A weighted score is a data quality dimension score that is computed by the integration of the importance of the data quality rules for the task at hand and/or by the importance of the task the data is used for.

**Time of Assessment:** The time of assessment represents the date and time when the data quality score was computed.

## 4.7 Data Cleansing Rules

In cases with unambiguous rules, it is possible to define data cleansing rules that can be used to automatically update all incorrect values in the original data source. In the following, we describe the classes and properties required for representing data cleansing rules.

**Data Cleansing Rule:** A data cleansing rule is an unambiguous rule that precisely specifies the required state of a data value.

**Strict Value Combination Rule:** A strict value combination is a combination of two values of different properties that may only be assigned to each other, but not to other values.

**Whitespace Removal Rule:** A white space removal rule states that whitespaces at the beginning of a string and at the end of a string shall be removed for the specified property.

**Value Substitution Rule:** A value substitution rule specifies a value to be removed and a new value that shall substitute the removed value.

**Data Cleansing:** The data cleansing property specifies whether the rule shall be applied to cleanse the data in the data source.

**Current / New Value:** The current / new value properties specify the value to be removed and the new value of value substitution rules.

Furthermore, data cleansing rules inherit the properties (1) rule name, (2) tested class, (3) tested property, and (4) validity from the data quality rules class.

## 4.8 Task Dependency

Metrics for data quality assessment can be task-dependent or task-independent. Task-independent rules are of general applicability, while task-dependent rules are only appropriate for certain tasks [12]. Hence, it must be possible to model a relationship between a data quality rule and its relevant tasks in cases where the rule is task-dependent. Moreover, the vocabulary must provide means to clearly identify task-independent rules.

## 5. ONTOLOGY CODING

We chose OWL<sup>2</sup> for the design of the ontology, since we aim to provide a vocabulary for DQM in Semantic Web Architectures, want to be able to use OWL reasoners, and since OWL is widely used in the Semantic Web. We based the formalization of the ontology on the results of our domain capture. The UML class diagram depicted in figure 1 provides an overview of the DQM ontology, which currently consists of 45 classes and 56 properties. As a prerequisite, we avoided designing an OWL Full ontology, so that reasoning will still be decidable when the ontology is used. This forced us to create several datatype properties with datatype `xsd:anyURI`, since DQM requires to use class and property URIs as objects, which is otherwise not allowed in OWL DL. With this workaround we are able to express statements about ontology classes and properties, without disturbing the performance of reasoning. Before we created the DQM ontology, we analyzed existing ontologies for reusable classes and properties. For the description of rule creators, we suggest to combine the DQM vocabulary with the Provenance Vocabulary<sup>3</sup>, in particular with its data creation classes and properties. Alternatively, we can use the property `dc:creator` from the Dublin Core Metadata ontology<sup>4</sup>. In addition, we suggest to use the property `dc:source` for the description of the rule source.

```

INSERT{
  _:b0 a dqm:SyntaxViolation ;
    dqm:affectedClass ?class;
    dqm:affectedInstance ?s2;
    dqm:affectedProperty1 ?prop;
    dqm:timeOfIdentification ?time;
    dqm:ruleOfIdentification ?s .
  ?s dqm:ruleViolation _:b0 .
}
WHERE {
  ?s a dqm:SyntaxRule .
  OPTIONAL{
    ?s dqm:testClass ?class}
  ?s dqm:testProperty1 ?prop ;
    dqm:regex ?regex .
  LET(?propURI := smf:buildURI(?prop)) .
  LET(?classURI := smf:buildURI(?class)) .
  OPTIONAL{
    ?s2 a ?classURI ;
      ?propURI ?value .
  }
  FILTER(!regex(str(?value),?regex)) .
  LET(?time:=afn:now())
}

```

Figure 2. SPARQL query to classify syntax violations

All other elements have been created in the `dqm:` namespace, since we could not find appropriate vocabulary elements in available ontologies. Regarding the representation of task dependency, the data quality management ontology is compatible to any task ontology, i.e. ontologies that describe tasks and processes. Such ontologies can be connected by using the object

<sup>2</sup> <http://www.w3.org/TR/owl-features/>

<sup>3</sup> <http://purl.org/net/provenance/ns#>

<sup>4</sup> <http://www.dublincore.org/2010/10/11/dcelements.rdf>

property `dqm:dependsOn` of the domain class `dqm:DataQualityRule`. Alternatively, we created the class `dqm:Task` with properties that represent relevant information for DQM for cases without an appropriate task ontology.

## 6. PRELIMINARY EVALUATION

We performed a preliminary evaluation of our approach using a small test data set containing artificially created data quality problems. We integrated the test data set with our DQM ontology and defined data quality rules about the classes and properties of the test data set. Also, we executed data quality problem classification queries over the data set. The query depicted in figure 2 classifies all syntax violations and adds RDF triples to the local graph. In a second step, we executed a data quality assessment query depicted in figure 3 that uses only explicitly flagged syntax rules to calculate syntactic accuracy. It is also imaginable to select only rules from a certain creator or a certain task above a certain confidence level for that purpose. The SPARQL queries thereby only depend on vocabulary elements of the DQM ontology, so that the queries are applicable on any data set. Due to the use of OWL datatype properties, we had to make the URI's dereferencable again using a SPARQL extension function.

```

INSERT{
  _:b0 a dqm:SyntacticAccuracy ;
    dqm:numberOfViolations ?violations;
    dqm:totalTestedInstances ?total;
    dqm:plainScore ?plainscore;
    dqm:timeOfAssessment ?time;
    dqm:assessedClass ?class;
    dqm:assessedProperty ?prop.
}
WHERE{
  OPTIONAL{SELECT (COUNT(DISTINCT(?s2)) AS ?total) ?prop ?class
  WHERE {
    ?s a dqm:SyntaxRule;
      dqm:assessment "true"^^xsd:boolean .
    OPTIONAL{
      ?s dqm:testClass ?class}
    ?s dqm:testProperty1 ?prop ;
      dqm:assessment ?assessment;
    LET(?propURI := smf:buildURI(?prop)) .
    LET(?classURI := smf:buildURI(?class)) .
    ?s2 a ?classURI ;
      ?propURI ?value .
  }
  GROUP BY ?class ?prop}
  OPTIONAL{SELECT (COUNT(DISTINCT(?s2)) AS ?violations) ?prop ?class
  WHERE {
    ?s a dqm:SyntaxRule;
      dqm:assessment "true"^^xsd:boolean .
    OPTIONAL{
      ?s dqm:testClass ?class}
    ?s dqm:testProperty1 ?prop ;
      dqm:assessment ?assessment;
      dqm:regex ?regex .
    LET(?propURI := smf:buildURI(?prop)) .
    LET(?classURI := smf:buildURI(?class)) .
    OPTIONAL{
      ?s2 a ?classURI ;
        ?propURI ?value .
    }
    FILTER(!regex(str(?value),?regex)) .
  }
  GROUP BY ?class ?prop}
  LET(?plainscore := ((?total - ?violations)/?total)) .
  LET(?time:=afn:now())
}

```

Figure 3. Computation of syntactic accuracy for all classes and properties specified in instances of `dqm:SyntaxRule`



## 7. RELATED WORK

At present, we do not know of a comprehensive standard vocabulary to represent DQM information in Semantic Web environments. However, there are a few approaches that are related. The SPARQL Inferencing Notation (SPIN) provides a vocabulary that allows the representation of SPARQL queries in RDF. In [13] we have shown how SPIN can be used to represent data quality rules. Although SPIN is very useful for representing SPARQL-based rules in RDF, a specialized vocabulary for DQM provides more guidance and is better fitted to the requirements. The Web Information Quality Assessment Policy Language (WIQA-PL) is based on SPARQL grammar and facilitates the definition of information filtering policies allowing data consumers to retrieve only information that meets their quality requirements expressed in WIQA-PL [16]. The major drawback of this approach is that the quality requirements are not represented in RDF itself. Hence, the management of the policies may become very complex as the amount of policies is growing. Moreover, it requires a framework that can deal with WIQA-PL. Recently there have been a couple of provenance vocabularies published that shall provide metadata for data quality assessment. For instance, Hartig and Zhao published the Provenance Vocabulary<sup>1</sup> and showed how to assess timeliness with the help of provenance information [14]. The Provenance Vocabulary is focused on representing metadata about the creation and access of data rather than directly representing quality requirements. The DQM Ontology could complement the Provenance Vocabulary and might cover most information required for the holistic management of data quality. Finally, the Object Management Group has published a vocabulary for representing the Semantics of Business Vocabulary and Business Rules (SBVR) to facilitate the standard expression of policies and compliance rules [15]. Due to its focus on business rules, SBVR misses vocabulary elements that are specifically required for DQM.

## 8. Conclusions and Outlook on Future Work

In this paper, we have proposed a generic vocabulary that facilitates the representation of knowledge required for data quality monitoring, data quality assessment, data cleansing, and quality-driven data retrieval in Semantic Web architectures. The vocabulary was designed so that standard SPARQL queries can process the knowledge to automate these activities and, therefore, reduce manual effort. Due to the standardized representation of quality requirements in RDF/OWL, it is possible to capture, manage, and share data quality requirements and check their consistency. This enables us to create a consensual understanding about data quality and helps reusing data quality knowledge. Due to its early stage, the current version of the DQM ontology may still miss important elements. However, we will continue the application of the proposed vocabulary on real-world settings to discover further extensions and improvements. Moreover, it is planned to publish the DQM ontology at <http://semwebquality.org> for public usage.

## 9. REFERENCES

[1] Redman, T. C. 2001. *Data quality : the field guide* Digital Press, Boston.

- [2] Berners-Lee, T., Hendler, J., and Lassila, O. 2001. The Semantic Web, *Scientific American*, vol. 284, no. 5, 34-43.
- [3] Wang, R. Y., and Strong, D. M. 1996. Beyond accuracy: what data quality means to data consumers, *Journal of Management Information Systems*, vol. 12, no. 4, 5-33.
- [4] Uschold, M., and Gruninger, M. 1996. Ontologies: Principles, Methods, and Applications, *The Knowledge Engineering Review*, vol. 11, no. 2, 93-155.
- [5] Leser, U., and Naumann, F. 2007. *Informationsintegration : Architekturen und Methoden zur Integration verteilter und heterogener Datenquellen* dpunkt-Verl., Heidelberg.
- [6] Oliveira, P., Rodrigues, F., and Henriques, P. R. 2005 A Formal Definition of Data Quality Problems. In *International Conference on Information Quality (MIT IQ Conference)* (Cambridge, MA, USA, 2005).
- [7] Oliveira, P., Rodrigues, F., Henriques, P. R. et al. 2005 A Taxonomy of Data Quality Problems. In *2nd Int. Workshop on Data and Information Quality (in conjunction with CAiSE'05)* (Porto, Portugal, 2005).
- [8] Rahm, E., and Do, H.-H. 2000. Data Cleaning: Problems and Current Approaches, *IEEE Data Engineering Bulletin*, vol. 23, no. 4, 3-13.
- [9] Ge, M., and Helfert, M. 2008 Data and Information Quality Assessment in Information Manufacturing Systems. In *11th International Conference on Business Information Systems (BIS2008)* (Innsbruck, Austria, 2008). 380-389.
- [10] Batini, C., Cappiello, C., Francalanci, C. et al. 2009. Methodologies for data quality assessment and improvement, *ACM Comput. Surv.*, vol. 41, no. 3, 1-52. DOI=<http://doi.acm.org/http://doi.acm.org/10.1145/1541880.1541883>.
- [11] Batini, C., and Scannapieco, M. 2006. *Data quality : concepts, methodologies and techniques* Springer, Berlin.
- [12] Pipino, L. L., Lee, Y. W., and Wang, R. Y. 2002. Data quality assessment, *Commun. ACM*, vol. 45, no. 4, 211-218. DOI=<http://doi.acm.org/http://doi.acm.org/10.1145/505248.506010>.
- [13] Fürber, C., and Hepp, M. 2010 Using Semantic Web Resources for Data Quality Management. In *17th International Conference on Knowledge Engineering and Knowledge Management (EKAW2010)* (Lisbon, Portugal, 2010). 211-225.
- [14] Hartig, O., and Zhao, J. 2009 Using Web Data Provenance for Quality Assessment. In *First International Workshop on the role of Semantic Web in Provenance Management (Co-located with the 8th International Semantic Web Conference, ISWC-2009)* (Washington D.C., USA., 2009).
- [15] Semantics of Business Vocabulary and Business Rules (SBVR)," Object Management Group
- [16] Bizer, C. and Cyganiac, R. 2009, Quality-Driven Information Filtering using the WIQA Policy Framework, *Web Semantics*, Vol. 7, Issue 1.