

# The DAQUINCIS Broker: Querying Data and Their Quality in Cooperative Information Systems

Massimo Mecella<sup>1</sup>, Monica Scannapieco<sup>1,2</sup>, Antonino Virgillito<sup>1</sup>,  
Roberto Baldoni<sup>1</sup>, Tiziana Catarci<sup>1</sup>, and Carlo Batini<sup>3</sup>

<sup>1</sup> *Università di Roma “La Sapienza”*

*Dipartimento di Informatica e Sistemistica*

Via Salaria 113, I-00198 Roma, Italy

{mecella,monscan,virgi,baldoni,catarci}@dis.uniroma1.it

<sup>2</sup> *Consiglio Nazionale delle Ricerche*

*Istituto di Analisi dei Sistemi ed Informatica*

<sup>3</sup> *Università di Milano “Bicocca”*

*Dipartimento di Informatica, Sistemistica e Comunicazione*

Edificio U7, Via Bicocca degli Arcimboldi 8, I-20126 Milano, Italy

batini@disco.unimib.it

**Abstract.** *In cooperative information systems, the quality of data exchanged and provided by different data sources is extremely important. A lack of attention to data quality can imply data of low quality to spread all over the cooperative system. At the same time, improvement can be based on comparing data, correcting them and disseminating high quality data. In this paper, a framework and a related architecture for managing data quality in cooperative information systems is proposed, as developed in the context of the DAQUINCIS research project. Then the focus concentrates (i) on an XML-based model for data and quality data, and (ii) on the design of a broker, which selects the best available data from different sources; such a broker also supports the improvement of data based on feedbacks to data sources. The broker is the basic component of the DAQUINCIS architecture.*

## 1 Introduction

A *Cooperative Information System (CIS)* is a large scale information system that interconnects various systems of different and autonomous organizations, geographically distributed and sharing common objectives. Among the different resources that are shared by organizations, data are fundamental; in real world scenarios, an organization  $\mathcal{A}$  may not request data from an organization  $\mathcal{B}$  if it does not “trust”  $\mathcal{B}$  data, i.e., if  $\mathcal{A}$  does not know that the quality of the data that  $\mathcal{B}$  can provide is high. As an example, in an *e-Government* scenario in which public administrations cooperate in order to fulfill service requests from citizens and enterprises [1], administrations very often prefer asking citizens for data, rather than other administrations that have stored the same data, because the

quality of such data is not known. Therefore, lack of cooperation may occur due to lack of quality certification.

Uncertified quality can also cause a deterioration of the data quality inside single organizations. If organizations exchange data without knowing their actual quality, it may happen that data of low quality spread all over the CIS.

On the other hand, CIS's are characterized by high data replication, i.e., different copies of the same data are stored by different organizations. From a data quality perspective this is a great opportunity: improvement actions can be carried out on the basis of comparisons among different copies, in order either to select the most appropriate one or to reconcile available copies, thus producing a new improved copy to be notified to all involved organizations.

In this paper, we introduce a framework and an overall architecture for managing data quality in CIS's, as developed in the context of the DAQUINCIS research project<sup>1</sup>. The architecture aims at avoiding dissemination of low qualified data through the CIS, by providing a support for data quality diffusion and improvement. At the best of our knowledge, this is a novel contribution in the information quality area, that aims at integrating, in the specific context of CIS, both modeling and architectural issues. To enforce this vision, the current work, beside presenting the general architecture, focuses on two specific elements of the problem, that we consider of primary importance. More specifically:

- we first face the problem of lack of quality certification by proposing a model for each organization to export data with associated quality information. The model is XML-based in order to address interoperability issues existing in cooperative information systems;
- then, we present the distributed design of a single architectural element, based on the model cited above, namely the Data Quality Broker, which allows each organization involved in the CIS to retrieve data specifying their quality requirements. The Data Quality Broker offers only data that satisfy the given requirements and notifies organizations about the highest quality values found within the CIS. Its design takes into account reliable communication issues and shows the feasibility of our approach in practical scenarios.

The structure of the paper is as follows. In Section 2, the proposed framework for Cooperative Information Systems is introduced, and the DAQUINCIS architecture specifically addressing quality related issues is described. In Section 3, a model for both the exchanged data and their quality is presented. In Section 4 the Data Quality Broker is described and its distributed design is discussed. In Section 5, related research work is discussed and compared, and finally Section 6 concludes the paper by drawing future work and possible extensions.

---

<sup>1</sup> “DAQUINCIS – Methodologies and Tools for Data Quality in Cooperative Information Systems” is an Italian research project carried out by Università di Roma “La Sapienza”, Università di Milano “Bicocca” and Politecnico di Milano (<http://www.dis.uniroma1.it/~dq/>).

## 2 A Cooperative Architecture for Data Quality

### 2.1 Definition of Cooperative Information System

In current business scenarios, organizations need to cooperate in order to offer services to their customers and partners. Organizations that cooperate have business links (i.e., relationships, exchanged documents, resources, knowledge, etc.) connecting each other. Specifically, organizations exploit business services (e.g., they exchange data or require services to be carried out) on the basis of business links, and therefore the network of organizations and business links constitutes a cooperative business system.

As an example, a supply chain, in which some enterprises offer basic products and some others assemble them in order to deliver final products to customers, is a cooperative business system. As another example, a set of public administrations which need to exchange information about citizens and their health state in order to provide social aids, is a cooperative business system derived from the Italian *e-Government* scenario [1].

A cooperative business system exists independently of the presence of a software infrastructure supporting electronic data exchange and service provisioning. Indeed cooperative information systems are software systems supporting cooperative business systems; in the remaining of this paper, the following definition of CIS is considered:

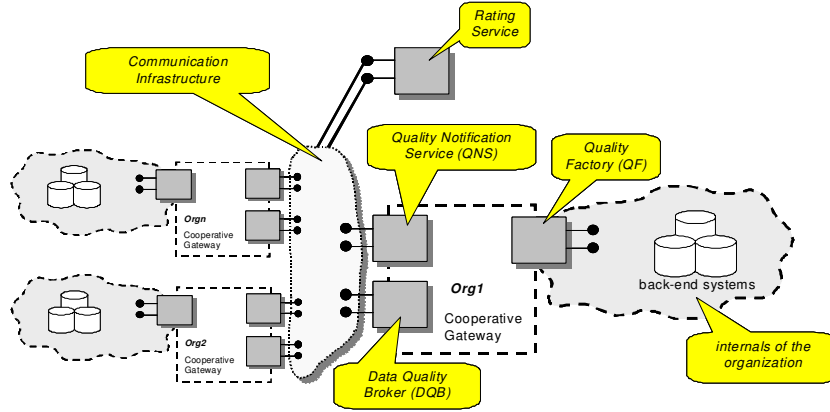
*A cooperative information system is formed by a set of organizations  $\{ Org_1, \dots, Org_n \}$  which cooperate through a communication infrastructure  $\mathbb{N}$ , which provide software services to organizations as wells as reliable connectivity. Each organization  $Org_i$  is connected to  $\mathbb{N}$  through a gateway  $G_i$ , on which services offered by  $Org_i$  to other organizations are deployed.*

### 2.2 The Architecture for Data Quality

A typical feature of CIS's is the high degree of data replicated in different organizations. As an example, in an *e-Government* scenario, the personal data of a citizen are stored by almost all administrations. On the basis of the proposed definition of CIS, more than one organization can implement the same service providing access to data; but several organizations can provide the same data though with different quality levels. Any requester of data may want to have the data with the highest quality level, among the provided ones. Thus only the highest quality data are returned to the requester, limiting the dissemination of low quality data. Moreover, the comparison of the gathered data values can be used to enforce a general improvement of data quality in all organizations.

In the context of the DAQUINCIS project, we propose an architecture for the management of data quality in CIS's; such an architecture allows the diffusion of data and related quality and exploits data replication to improve the overall quality of cooperative data. According to the logical model of a CIS presented in the previous section, we need to define both a model for the organizations

to exchange data and quality data and a set of services that realize quality management functions.



**Fig. 1.** An architecture for data quality diffusion and improvement

The model for data quality we propose in this paper is called *Data and Data Quality ( $D^2Q$ ) model*. It includes the definitions of (i) constructs to represent data, (ii) a common set of data quality properties, (iii) constructs to represent them and (iv) the association between data and quality data. The  $D^2Q$  model is described in Section 3.

In order to produce data and quality data according to the  $D^2Q$  model, each organization holds a **Quality Factory** that is responsible for evaluating the quality of its own data. The overall architecture is depicted in Figure 1. In the following we give a description of each element:

- **Data Quality Broker:** it is the core of the architecture. It performs, on behalf of a requesting organization, a data request on all organizations, also specifying a set of quality requirements that the desired data have to satisfy (*quality brokering function*). Different copies of the same data received as responses to the request are reconciled and a best-quality value is selected and proposed to organizations, that can choose to discard their data and adopt higher quality ones (*quality improvement function*). Quality brokering and improvement are described in Section 4. If the requirements specified in the request cannot be satisfied, then the broker initiates a negotiation with the requester that can optionally weaken the constraints on the desired data. The Data Quality Broker is in essence a data integration system [2] which allows to pose quality-enhanced query over a global schema and to select data satisfying such requirements.
- **Quality Notification Service:** it is a publish/subscribe engine used as a general message bus between architectural components and/or organizations.

More specifically, it allows quality-based subscriptions for organizations to be notified on changes of the quality of data. For example, an organization may want to be notified if the quality of data it uses degrades below a certain acceptable threshold, or when high quality data are available.

- **Rating Service:** it associates trust values to each data source in the CIS. These are used to determine the reliability of the quality evaluation performed by organizations. In this paper, for sake of simplicity, we assume that all organizations are reliable, i.e., provide trustable values with respect to quality of data.

In this paper, we only focus on the architectural design of the data quality broker, which is detailed in Section 4. The reader interested in other elements can refer to [3–5].

### 3 The $D^2Q$ Model

All cooperating organizations export their application data and quality data (i.e., data quality dimension values evaluated for the application data) according to a specific data model. The model for exporting data and quality data is referred to as *Data and Data Quality ( $D^2Q$ ) model*. In this section, we first introduce the data quality dimensions used in this paper (Section 3.1), then we describe the  $D^2Q$  model with respect to the data features (Section 3.2) and the quality features (Section 3.3).

#### 3.1 Data Quality Dimensions

Data quality dimensions are properties of data such as correctness or degree of updating. The data quality dimensions used in this work concern only data values; instead, they do not deal with aspects concerning quality of logical schema and data format [6].

In this section, we propose some data quality dimensions to be used in CIS's. The need for providing such definitions stems from the lack of a common reference set of dimensions in the data quality literature [7] and from the peculiarities of CIS's; we propose four dimensions, inspired by the already proposed definitions, and stemming from real requirements of CIS's scenarios that we experienced [1].

In the following, the general concept of *schema element* is used, corresponding, for instance, to an entity in an Entity-Relationship schema or to a class in a Unified Modeling Language diagram. We define: (i) accuracy, (ii) completeness, (iii) currency, and (iv) internal consistency.

**Accuracy** In [6], accuracy refers to the proximity of a value  $v$  to a value  $v'$  considered as correct. More specifically:

*Accuracy is the distance between  $v$  and  $v'$ , being  $v'$  the value considered as correct.*

Let us consider the following example: **Citizen** is a schema element with an attribute **Name**, and **p** is an instance of **Citizen**. If **p.Name** has a value  $v = \text{JHN}$ , while  $v' = \text{JOHN}$ , this is a case of low accuracy, as **JHN** is not an admissible value according to a dictionary of English names.

Accuracy can be checked by comparing data values with reference dictionaries (e.g., name dictionaries, address lists, domain related dictionaries such as product or commercial categories lists). As an example, in the case of values on string domain, edit distance functions can be used to support such a task [8]; such functions consider the minimum number of operations on individual characters needed in order to transform a string into another.

Another interesting definition of accuracy is provided in [7], on the basis of ontological considerations; in such a case, the difference is between what is observed in the system state vs. that in the “real world” state. As we are in a multi-system scenario, we prefer a definition not including the concept of system.

**Completeness** *Completeness is the degree to which values of a schema element are present in the schema element instance.*

According to such a definition, we can consider: (i) the completeness of an attribute value, as dependent from the fact that the attribute is present or not; (ii) the completeness of a schema element instance, as dependent from the number of the attribute values that are present.

A recent work [9] distinguishes other kinds of completeness, namely: schema completeness, column completeness and population completeness. The measures of such completeness types can give very useful information for a “general” assessment of the data completeness of an organization, but, as it will be clarified in Section 3.3, our focus in this paper is on associating a quality evaluation on “each” data a cooperating organization makes available to the others.

In evaluating completeness, it is important to consider the meaning of null values of an attribute, depending on the attribute being mandatory, optional, or inapplicable: a null value for a mandatory attribute is associated with a lower completeness, whereas completeness is not affected by optional or inapplicable null values.

As an example, consider the attribute **E-mail** of the **Citizen** schema element; a null value for **E-mail** may have different meanings, that is (i) the specific citizen has no e-mail address, and therefore the attribute is inapplicable (this case has no impact on completeness), or (ii) the specific citizen has an e-mail address which has not been stored (in this case completeness is low).

**Currency** The currency dimension refers only to data values that may vary in time; as an example, values of **Address** may vary in time, whereas **DateOfBirth** can be considered invariant. Currency can be defined as:

*Currency is the distance between the instant when a value changes in the real world and the instant when the value itself is modified in the information system.*

More complex definitions of currency have been proposed in the literature [10]; they are especially suited for specific types of data, i.e., they take into account elements such as “volatility” of data (e.g., data such as stock price quotes that change very frequently). However, in this work, we stay with the provided general definition of currency, leaving its extensions to future work.

Currency can be measured either by associating to each value an “updating time-stamp” [11] or a “transaction time” in temporal databases [12].

**Internal Consistency** Consistency implies that two or more values do not conflict each other. Internal consistency means that all values being compared in order to evaluate consistency are within a specific instance of a schema element.

A semantic rule is a constraint that must hold among values of attributes of a schema element, depending on the application domain modeled by the schema element. Then, internal consistency can be defined as:

*Internal Consistency is the degree to which the values of the attributes of an instance of a schema element satisfy the specific set of semantic rules defined on the schema element.*

As an example, if we consider `Citizen` with attributes `Name`, `DateOfBirth`, `Sex` and `DateOfDeath`, some possible semantic rules to be checked are:

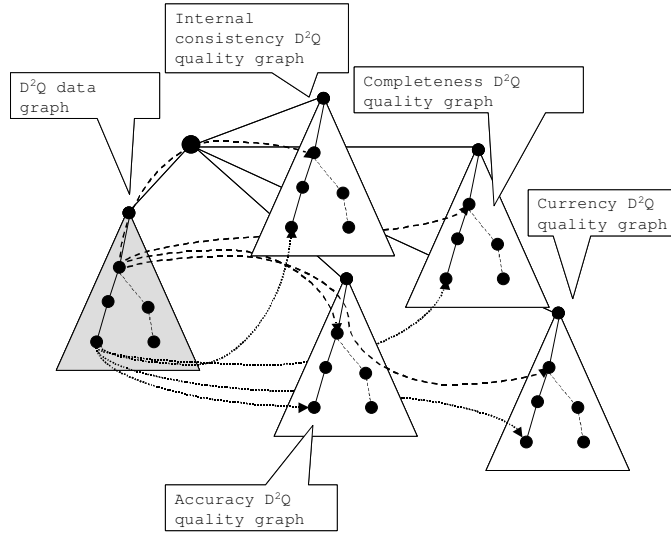
- the values of `Name` and `Sex` for an instance `p` are consistent. If `p.Name` has a value  $v = \text{JOHN}$  and the value of `p.Sex` is `FEMALE`, this is a case of low internal consistency;
- the value of `p.DateOfBirth` must precede the value of `p.DateOfDeath`.

Internal consistency has been widely investigated both in the database area through integrity constraints (e.g., [13]) and in the statistics area, through edits checking (e.g., [14]).

### 3.2 Data Model

The  $D^2Q$  model is inspired by the data model underlying XML-QL [15]. A database view of XML is adopted: an XML Document is a set of data items, and a Document Type Definition (DTD) is the schema of such data items, consisting of *data* and *quality classes*. In particular, a  $D^2Q$  XML document contains both application data, in the form of a  $D^2Q$  *data graph*, and the related data quality values, in the form of four  $D^2Q$  *quality graphs*, one for each quality dimension introduced in Section 3.1. Specifically, nodes of the  $D^2Q$  data graph are linked to the corresponding ones of the  $D^2Q$  quality graphs through links, as shown in Figure 2. Organizations in the CIS exchange each other data and quality data as  $D^2Q$  XML documents.

A  $D^2Q$  XML document corresponds to a set of conceptual data items, which are instances of conceptual schema elements; schema elements are data and quality classes, and instances are data and quality objects. Data classes and objects are straightforwardly represented as  $D^2Q$  data graphs, as detailed in the



**Fig. 2.** The generic structure of a  $D^2Q$  XML document

following of this section, and quality classes and objects are represented as  $D^2Q$  quality graphs, as detailed in Section 3.3.

As a running example, consider the document `citizens.xml`, shown in Figure 3, which contains entries about citizens with the associated quality data. Such a document corresponds to a set of conceptual data items, which are instances of conceptual schema elements; schema elements are data and quality classes, and instances are data and quality objects. Specifically, an instance of `Citizen` and the related `Accuracy` values are depicted.

A data class  $\delta (\pi_1, \dots, \pi_n)$  consists of:

- a name  $\delta$ ;
- a set of properties  $\pi_i = \langle name_i : type_i \rangle, i = 1 \dots n, n \geq 1$ , where  $name_i$  is the name of the property  $\pi_i$  and  $type_i$  can be:
  - either a basic type<sup>2</sup>;
  - or a data class;
  - or a type **set-of**  $\langle X \rangle$ , where  $\langle X \rangle$  can be either a basic type or a data class.

We define a  $D^2Q$  data graph as follows:

A  $D^2Q$  data graph  $\mathbb{G}$  is a graph with the following features:

- a set of nodes  $\mathcal{N}$ ; each node (i) is identified by an object identifier and (ii) is the source of 4 different links to quality objects, each one for a different quality

<sup>2</sup> Basic types are the ones provided by the most common programming languages and SQL, that is `Integer`, `Real`, `Boolean`, `String`, `Date`, `Time`, `Interval`, `Currency`, `Any`.

```

<Citizens>
  <Citizen Accuracy="o00" ... >
    <Name Accuracy="o01" ... >Maria</Name>
    <Surname Accuracy="o02" ... >Rossi</Surname>
    <ResidenceAddress Accuracy="o03" ... >
      <Street Accuracy="o04" ... >
        Via Salaria 113
      </Street>
      <City Accuracy="o05" ... >Roma</City>
      <ZIPCode Accuracy="o06" ... >
        00198
      </ZIPCode>
      <Country Accuracy="o07" ... >
        Italy
      </Country>
    </ResidenceAddress>
    <TelephoneNumber Accuracy="o08" ... >
      +390649918479
    </TelephoneNumber>
    <TelephoneNumber Accuracy="o09" ... >
      +393391234567
    </TelephoneNumber>
  </Citizen>
  <Accuracy_Citizen OID="o00">
    <Accuracy_Name OID="o01">
      0.7
    </Accuracy_Name>
    <Accuracy_Surname OID="o02">
      0.7
    </Accuracy_Surname>
    <Accuracy_ResidenceAddress OID="o03">
      <Accuracy_Street OID="o04">
        0.3
      </Accuracy_Street>
      <Accuracy_CityName OID="o05">
        0.9
      </Accuracy_CityName>
      <Accuracy_ZIPCode OID="o06">
        0.9
      </Accuracy_ZIPCode>
      <Accuracy_Country OID="o07">
        0.9
      </Accuracy_Country>
    </Accuracy_ResidenceAddress>
    <Accuracy_TelephoneNumber OID="o08">
      0.5
    </Accuracy_TelephoneNumber>
    <Accuracy_TelephoneNumber OID="o09">
      0.2
    </Accuracy_TelephoneNumber>
  </Accuracy_Citizen>
  ... ..
</Citizens>

```

**Fig. 3.** The XML document of the running example

- dimension. A link is a pair attribute-value, in which attribute represents the specific quality dimension for the element tag and value is an IDREF link<sup>3</sup>;*
- a set of edges  $\mathcal{E} \subset \mathcal{N} \times \mathcal{N}$ ; each edge is labeled by a string, which represents an element tag of an XML document;
  - a single root node  $\mathcal{R}$ ;
  - a set of leaves; leaves are nodes that (i) are not identified and (ii) are labeled by strings, which represent element tag values, i.e., the values of the element tags labeling edges to them.

Data class instances can be represented as  $D^2Q$  data graphs, according to the following rules.

Let  $\delta (\pi_1, \dots, \pi_n)$  be a data class with  $n$  properties, and let  $\mathcal{O}$  be a data object, i.e., an instance of the data class. Such an instance is represented by a  $D^2Q$  data graph  $\mathbb{G}$  as follows:

- The root  $\mathcal{R}$  of  $\mathbb{G}$  is labeled with the object identifier of the instance  $\mathcal{O}$ .
- For each  $\pi_i = \langle \text{name}_i : \text{type}_i \rangle$  the following rules hold:
  - if  $\text{type}_i$  is a basic type, then  $\mathcal{R}$  is connected to a leaf  $lv_i$  by the edge  $\langle \mathcal{R}, lv_i \rangle$ ; the edge is labeled with  $\text{name}_i$  and the leaf  $lv_i$  is labeled with the property value  $\mathcal{O}.\text{name}_i$ ;
  - if  $\text{type}_i$  is a data class, then  $\mathcal{R}$  is connected to the  $D^2Q$  data graph which represents the property value  $\mathcal{O}' = \mathcal{O}.\text{name}_i$  by an edge labeled with  $\text{name}_i$ ;
  - if  $\text{type}_i$  is a **set-of**  $\langle \mathbf{X} \rangle$ , then:
    - \* let  $C$  be the cardinality of  $\mathcal{O}.\text{name}_i$ ;  $\mathcal{R}$  is connected to  $C$  elements as it follows: if (i)  $\langle \mathbf{X} \rangle$  is a basic type, then the elements are leaves (each of them labeled with a property value of the set); otherwise if (ii)  $\langle \mathbf{X} \rangle$  is a data class, then the elements are  $D^2Q$  data graphs, each of them representing a data object of the set;
    - \* edges connecting the root to the elements are all labeled with  $\text{name}_i$ .

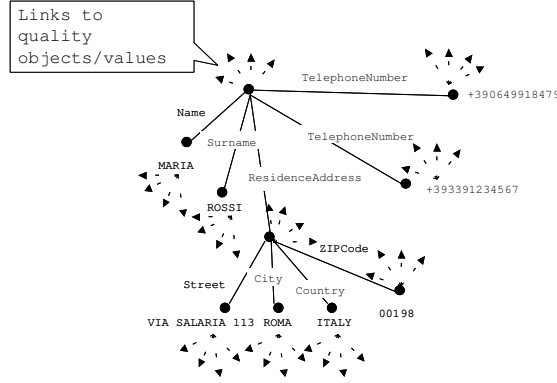
In Figure 4, the  $D^2Q$  data graph of the running example is shown: an object instance **Maria Rossi** of the data class **Citizen** is considered. The data class has **Name** and **Surname** as properties of basic types, a property of type **set-of**  $\langle \text{TelephoneNumber} \rangle$  and another property of data class type **ResidenceAddress**; the data class **ResidenceAddress** has all properties of basic types.

### 3.3 Quality Model

So far the data portion of the  $D^2Q$  model has been described. However, organizations export XML documents containing not only data objects, but also quality data concerning the four dimensions introduced in Section 3.1.

The assumption that each organization is able and is willing to export quality values is motivated by the cooperative scenarios that we experienced; as an example, in a recent Italian *e-Government* project concerning toponymic data,

<sup>3</sup> The use of links will be further explained in Section 3.3, when quality graphs are introduced.



**Fig. 4.** The  $D^2Q$  data graph of the running example

each organization participating in the project has to export quality values (calculated on the basis of statistical considerations). In our architecture, a specific element, i.e., the Quality Factory, is in charge of providing such values; the design of such an element is out of the scope of this paper, and has been addressed in [16].

Quality data are represented as graphs, too; they correspond to a set of conceptual quality data items, which are instances of conceptual quality schema elements; quality schema elements are referred to as *quality classes* and instances as *quality objects*. A quality class models a specific quality dimension for a specific data class: the property values of a quality object represent the quality dimension values of the property values of a data object. Therefore, each data object (i.e., node) and value (i.e., leaf) of a  $D^2Q$  data graph is linked to respectively four quality objects and values.

Let  $\delta (\pi_1, \dots, \pi_n)$  be a data class. A quality class  $\delta^D (\pi_1^D, \dots, \pi_n^D)$  consists of:

- a name  $\delta^D$ , with  $D \in \{ \text{Accuracy, Completeness, Currency, InternalConsistency} \}$ ;
- a set of tuples  $\pi_i^D = \langle \text{name}_i^D : \text{type}_i^D \rangle, i = 1 \dots n, n \geq 1$ ,

where:

- $\delta^D$  is associated to  $\delta$  by a one-to-one relationship and corresponds to the quality dimension  $D$  evaluated for  $\delta$ ;
- $\pi_i^D$  is associated to  $\pi_i$  of  $\delta$  by a one-to-one relationship, and corresponds to the quality dimension  $D$  evaluated for  $\pi_i$ ;
- $\text{type}_i^D$  is either a basic type or a quality class or a **set-of** type, according to the structure of the data class  $\delta$ .

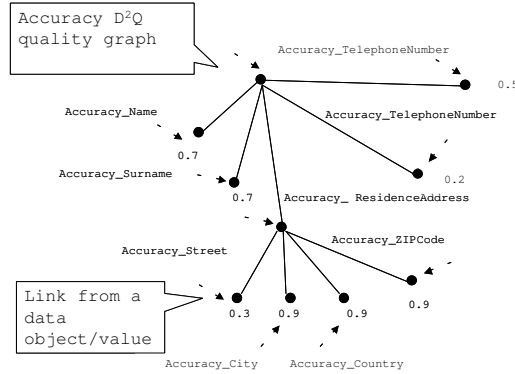
In order to represent quality objects, we define a  $D^2Q$  quality graph as follows:

A  $D^2Q$  quality graph  $G^D$  is a  $D^2Q$  data graph with the following additional features:

- no node nor leaf is linked to any other element;

- labels of edges are strings of the form  $D\_name$  (e.g., `Accuracy_Citizen`);
- labels of leaves are strings representing quality values;
- leaves are identified by object identifiers.

A quality class instance can be straightforwardly represented as a  $D^2Q$  quality graph, on the basis on rules analogous to the ones previously presented for data objects and  $D^2Q$  data graphs. As an example, in Figure 5, the  $D^2Q$  quality graph concerning accuracy of the running example is shown, and links are highlighted; for instance, the accuracy of *Maria* is 0.7.



**Fig. 5.** The accuracy  $D^2Q$  quality graph of the running example

Data and quality data are exchanged as  $D^2Q$  XML documents in the CIS. Specifically, for each data class instance there is a  $D^2Q$  data graph linked to four  $D^2Q$  quality graphs, expressing the quality of the data objects for each dimension introduced in Section 3.1.

Let  $\{ \mathcal{O}_1, \dots, \mathcal{O}_m \}$  be a set of  $m$  objects which are instances of the same data class  $\delta$ ; a  $D^2Q$  XML document is a graph consisting of:

- a root node  $ROOT$ ;
- $m$   $D^2Q$  data graph  $\mathcal{G}_i$ ,  $i = 1 \dots m$ , each of them representing the data objects  $\mathcal{O}_i$ ;
- $4 * m$   $D^2Q$  quality graph  $\mathcal{G}_i^D$ ,  $i = 1 \dots m$ , each of them representing the quality graph related to  $\mathcal{O}_i$  concerning the quality dimension  $D$ ;
- $ROOT$  is connected to the  $m$   $D^2Q$  data graphs by edges labeled with the name of the data class, i.e.,  $\delta$ ;
- for each quality dimension  $D$ ,  $ROOT$  is connected to the  $m$   $D^2Q$  quality graph  $\mathcal{G}_i^D$  by edges labeled with the name of the quality class, i.e.,  $\delta^D$ .

The model proposed in this work adopts several graphs instead of embedding metadata within the data graph. Such a decision increases the document size, but on the other hand allows a modular and “fit-for-all” design: (i) extending the model to new dimensions is straightforward, as it requires to define the new dimension quality graph, and (ii) specific applications, requiring only some dimension values, will adopt only the appropriate subset of the graphs.

## 4 The Data Quality Broker

The *Data Quality Broker (DQB)* is the core component of the architecture. It is implemented as a peer-to-peer distributed service: each organization hosts a copy of the Data Quality Broker that interacts with other copies. We first provide an overview of the high-level behavior of the Data Quality Broker in Sections 4.1 and 4.2; then, in Sections 4.3 and 4.4, we explain the details of its design and implementation.

### 4.1 Overview

The general task of the Data Quality Broker is to allow users to select data in the CIS according to their quality. Specifically, a user inside an organization can issue a query on a  $D^2Q$  *global schema*, specifying constraints on quality values. Then, the Data Quality Broker selects, among all the copies of the requested data that are in the CIS, only those satisfying all specified constraints.

Quality constraints can be related to quality of *data* and/or quality of *service* (QoS). Specifically, besides the four data quality dimensions previously introduced, two QoS dimensions are defined, namely: *(i) responsiveness*, defined by the average round-trip delay evaluated by each organization periodically pinging gateway in the CIS; and *(ii) availability* of a source, evaluated by pinging a gateway and considering if a timeout expires; in such a case the source is considered not available for being currently queried.

A user can request data with a specified quality (e.g., “return citizens with accuracy greater than 0.7”) or declare a limit on the query processing time, restricting the query only to available sources (e.g., “return only citizens from the first responding source”) or include both kinds of constraints (e.g., “return citizens with accuracy greater than 0.7 provided by the first responding source”).

The Data Quality Broker performs two specific tasks:

- *query processing* and
- *quality improvement*.

The semantics of query processing is described in Section 4.2, and its realization is detailed in Section 4.3. The quality improvement feature consists of notifying organizations with low quality data about higher quality data that are available in the CIS. This step is enacted each time copies of the same data with different quality are collected during the query processing activity. The best quality copy is sent to all organizations having lower quality copies. Organizations involved in the query have the choice of updating or not their data. This gives a non-invasive feedback that allows to enhance the overall quality of data in the CIS, while preserving organizations’ autonomy. The details of how the improvement feature is realized are provided in Section 4.3.

## 4.2 Query Processing

The Data Quality Broker performs query processing according to a *global-as-view* (GAV) approach, by unfolding queries posed over a global schema, i.e., replacing each atom of the original query with the corresponding view on local data sources [17, 2]. Both the global schema and local schemas exported by cooperating organizations are expressed according to the  $D^2Q$  model. The adopted query language is XQuery [18]. The unfolding of an XQuery query issued on the global schema can be performed on the basis of well-defined mappings with local sources. Details concerning such a mapping are beyond the scope of this paper (details can be found in [19]).

Under our hypothesis, data sources have different copies of the same data at different quality levels, i.e., there are *instance-level conflicts*. We resolve these conflicts at query execution time by relying on quality values associated to data: when a set of different copies of the same data are returned, we look at the associated quality values, and select the copy to return as a result on the basis of such values. In [20] a technique with a query processing semantics similar to the one we propose can be found; the main difference of that work with respect to our approach is that we have quality metadata associated to data that allows us to resolve conflicts; instead, in [20] conflict resolution functions are explicitly provided.

Specifically, the detailed steps we follow to answer a query with quality constraints are:

1. let  $\mathcal{Q}$  be a query posed on the the global schema  $\mathbb{G}$ ;
2. The query  $\mathcal{Q}$  is unfolded according to the static mapping that defines each concept of the global schema in terms of the local sources; such a mapping is defined in order to retrieve all copies of same data that are available in the CIS. Therefore, the query  $\mathcal{Q}$  is decomposed in  $\mathcal{Q}_1, \dots, \mathcal{Q}_n$  queries to be posed over local sources;
3. The execution of the queries  $\mathcal{Q}_1, \dots, \mathcal{Q}_n$  returns a set of results  $\mathcal{R}_1, \dots, \mathcal{R}_n$ . On such a set the following property is checked: given  $\mathcal{R}_i$  and  $\mathcal{R}_j$ ,  $\mathcal{R}_i$  is *extensionally correspondent* to  $\mathcal{R}_j$ , iff for each tuple  $\mathfrak{t}$  in  $\mathcal{R}_i$ , a tuple  $\mathfrak{t}'$  exists in  $\mathcal{R}_j$ , such that  $\mathfrak{t}$  and  $\mathfrak{t}'$  are related to a same object but have inconsistent values. Similarly, for each tuple  $\mathfrak{t}'$  in  $\mathcal{R}_j$ , a tuple  $\mathfrak{t}$  in  $\mathcal{R}_i$  should exist that refers to the same object. This step is performed by using a well-known algorithm for record matching [21].
4. On the basis of quality constraints specified in the query, a (set of) admissible results is constructed, selecting all  $\mathcal{R}_i$  satisfying the quality constraints.

The global schema and the local schemas exported by organizations have similar structural properties, in order to compare the values of the quality descriptors of the respective data items. Wrappers in each organization (see Section 4.3) are in charge of the translation and transformation between local schemas and internal database schemas.

If data graphs are re-structured (for data integration purposes, e.g., when a new organization adheres to the CIS), the data quality graphs must be re-structured as well. In general, a GAV approach imposes some constraints on the

opportunity of frequently restructuring the global schema. In the cooperative scenarios that we experienced [1], such constraints are not critical, as the CIS is built for including the most of organizations (i.e., public administrations) since the beginning, and the entrance of a new organization in the cooperative system is very rare (indeed, the creation of the CIS and the participating administrations are regulated by laws, Government directives, etc.).

### 4.3 Internal Architecture and Deployment

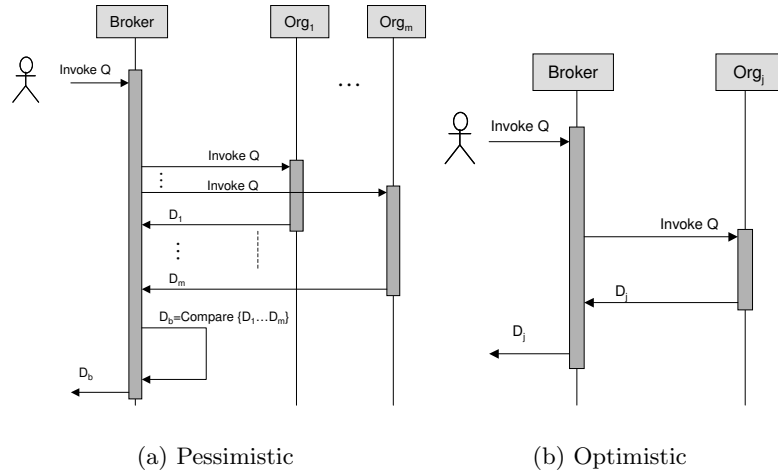
In this section we detail the design of the Data Quality Broker, by considering its interaction modes and its constituting modules; then, an example of query processing execution is shown.

The Data Quality Broker can operate either in *pessimistic* or in *optimistic* mode. In the former, the broker accesses all sources that provide data satisfying the query and builds the result, whereas in the latter the broker accesses only those organizations that *probably* will provide good quality data. Such information can be derived on the basis of statistics on quality values. In the optimistic mode, the broker does not contact all sources and does not have to wait for all responses; however, this optimization can lead to non accurate results. Conversely, in the pessimistic mode, all sources are always contacted; it takes more time to retrieve the final result, but it is always the most accurate one.

Figure 6 shows the sequence of operations performed in each of the two modes: in pessimistic mode (Figure 6(a)), all organizations storing data satisfying a query are queried and all results have to be waited for before proceeding. After results are retrieved, a comparison phase is required to choose the best data. In optimistic mode (Figure 6(b)) only the organization  $Org_j$  that has the best overall statistical quality evaluation is involved in the query and no comparison is required.

Clients of the broker can invoke one of the following operations: (i) `invoke(query, mode)`, for submitting a query  $Q$  over the global schema including quality constraints, and (ii) `propose(data)`, for notifying all sources, that have previously sent data with lower quality than the selected one, with higher quality results. Indeed each time a query is posed, different copies of same data are received as answers; the broker submits to organizations the best quality copy  $\mathcal{R}$  selected among the received ones.

Internally the Data Quality Broker is composed of five interacting modules (Figure 7). The Data Quality Broker is deployed as a peer-to-peer service: each organization hosts an independent copy of the broker and the overall service is realized through an interaction among the various copies, performed via the Transport Engine module. The modules Query Engine, Transport Engine and Schema Mapper are general and can be installed without modifications in each organization. The module Wrapper has to be customized for the specific data storage system. The module Comparator can also be customized to implement different criteria for quality comparison.



**Fig. 6.** Broker invocation modes

*Query Engine (QE)*: it receives the query and splits the query in sub-queries (local to the sources) by interacting with the Schema Mapper (which manages the mapping between the global schema and the local schemas). Then, the *QE* also interacts with the *TE* in order to: (i) send local queries to other copies of the *QE* and receive the answers; (ii) be notified about QoS parameters to be used for optimization.

*Wrapper (Wr)*: translates the query from the language used by the broker to the one of the specific data source. In this work the wrapper is a read-only access module, that is it returns data stored inside organizations without updating them.

*Transport Engine (TE)*: it provides general connectivity among all Data Quality Broker instances in the CIS. Copies of the *TE* interact with each other in two different scenarios, corresponding to two different interaction types:

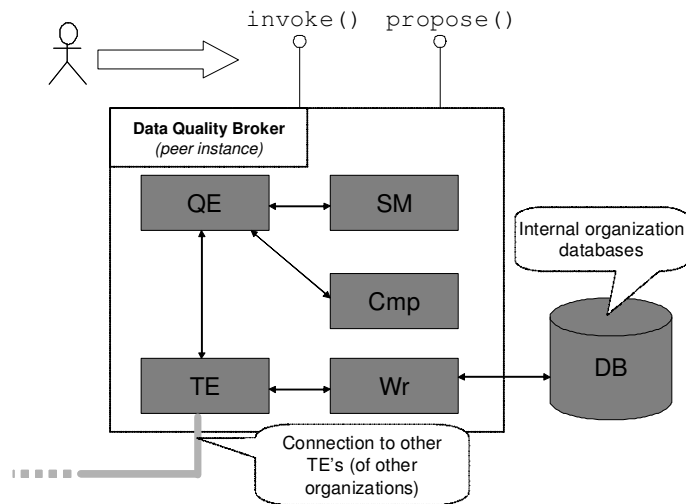
- Query execution (request/reply interaction): the requesting *TE* sends a query to the local *TE* of the target data source and remains blocked waiting for the result. A request is always synchronous but it can be always stopped from the outside, for example when the *QE* decides to block the execution of a query to enforce QoS constraints. In this case, the results are discarded.
- Quality feedback (asynchronous interaction): when a requesting *QE* has selected the best quality result of a query, it contacts the local *TE*'s to enact quality feedback propagation. The `propose()` operation is executed as a callback on each organization, with the best quality selected data as a parameter. The `propose()` can be differently implemented by each organization: a remote *TE* simply invokes this operation, whose behaviour depends from the

strategy that an organization chooses for the improvement of its own data on the basis of external notifications.

The other function performed by the *TE* is the evaluation of the QoS parameters. This feature is encapsulated into the *TE* as it can be easily implemented exploiting *TE*'s communication capabilities. Specifically, the *TE* periodically pings data sources in order to calculate responsiveness. Moreover, the timeout value necessary to calculate the availability QoS parameter is configured in the *TE*.

*Schema Mapper (SM)*: stores the static mappings between the global schema and the local schemas of the various data sources.

*Comparator (CMP)*: compares the quality values returned by the different sources to choose the best one. *CMP* compares quality value vectors according to ranking methods known in literature, such as Simple Additive Weighting (SAW) [22] or Analytical Hierarchy Process (AHP) [23]



**Fig. 7.** Broker modules

As an example, we give details of pessimistic mode query invocations, by describing the interaction among peer Data Quality Broker instances and among modules inside each Data Quality Broker when a query with quality requirements is submitted by a client.

After receiving the query, the *QE* (local to the client) interacts with the *SM* and generates a set of sub-queries, which in turn are sent through the local *TE* to all *TE*'s of the involved organizations. All *TE*'s transfer the query to local

$QE$ 's that execute local queries by interacting with local  $Wr$ 's. Query results are passed back to the local  $TE$  that passes all results to the local  $QE$ . This module selects the best quality result through the  $CMP$  and passes it to the client. Once the best quality data is selected the feedback process starts.

This process involves all sources that have previously sent data with lower quality than the selected one; each source may adopt the proposed data (updating its local data store) or not. Indeed criteria to decide whether updating the local data store with the feedback data may vary from one organization to another; such criteria can be set by providing proper implementations of the `propose()`.

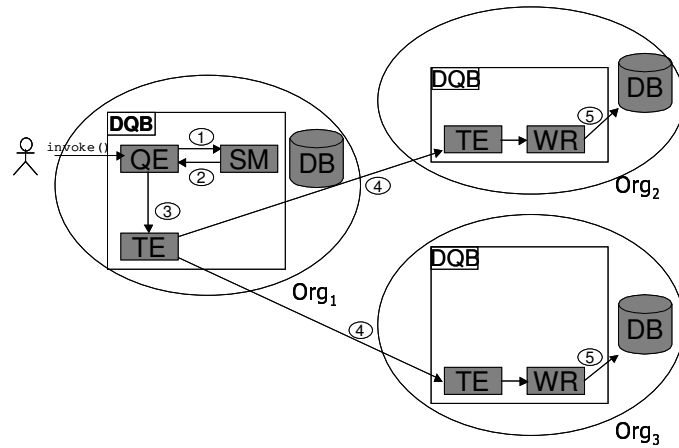
Figure 8 depicts an invocation by a client of the organization  $Org_1$ , specifically the `invoke('Select Mecella from Citizen with accuracy  $\geq 0.7$ ')`, PESSIMISTIC). For the sake of clarity, only involved modules are depicted.

The sequence of actions performed during the query invocation is described in the following. We indicate with a subscript the specific instance of a module belonging to a particular organization: for example,  $QE_i$  indicates the Query Engine of organization  $Org_i$ .

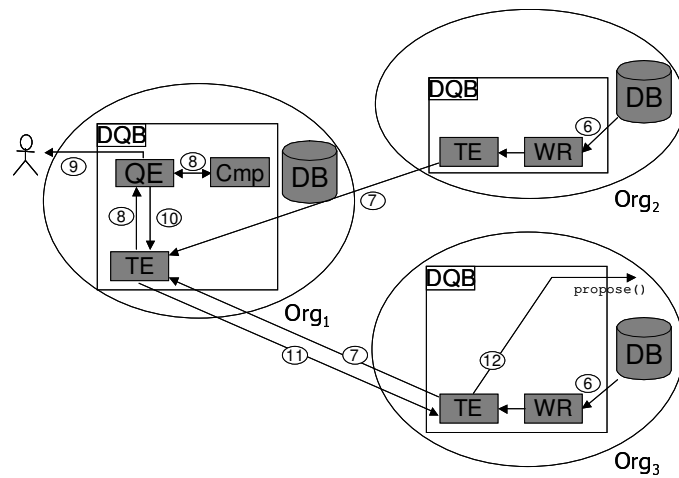
1.  $QE_1$  receives the query  $Q$  posed on the global schema, as a parameter of the `invoke()` function;
2.  $QE_1$  retrieves the mapping with local sources from  $SM_1$ ;
3.  $QE_1$  performs the unfolding that returns queries for  $Org_2$  and  $Org_3$  that are passed to  $TE_1$ ;
4.  $TE_1$  sends the request to  $Org_2$  and  $Org_3$ ;
5. The recipient  $TE$ 's pass the request to the  $Wr$ 's modules. Each  $Wr$  accesses the local data store to retrieve data;
6.  $Wr_3$  retrieves a query result, e.g.,  $\mathcal{R}_a$  with accuracy 0.7, and  $Wr_2$  retrieves another query result  $\mathcal{R}_b$  with accuracy 0.9;
7. Each  $TE$  sends the result to  $TE_1$ ;
8.  $TE_1$  sends the collected results ( $\mathcal{R}_a, \mathcal{R}_b$ ) to  $QE_1$ .  $QE_1$  selects through  $CMP_1$  the result with the greatest accuracy, i.e.,  $Org_2$ 's result ( $\mathcal{R}_b$  with accuracy 0.9);
9.  $QE_1$  sends the selected result to the client;
10.  $QE_1$  starts the feedback process sending the selected result ( $\mathcal{R}_b$  with accuracy 0.9) to  $TE_1$ ;
11.  $TE_1$  sends it to  $Org_3$ ;
12.  $TE_3$  receives the feedback data and makes a call `propose( $\mathcal{R}_b$  with accuracy 0.9)`.

#### 4.4 Implementation

We have developed a prototype of the broker [19, 24], implementing the pessimistic query invocation mode. In our prototype, the Data Quality Broker is realized as a Web Service [25], deployed on all the organizations. A Web Service-based solution is suitable for integrating heterogeneous organizations as it allows to deploy the Data Quality Broker on the Internet, regardless from the access



(a) Step 1 – 5



(b) Step 6 – 12

**Fig. 8.** Details of query invocation in the pessimistic mode

limitations imposed by firewalls, interoperability issues, etc. Moreover, by exploiting standard Web Service technologies, the technological integration among different organizations is straightforward; specifically, we tested interoperability among different versions of the Data Quality Broker, realized using respectively the JAX-RPC<sup>4</sup> framework and the .NET one<sup>5</sup>.

As we discussed above, the component responsible for the communication between the Data Quality Broker instances is the Transport Engine. Besides the operations previously described (i.e., `propose()` and `invoke()`), it exposes also the `ping()` operation, which is used to estimate the query responsiveness. Such operations are invoked through standard SOAP messages sent over HTTP; specifically, invocations to multiple Transport Engines are performed in parallel. Remote invocations to the `invoke()` operation require the client Transport Engine to wait for all results.

If we assume that Data Quality Broker instances are subject to faults this can cause the client to wait indefinitely for responses from faulty instances. From the distributed system theory stems that in communication systems, such as the Internet, where delays on message transfer cannot be predicted, it is impossible to distinguish a faulty software component from a very slow one [26, 27]. Then, the client Transport Engine sets a timeout for each invoked source. The timeout value is estimated for each single source on the basis of its responsiveness. After the timeout expires, no result is returned for that query. This is a best-effort semantics that does not guarantee to always return the best quality value but rather enforces the correct termination of each query invocation. Replication techniques can be exploited in order to enhance the availability of each web service, ensuring more accurate results even in presence of faults [28].

## 5 Related Work

Data Quality has been traditionally investigated in the context of single information systems. Only recently, there is a growing attention towards data quality issues in the context of multiple and heterogeneous information systems [29–31].

In cooperative scenarios, the main data quality issues regard [30]: *(i)* assessment of the quality of the data owned by each organization; *(ii)* methods and techniques for exchanging quality information; *(iii)* improvement of quality within each cooperating organization; and *(iv)* heterogeneity, due to the presence of different organizations, in general with different data semantics.

For the assessment *(i)* issue, some of the results already achieved for traditional systems can be borrowed, e.g., [32, 33].

Methods and techniques for exchanging quality information *(ii)* have been only partially addressed in the literature. In [29], an algorithm to perform query planning based on the evaluation of data sources' quality, specific queries' quality and query results' quality is described. The approach of [29] is different from our approach mainly because we assume to have quality metadata associated

<sup>4</sup> <http://java.sun.com/xml/jaxrpc/>

<sup>5</sup> <http://msdn.microsoft.com/netframework/>

to each quality value: this gives us the possibility of more accurate answers in selecting data on the basis of quality. Specifically, the granularity of quality data in our framework is finer than the one proposed in [29], as our framework is also targeted to improvement actions.

When considering the issue of exchanging data and the associated quality, a model to export both data and quality data needs to be defined. Some conceptual models to associate quality information to data have been proposed that include an extension of the Entity-Relationship model [34], and a data warehouse conceptual model with quality features described through the Description Logic formalism [33]. Both models are thought for a specific purpose: the former to introduce quality elements in relational database design; the latter to introduce quality elements in the data warehouse design. Whereas, in the present paper the aim is to enable quality exchanging in a generic CIS, independently of the specific data model and system architecture. Moreover, previous models are not well suited for semi-structured data.

Our work is similar to the extensions of the relational model described in [35], chapters 2 and 3; in particular, the attribute-based model can be considered as a possible mechanism for effectively storing data and quality data inside organizations (by assuming that back-end systems are on relational database), to be then converted and exported as  $D^2Q$  documents.

In [36], the problem of the quality of web-available information has been faced in order to select data with high quality coming from distinct sources: every source has to evaluate some pre-defined data quality parameters, and to make their values available through the exposition of metadata. Our proposal is different as we propose an ad-hoc service that brokers data requests and replies on the basis of data quality information. Moreover, we also take into account improvement features (*iii*) that are not considered in [36].

The heterogeneity (*iv*) issues are taken into account by data integration works. Data integration is the problem of combining data residing at different sources, and providing the user with a unified view of these data [2]. As described in [20], when performing data integration two different types of conflicts may arise: *semantic conflicts*, due to heterogeneous source models, and *instance-level conflicts*, due to what happens when sources record inconsistent values on the same objects. The Data Quality Broker described in this paper is a system solving instance-level conflicts.

Other notable examples of data integration systems within the same category are AURORA [20] and the system described in [37]. AURORA supports conflict tolerant queries, i.e. it provides a dynamic mechanism to resolve conflicts by means of defined conflict resolution functions. The system described in [37] describes how to solve both semantic and instance-level conflicts. The proposed solution is based on a multidatabase query language, called FraQL, which is an extension of SQL with conflict resolution mechanisms. Similarly to both such systems, the Data Quality Broker supports dynamic conflict resolution, but differently from them the Data Quality Broker relies onto quality metadata for solving instance-level conflicts.

A system that also takes into account metadata (i.e., not necessarily and specifically regarding quality of data) for instance-level conflict resolution is described in [38]; such a system adopts the ideas of the COIN framework [39]: context dependent and independent conflicts are distinguished and accordingly to this very specific direction, conversion rules are discovered on pairs of systems. Moreover, differently from the COIN prototype, the Data Quality Broker is not a centralized system, but it is based on a distributed architecture.

## 6 Conclusions and Future Work

Managing data quality in CIS's merges issues from many research areas of computer science such as databases, software engineering, distributed computing, security, and information systems. This implies that the proposal of integrated solutions is very challenging. In this paper, an overall architecture to support data quality management in CIS's has been proposed. The specific contribution of the paper are (i) a model for data and quality data exported by cooperating organizations, and (ii) the design of a service for querying and improving quality data and.

Up to now, the pessimistic mode of the Data Quality Broker has been implemented. We plan to investigate in future work the research issues described throughout the paper concerning the optimistic invocation mode. We are currently investigating techniques for query optimization based on both data quality parameters and QoS parameters. The optimization based on data quality parameters can be performed on the basis of statistics on quality values. As an example, statistic values could be calculated on current data exported by organizations (e.g., “the medium accuracy of citizens currently provided by organization  $\mathcal{A}$  is high”). As another example, statistic values could be calculated on the basis of past performances of organizations in providing good quality data (e.g., “the accuracy of citizens provided by organization  $\mathcal{A}$  in the last  $N$  queries is high”). In such a way, the number of possible query answers is reduced by eliminating those that may not conform to quality requirements according to the statistical evaluation.

The complete development of a framework for data quality management in CIS's requires the solution of further issues. An important aspect concerns the techniques to be used for quality dimension measurement. In both statistical and machine learning areas, some techniques could be usefully integrated in the proposed architecture. The general idea is to give to each data value a quality evaluation with a certain estimated probability, instead of a deterministic quality evaluation. In this way, the task of assigning quality values to each data value could be considerably simplified.

## Acknowledgments.

The authors would like to thank all people involved in the DAQUINCIS project, in particular Sara Tucci Piergiovanni, the anonymous reviewers for their useful

suggestions, and Maurizio Lenzerini, Domenico Lembo and Carlo Marchetti for interesting discussions on some of the topics discussed in the paper.

This work is supported by MIUR, COFIN 2001 Project “DaQuinCIS - Methodologies and Tools for Data Quality in Cooperative Information Systems” (<http://www.dis.uniroma1.it/~dq/>).

## References

1. C. Batini and M. Mecella, “Enabling Italian e-Government Through a Cooperative Architecture,” *IEEE Computer*, vol. 34, no. 2, 2001.
2. M. Lenzerini, “Data Integration: A Theoretical Perspective,” in *Proceedings of the 21st ACM Symposium on Principles of Database Systems (PODS 2002)*, Madison, Wisconsin, USA, 2002.
3. C. Marchetti, M. Mecella, M. Scannapieco, A. Virgillito, and R. Baldoni, “Data Quality Notification in Cooperative Information Systems,” in *Proceedings of the First International Workshop on Data Quality in Cooperative Information Systems*, Siena, Italy, 2003.
4. L. De Santis, M. Scannapieco, and T. Catarci, “A Trust Model for Tightly Coupled P2P Systems,” in *Proceedings of the 11<sup>o</sup> Convegno Nazionale su Sistemi Evoluti per Basi di Dati (SEBD 2003)*, Cetraro (CS), Italy, 2003.
5. P. Bertolazzi, L. De Santis, and M. Scannapieco, “Automatic Record Matching in Cooperative Information Systems,” in *Proceedings of the ICDT’03 International Workshop on Data Quality in Cooperative Information Systems (DQCIS’03)*, Siena, Italy, 2003.
6. T.C. Redman, *Data Quality for the Information Age*, Artech House, 1996.
7. Y. Wand and R.Y. Wang, “Anchoring Data Quality Dimensions in Ontological Foundations,” *Communications of the ACM*, vol. 39, no. 11, 1996.
8. P.A.V. Hall and G.R. Dowling, “Approximate String Matching,” *ACM Computing Surveys*, vol. 12, no. 4, 1980.
9. L.L. Pipino, Y.W. Lee, and R.Y. Wang, “Data Quality Assessment,” *Communications of the ACM*, vol. 45, no. 4, 2002.
10. D.P. Ballou, R.Y. Wang, H. Pazer, and G.K. Tayi, “Modeling Information Manufacturing Systems to Determine Information Product Quality,” *Management Science*, vol. 44, no. 4, 1998.
11. P. Missier, M. Scannapieco, and C. Batini, “Cooperative Architectures: Introducing Data Quality,” Technical Report 14-2001, Dipartimento di Informatica e Sistemistica, Università di Roma “La Sapienza”, Roma, Italy, 2001.
12. A. Tansell, R. Snodgrass, J. Clifford, S. Gadia, and A. Segev (eds.), *Temporal Databases*, Benjamin-Cummings, 1993.
13. E.F. Codd, “Relational Database: a Practical Foundation for Productivity (1981 ACM Turing Award Lecture),” *Communications of the ACM*, vol. 25, no. 2, 1982.
14. R. Bruni and A. Sassano, “Errors Detection and Correction in Large Scale Data Collecting,” in *Proceedings of the 4th International Conference on Advances in Intelligent Data Analysis*, Cascais, Portugal, 2001.
15. A. Deutsch, M. Fernandez, D. Florescu, A. Levy, and D. Suciu, “XML-QL: A Query Language for XML,” in *Proceedings of the 8th International World Wide Web Conference (WWW8)*, Toronto, Canada, 1999.

16. C. Cappiello, C. Francalanci, B. Pernici, P. Plebani, and M. Scannapieco, "Data Quality Assurance in Cooperative Information Systems: a Multi-dimension Quality Certificate," in *Proceedings of the ICDT'03 International Workshop on Data Quality in Cooperative Information Systems (DQCIS'03)*, Siena, Italy, 2003.
17. J.D. Ullman, "Information Integration Using Logical Views," in *Proceedings of the Sixth International Conference on Database Theory (ICDT '97)*, Delphi, Greece, 1997.
18. S. Boag, D. Chamberlin, M.F. Fernandez, D. Florescu, J. Robie, and J. Simèon, "XQuery 1.0: An XML Query Language," W3C Working Draft, November 2002.
19. D. Milano, *Progetto DaQuinCIS: Estensione di XQuery e Query processing in un Sistema di Integrazione Basato sulla Qualità dei Dati*, Tesi di Laurea in Ingegneria Informatica, Università di Roma "La Sapienza", Facoltà di Ingegneria, 2003 (in Italian) (the thesis is available by writing an e-mail to: [monscan@dis.uniroma1.it](mailto:monscan@dis.uniroma1.it)).
20. L.L. Yan and M.T. Ozsü, "Conflict Tolerant Queries in AURORA," in *Proceedings of the Fourth International Conference on Cooperative Information Systems (CoopIS'99)*, Edinburgh, Scotland, UK, 1999.
21. M.A. Hernandez and S.J. Stolfo, "Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem," *Journal of Data Mining and Knowledge Discovery*, vol. 1, no. 2, 1998.
22. C. Hwang and K. Yoon, *Multiple Attribute Decision Making*, Lectures Notes in Economics and Mathematical Systems-Springer Verlag 186, 1981.
23. T.L. Saaty, *The Analytic Hierarchy Process*, McGraw-Hill, 1980.
24. G. Palmieri, *Progetto DaQuinCIS: Architettura Basata su Tecnologie Web Service ed Ottimizzazione di Query XML*, Tesi di Laurea in Ingegneria Informatica, Università di Roma "La Sapienza", Facoltà di Ingegneria, 2003 (in Italian) (the thesis is available by writing an e-mail to: [monscan@dis.uniroma1.it](mailto:monscan@dis.uniroma1.it)).
25. A. Buchmann, F. Casati, L. Fiege, M.C. Hsu, and M.C. Shan, Eds., *Proceedings of the 3rd VLDB International Workshop on Technologies for e-Services (VLDB-TES 2002)*, Hong Kong, China, 2002.
26. T.D. Chandra and S. Toueg, "Unreliable failure detectors for reliable distributed systems," *Journal of the ACM (JACM)*, vol. 43, no. 2, pp. 225–267, 1996.
27. M.J. Fischer, N.A. Lynch, and M.S. Paterson, "Impossibility of distributed consensus with one faulty process," *Journal of the ACM (JACM)*, vol. 32, no. 2, pp. 374–382, 1985.
28. R. Guerraoui and A. Schiper, "Software-Based Replication for Fault Tolerance," *IEEE Computer*, vol. 30, no. April 1997, 1997.
29. F. Naumann, U. Leser, and J.C. Freytag, "Quality-driven Integration of Heterogeneous Information Systems," in *Proceedings of 25th International Conference on Very Large Data Bases (VLDB'99)*, Edinburgh, Scotland, UK, 1999.
30. P. Bertolazzi and M. Scannapieco, "Introducing Data Quality in a Cooperative Context," in *Proceedings of the 6th International Conference on Information Quality (IQ'01)*, Boston, MA, USA, 2001.
31. L. Berti-Equille, "Quality-Extended Query Processing for Distributed Processing," in *Proceedings of the ICDT'03 International Workshop on Data Quality in Cooperative Information Systems (DQCIS'03)*, Siena, Italy, 2003.
32. H. Galhardas, D. Florescu, D. Shasha, and E. Simon, "An Extensible Framework for Data Cleaning," in *Proceedings of the 16th International Conference on Data Engineering (ICDE 2000)*, San Diego, CA, USA, 2000.
33. M. Jarke, M. Lenzerini, Y. Vassiliou, and P. Vassiliadis, Eds., *Fundamentals of Data Warehouses*, Springer Verlag, 1999.

34. R.Y. Wang, H.B. Kon, and S.E. Madnick, "Data Quality Requirements: Analysis and Modeling," in *Proceedings of the 9th International Conference on Data Engineering (ICDE '93)*, Vienna, Austria, 1993.
35. R.Y. Wang, M. Ziad, and Y.W. Lee, *Data Quality*, Kluwer Academic Publisher, 2001.
36. G. Mihaila, L. Raschid, and M. Vidal, "Querying Quality of Data Metadata," in *Proceedings of the 6th International Conference on Extending Database Technology (EDBT'98)*, Valencia, Spain, 1998.
37. K. Sattler, S. Conrad, and G. Saake, "Interactive Example-driven Integration and Reconciliation for Accessing Database Integration," *Information systems*, vol. 28, 2003.
38. K. Fan, H. Lu, S.E. Madnick, and D. Cheung, "Discovering and Reconciling Value Conflicts for Numerical Data Integration," *Information systems*, vol. 28, 2003.
39. "The COntext INterchange (COIN) Project," <http://context.mit.edu/~coin/>, 1996 - 1999.