Data Quality in *e*-Business Applications

Monica Scannapieco[1,2], Valeria Mirabella[1], Massimo Mecella[1], and Carlo Batini[3]

[1] Università di Roma "La Sapienza"
Dipartimento di Informatica e Sistemistica
Via Salaria 113, 00198 Roma, Italy
{monscan,mirabel,mecella}@dis.uniroma1.it
[2] Consiglio Nazionale delle Ricerche
Istituto di Analisi dei Sistemi ed Informatica (IASI-CNR)
Viale Manzoni 30, 00185 Roma, Italy
[3] Università di Milano "Bicocca"
Dipartimento di Informatica, Sistemistica e Comunicazione
Via Bicocca degli Arcimboldi 8, 20126 Milano, Italy
batini@disco.unimib.it

Abstract. In *e*-Business scenarios, an evaluation of the quality of exchanged data is essential for developing service-based applications and correctly performing cooperative activities. Data of low quality can spread all over the cooperative system, but at the same time, improvement can be based on comparing data, correcting them and disseminating high quality data. In this paper, an XML-based broker service for managing data quality in cooperative systems is presented, which selects the best available data from different services. Such a broker also supports data quality improvements based on feedbacks to source services.

Keywords: Data Quality, Broker, XML model, *e*-Business, *e*-Service

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 [13]. CIS's are the enabling paradigm in order to develop complex *e*-Business applications [19].

Among the different resources that are shared by organizations which conduct their businesses supported by a CIS, data are fundamental; in "real world" scenarios, organizations may not request data from others if they do not "trust" each others, i.e., if they do not know that the quality of the provided data is high. Therefore, lack of cooperation may occur due to lack of quality certification. Moreover, not certified quality can 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. As an example, in *e*-Governement scenarios, personal data about citizens are stored by the information systems of different administrations. 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 the literature, CIS's have been widely considered and various approaches are proposed for their design and development (e.g., [9,12,17]); in particular, service-based CIS's consider cooperation among different organizations to be obtained by sharing and integrating services across networks; such services, commonly referred to as *e*-Services and Web-Services [7], are exported by different organizations as well-defined functionalities that allow users and applications to access and perform tasks offered by back-end business applications.

In this paper, we propose a *Data Quality Broker* service. Such a service exploits the opportunity offered by data replication in cooperative environments with two aims: *(i)* brokering of data with highest quality levels, i.e., once a data request is issued by an organization, only data with the best quality are provided as responses, and *(ii)* improving the quality inside organizations by proposing the best available copy of data.

The structure of the paper is as follows. In Section 2, basic concepts of *e*-Service architecture and data quality are discussed, together with relevant research work. In Section 3, a framework for cooperative information systems specifically addressing quality related issues is proposed. In Section 4, the service for brokering high quality data among organizations is described, and an explanatory example is presented in Section 5. Finally, Section 6 concludes the paper by drawing future work.

2 Background and Related Work

2.1 *e*-Service Architectures

e-*Services* (also referred to as *Web Services*) are an evolutionary model in the utilization of the Web; until now the Web has provided the functionality for browsing of linked documents, manually-initiated transactions and manual downloading of files; conversely in the *e*-Service model *(i)* interactions are automatically initiated by other programs over the Web, not using a browser, *(ii)* services can be described, published, discovered and invoked dynamically over the Web, and *(iii)* communication is at the application-to-application level.

e-Services are self-contained, modular applications; they are the logical evolution from object oriented systems to systems of services, by following a component-based approach in which components (i.e., services) are large-grained and loosely coupled [23].

A framework for *e*-Services consists of *(i)* some basic operations (i.e., `describe`, `publish`, `unpublish` and `invoke`) and *(ii)* roles (i.e., service providers, service requesters and service repositories) [23], as shown in Figure 1:

Service Provider: it is the subject providing software applications for specific needs as services; *(i)* from a business perspective, this is the owner of the service (e.g., the subject which is possibly paid for its services), and *(ii)* from the architectural perspective, this is the platform the service is deployed onto. Available services are described by using a *service description language.*

Service Requestor: it is the party that uses the services; *(i)* from a business perspective, this is the business requiring certain services to be fulfilled (e.g., the payer subject), and *(ii)* from an architectural perspective, this is the application invoking the service.

Service Repository: it is the party providing a *repository* of service descriptions, where providers publish their services and requestors find services.

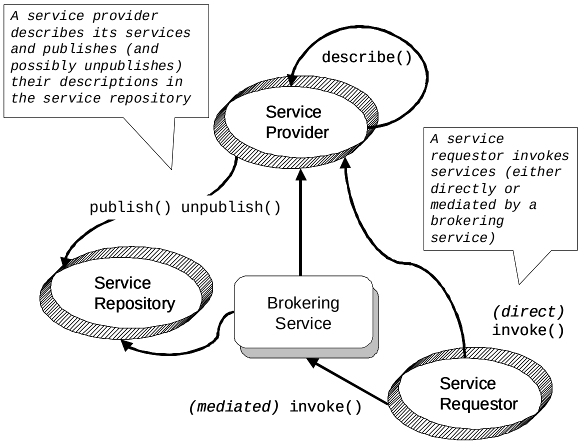
A service provider describes its services and publishes (and possibly unpublishes) their descriptions in the service repository
describe()
Service Provider
A service requestor invokes services (either directly or mediated by a brokering service)
publish() unpublish()
Service Repository
Brokering Service
(direct) invoke()
Service Requestor
(mediated) invoke()

Fig. 1. Basic elements of an *e*-Service framework

Composite *e*-Services are added-value services which compose basic ones in order to offer more advanced functionalities; they act as mediators between service requestors and service providers, possibly discovering into the repository which basic *e*-Services to use. In this paper, the design of a composite *e*-Service for data quality management is presented; it is shown in Figure 1 as “Brokering Service”.

Some frameworks for *e*-Services have been proposed; a notable example is the UDDI (Universal Description, Discovery & Integration)[1] framework, where Business Registries store different types of information about services, namely, business contact information (“white pages”), business category information (“yellow pages”) and technical service information (“green pages”); other proposals for architectures for *e*-Services have been presented in the literature [8,30].

Different languages, e.g., Web Service Description Language (WSDL) [1], can be used to describe services without impacting on the repository. In this paper, an XML-based model for describing data exchanged among services and related

[1] UDDI.org: `http://www.uddi.org`.

quality values is proposed, in order to be able to design a composite *e*-Service managing data quality. Such a model can be used inside WSDL descriptions of *e*-Services.

2.2 Data Quality

Data Quality has been traditionally investigated in the context of single information systems: methodologies to manage data quality in such systems have been proposed both by researchers [25,29] and by industrial practitioners [14,24]. Only recently, a methodological framework for data quality in cooperative systems has been proposed [5]; it will be described in Section 3.

In cooperative scenarios the main data quality issues regard: *(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)* and the heterogeneity *(iv)* issues, some of the results already achieved for traditional systems can be borrowed, specifically:

- ❐ the assessment phase can be based on the results achieved in the data cleaning area [11,15], as well as on the results in the data warehouse area [16,27];
- ❐ heterogeneity has been widely addressed in the literature, focusing on both schema and data integration issues [2,6,18,26].

Methods and techniques for exchanging quality information *(ii)* have been only partially addressed in the literature. In [21], 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 meta-data. 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 (i.e., *(iii)*) that are not considered in [21].

Data quality dimensions characterize properties that are inherent to data. The quality dimensions used in this work are those that are used most frequently in the literature [28], namely: *(i)* syntactic and semantic accuracy, *(ii)* completeness, *(iii)* currency, *(iv)* internal consistency and *(v)* source reliability. In the following we only recall the adopted dimensions; further details and examples can be found in [4,5,20]. Such dimensions concern only data values; instead, they do not deal with aspects concerning quality of logical schema and data format [24]. The need for providing such definitions stems from the lack of a common reference set of dimensions in the data quality literature.

In the following definitions, 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:

- ❒ **Syntactic and Semantic Accuracy.** Accuracy is commonly referred to as the proximity of a value v to a a value v' considered as correct; we further distinguish between syntactic accuracy, being v' the value considered syntactically correct (i.e., it belongs to the domain of values of v), and semantic accuracy, being v' the value considered semantically correct (i.e., it is consistent with respect to the real world).
- ❒ **Completeness.** It is the degree to which values of a schema element are present in the schema element instance.
- ❒ **Currency.** The distance between the instant when a value is last updated and the instant when the value itself is used.
- ❒ **Internal Consistency.** It 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.
- ❒ **Source Reliability**. It is defined as the credibility of a source organization with respect to provided data quality values.

Notice that source reliability has a nature different from other dimensions, as it depends from the cooperative context in which data are exchanged. Therefore source reliability will be separately considered in the following.

3 A Framework for Data Quality in CIS's

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 [3].

A cooperative business system exists independently of the presence of a software infrastructure supporting electronic data exchange and service provisioning. Indeed CIS's are the software systems supporting cooperative business systems; in the remain of this paper, we define a CIS as formed by a set of organizations which cooperate through a communication software infrastructure. Each organization is connected to the communication infrastructure through a gateway, on which *e*-Services offered by the organization to other ones are deployed [20].

Offered *e*-Services can perform different operations, such as initiating complex transactions on back-end systems, providing access to data, etc. In the present work we only consider *read-only access* services, that is services taking as input data queries in an appropriate query language and returning application data stored inside organizations without modifying them. Results returned by such *e*-Services are expressed as XML documents that convey not only application data items, but also data about the quality of such data items.

The TDQM_CIS methodological cycle [5] defines the methodological framework in which the proposed architecture fits. The TDQM_CIS cycle derives from the extension of the TDQM cycle [29] to the context of CIS's; it consists of the five phases of Definition, Measurement, Exchange, Analysis and Improvement (see Figure 2).

The *Definition* phase implies the definition of a model for the data exported by each cooperating organization and of the quality data associated to them. Both data and quality data can be exported as XML documents, consisting of elements defined on the basis of the ODMG Object Model [10], and formally defined in [4,20].

In general, types of exchanged data items can be either classes, when instances (i.e., data items) have their own identities, or literals, when instances have not identities. As an example, consider a `Citizen` class, identified by its `SSN` (Social Security Number) and having `Name`, `Surname` and `BirthDate` as attributes; the attribute `BirthDate` may be of a literal type `Date` with three attributes, namely `Day`, `Month`, `Year`.

As far as quality data, they can be associated to classes and literals through *quality classes* and *literals* [4]; they are the aggregation of the values of a specific data quality dimension for each of the attributes of either the data classes or the literals to which they refers. Therefore, to each data item (i.e., class/literal instance) a set of quality elements (i.e., quality class/literal instances) are associated, one for each dimension. As an example, it is possible to consider a quality class `SyntacticAccuracy_Citizen` associated to the class `Citizen`, the attributes of which are `SSN`, `Name`, `Surname` and `BirthDate`, representing the syntactic accuracy values of the attributes of the `Citizen` class.

The *Measurement* phase consists of the evaluation of the data quality dimensions for the exported data. With reference to previous examples, an assessment phase realized by the organization exporting the class `Citizen` should allow to measure the values of syntactic accuracy of the attributes `SSN`, `Name`, `Surname` and of the attributes `Day`, `Month` and `Year` of `BirthDate`. The assessment of the quality of the exported data by each cooperating organization can be made by using traditional methods (e.g., the statistical methods proposed in [22]).

The *Exchange* phase implies the exact definition of the exchanged information, consisting of data and of appropriate quality data with respect to data quality dimensions.

The *Analysis* phase regards the interpretation of the quality values contained in the exchanged information by the organization that receives it. An example of a possible interpretation is to weight an "high" accuracy value with a "low" source reliability value.

The *Improvement* phase consists of all the actions that allow improvements of cooperative data by exploiting the opportunities offered by the cooperative environment. An example of improvement action is based on the analysis phase described above: interpreting the quality of exchanged information gives the opportunity of sending accurate feedbacks to data source organizations, which can then implement correction actions to improve their quality.

As highlighted in Figure 2, the focus in this paper is on the two phases of exchange and improvement: as it will be described in the following section, a quality broker and an improvement manager will realize these phases.

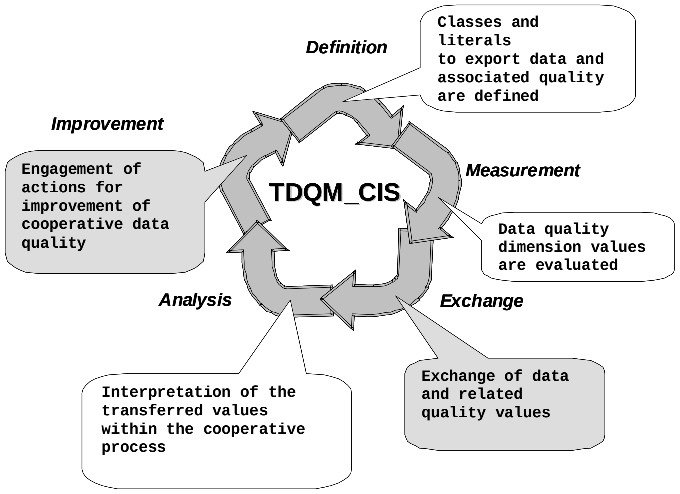
Definition
Classes and literals to export data and associated quality are defined
Improvement
Engagement of actions for improvement of cooperative data quality
TDQM_CIS
Measurement
Data quality dimension values are evaluated
Analysis
Exchange
Interpretation of the transferred values within the cooperative process
Exchange of data and related quality values

Fig. 2. The phases of the TDQM_CIS cycle; grey phases are specifically considered in this work

4 Design of the Architecture

In this section, we describe an *e*-Service-based architecture for management and improvement of data quality in CIS's. Cooperating organizations export data that are interesting for some other organizations, e.g., to carry out specific business processes they are involved in. Specifically, each organization offers *e*-Services allowing other organizations to query (a view of) its own internal data.

Cooperative data schemas define the structure of exported data; therefore the availability of such schemas and the opportunity of querying them constitutes the service offered to other organizations in order to cooperate. In addition to data schemas, each organization exports cooperative data quality schemas, which describe the quality of the exported data [4]. Both data schemas and quality schemas are described according to a global model, agreed upon by all organizations involved in a given CIS.

The architecture is shown in Figure 3. Offered *e*-Services, that is application components deployed on *cooperative gateways*, need to be published in the *repository*, in which for each data type the list of organizations (i.e., *e*-Services) providing it is stored.

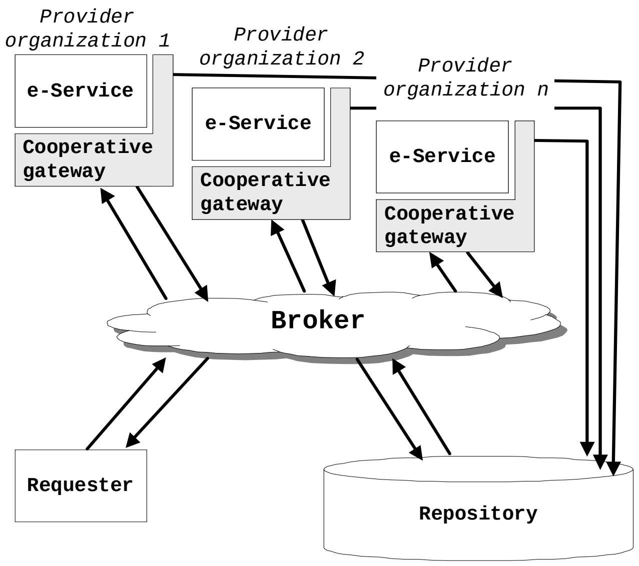

Fig. 3. The *e*-Service-based architecture supporting data quality

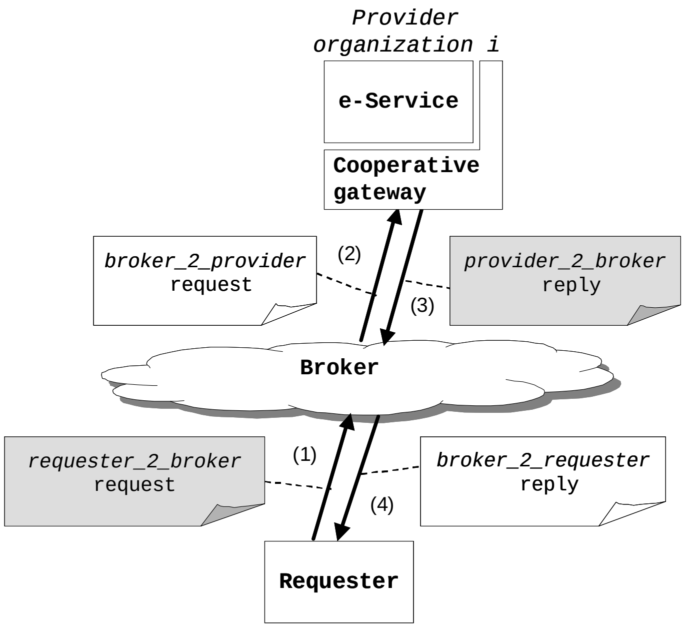

Fig. 4. Interactions broker-provider; grey elements are detailed and their definitions are in the Appendix

4.1 The Broker

When an organization needs to obtain some data, a *requester_2_broker request* is issued to the broker (see *(1)* in Figure 4); the requester_2_broker request specifies *(i)* which data are asked for (through a query in an appropriate query language), and *(ii)* quality requirements on acceptable replies.

The broker in turn accesses the repository in order to discover which *e*-Services are able to provide the requested data (i.e., to answer the query). Then the broker queries these *e*-Services (through *broker_2_provider requests*, see *(2)* in Figure 4) and obtains *provider_2_broker replies* which contain data and related quality values (see *(3)* in Figure 4).

The broker selects and orders replies by matching quality values against quality requirements, and finally sends back acceptable replies to the requester through a *broker_2_requester reply* (see *(4)* in Figure 4).

Both provider_2_broker replies and the broker_2_requester reply contain single type data items, that is either a single instance of a class/literal, or a list of instances of the same class/literal.

The *requester_2_broker request* Structure. In the phase of data request formulation, the requester organization specifies the quality requirements that the requested data need to satisfy. Specifically, the requester_2_broker request contains:

❒ A data query.
❒ Data quality requirements. A data quality requirement is an expression `< dimension > >= < value >`, where `< dimension >` is any data quality dimension but the source reliability and `< value >` is defined in the domain of the data quality dimension. Therefore a requirement specifies the minimum acceptable value for a given dimension. As an example of a data quality requirement, it is possible to specify for a data request that a `Citizen` data item needs to have `Syntactic_Accuracy` $>=$ `6`.
❒ Information allowing ordering the potential multiple replies. Specifically, the requester organization associates different weights to the specified quality requirements.
❒ The maximum number of replies satisfying the quality requirements that the requester organization would like to receive.

As regards data quality metrics, in this work we assume all data quality dimensions have an integer domain ranging from 1 to 10. This is a simplifying assumption: in future works we will investigate different domains for each dimension and for different types of data. From a technological point of view, the data request is an XML document, conform to the DTD shown in the Appendix.

The *provider_2_broker reply* Structure. The reply that a provider sends back to the broker includes both the requested data and the associated quality values. Mechanisms to calculate quality values for sets of classes and sets of literals (i.e., for aggregated data) starting from quality values of their properties are currently under investigation, on the basis of graph-based algorithms. The reply is an XML document, conform to the DTD shown in the Appendix.

As far as the structures of broker_2_provider request and broker_2_requester reply, they contain only data, and therefore they are analogous to the data parts of the previously described structures.

Selecting Replies. Upon receiving the replies to a data request, the broker makes two sequential activities:

❒ Matching of the data quality values of the received data against the quality requirements specified in the data request. All the replies that do not satisfy such requirements are discarded.

❒ Ordering the accepted replies. The accepted replies can be ordered on the basis of two factors: *(i)* the weights of the quality requirements, specified in the data request; *(ii)* an "evaluation" of the quality values specified in the data replies. The evaluation requires to consider the values of the provider organization as far as the source reliability dimension (stored in the repository, see Section 4.2); as an example, if for a specific data item, a syntactic accuracy value equal to `9` is provided by an organization with a source reliability value equal to `3`, then the effective accuracy value should be minor than `9`.

4.2 Other Elements

Some other elements are needed for the broker to carry out its work, specifically a *repository* and an *improvement manager*.

❒ The *repository* stores both *(i)* *e*-Service directory data, that is for each data type, all the *e*-Services (i.e., organizations) that can provide it are listed, and *(ii)* source reliability information, that is for each organization, a couple `< source reliability value, organization >` is maintained. Such values are dynamically used by the broker to weight quality values declared by each organization. The management of source reliability information should be conform to specific policies on which cooperating organizations have agreed upon. As an example, all the organizations can decide to trust quality declared at an initial instant `t0`, and assign the maximum source reliability value to each organization. These values can be eventually decreased when the declared quality is compared to the quality declared by other organizations.

❒ The *improvement manager* monitors the quality of exchanged data. Specifically, when an organization issues a request for data, several cases can occur, each of which corresponds to different actions that the improvement manager can engage:

Case 1 : the requested data are provided only by one organization, and the broker receives a reply from such a provider that fits the specified quality requirements. No improvement action is taken.

Case 2 : independently of the number of providers for the requested data, the broker receives replies such that all or part of them fit the specified quality requirements. The improvement manager notifies all the organizations that have provided replies with quality inferior than the requested quality. The same replies which are sent back to the requester by the broker, are also sent to them by the improvement manager.
On the basis of such notifications, organizations can take improvement actions internally. The exploitation of dynamic quality checks of data, every time a data request is issued, leads to a quality improvement of data on the long term.

Case 3 : none of the replies received by the broker fits the specified quality requirements. The broker can simply notify this to the requester, but the improvement manager can engage a merging activity in order to obtain a better quality copy of data, starting from the available ones. A lot of work about this task has been done in the data cleaning literature (see Section 2).

5 Example

In the following, we consider the Italian *e*-Government scenario [3], in order to provide an example of a possible use of the broker.

Let us consider a generic process in which a citizen goes to a public office (PO) to initiate a process he is interested in; currently, his personal data, including his residence address, are directly asked to him, by requesting him to present a certificate issued by a City Council. Current Italian *e*-Government projects aim at creating a CIS among public administrations, in which each administration offers *e*-Services to others; on the basis of such initiatives, this process can be reengineered by supposing that the residence address of the citizen is not provided by him, but the PO issues a request to the broker, in order to obtain the requested residence address from other organizations which are known to have such data; let us suppose these organizations are the Electric Power Company and the Department of Finance.

The PO makes a request to the broker, in which it specifies the quality requirements on the citizen's address, together with their weights; in Figure 5 the XML document of the `PO_2_broker request` is shown, and requirements and weights are highlighted.

After accessing the repository in order to discover providers for citizen addresses, the broker queries the two organizations and then receives the replies with the associated quality from the Electric Power Company and the Department of Finance; in Figure 6 and 7 the XML documents of the `provider_2_broker replies` are shown, and quality values are highlighted.

Finally the broker filters the received replies on the basis of the quality requirement satisfaction, and orders the filtered replies before sending them back to the PO. Specifically, upon receiving the quality values inside the replies, the broker can apply some rules to order the replies that satisfy the quality requirements on the basis of the weights specified for such requirements. Other

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Q-Query SYSTEM "http://www.x.y.z/PO_2_BrokerRequest">
<Q-Query>
    <Query> SELECT ResidenceAddress FROM Citizen WHERE SSN="SCNMNC74S60G230T"</Query>
    <Q-Requirements>
        <SyntacticAccuracy min_value="8" weight="0.3"/>
        <SemanticAccuracy weight="0"/>
        <Completeness min_value="6" weight="0.1"/>
        <InternalConsistency weight="0"/>
        <Currency min_value="9" weight="0.6"/>
    </Q-Requirements>
    <Constraints>
        <maxResults>"2"</maxResults>
    </Constraints>
</Q-Query>
```

(a) XML document

Dimension	**Accuracy**	**Currency**	**Completeness**
`Requirement`	8	9	6
`Weight`	0.3	0.6	0.1

(b) Requirements and Weights

Fig. 5. The `PO_2_broker request` XML document

parameters could be considered in this phase; as an example, source reliability values could be taken into account, in order to evaluate the reliability of the specified values.

In the current example, the broker simply orders the received replies on the basis of a *total quality value (TQV)*, calculated as:

$$Total\ Quality\ Value = \sum_{i \in \{accuracy, currency, completeness\}} quality\ value_i * weight_i$$

According to the values shown in previous figures, the total quality values are:

$$TQV\ Electric\ Power\ Company = 0,3 * 8 + 0,6 * 9 + 0,1 * 6 = 8,4$$

$$TQV\ Department\ of\ Finance = 0,3 * 9 + 0,6 * 10 + 0,1 * 7 = 9,4$$

Therefore, the broker sends to the PO the address provided by the Department of Finance as the best quality one and the address provided by Electric Power Company as a possible second choice.

6 Conclusions and Further Work

In this paper, the design of an *e*-Service-based architecture for brokering and improving data quality has been presented. *e*-Business applications are inher-

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE QueryResult SYSTEM "http://www.x.y.z/Provider_2_Broker.dtd">
<QueryResult>
    <Data>
        <DataElement>
            <Literal label="ResidenceAddress" SynAccuracy="o01" Completeness="o02" Currency="o03">
                <Attribute label="Number" value="" SynAccuracy="o04" Completeness="o08" Currency="o12"/>
                <Attribute label="Street" value="G. Mazzini" SynAccuracy="o05" Completeness="o09"
Currency="o13"/>
                <Attribute label="City" value="Rome" SynAccuracy="o06" Completeness="o10" Currency="o14"/>
                <Attribute label="Country" value="Italy" SynAccuracy="o07" Completeness="o11"
Currency="o15"/>
            </Literal>
        </DataElement>
    </Data>
    <QualityData>
        <QualityElement>
            <SynAccuracy>
                <SynAccuracyLiteral label="SynAccuracyResidenceAddress " degree="8" OID="o01">
                    <SynAccuracyAttribute label="SynAccuracyNumber" OID="o04"/>
                    <SynAccuracyAttribute label="SynAccuracyStreet" degree="7" OID="o05"/>
                    <SynAccuracyAttribute label="SynAccuracyCity" degree="9" OID="o06"/>
                    <SynAccuracyAttribute label="SynAccuracyCountry" degree="10" OID="o07"/>
                </SynAccuracyLiteral>
            </SynAccuracy>
            <Completeness>
                <CompletenessLiteral label="CompletenessResidenceAddress" degree="6" OID="o02">
                    <CompletenessAttribute label="CompletenessNumber" degree="0" OID="o08"/>
                    <CompletenessAttribute label="CompletenessStreet" degree="1" OID="o09"/>
                    <CompletenessAttribute label="CompletenessCity" degree="1" OID="o10"/>
                    <CompletenessAttribute label="CompletenessCountry" degree="1" OID="o11"/>
                </CompletenessLiteral>
            </Completeness>
            <Currency>
                <CurrencyLiteral label="CurrencyResidenceAddress" degree="9" OID="o03">
                    <CurrencyAttribute label="CurrencyNumber" OID="o12"/>
                    <CurrencyAttribute label="CurrencyStreet" degree="9" OID="o13"/>
                    <CurrencyAttribute label="CurrencyCity" degree="9" OID="o14"/>
                    <CurrencyAttribute label="CurrencyCountry" degree="9" OID="o15"/>
                </CurrencyLiteral>
            </Currency>
        </QualityElement>
    </QualityData>
</QueryResult>
```

(a) XML document

Accuracy	**Currency**	**Completeness**
8	9	6

(b) Quality values

Fig. 6. The `ElectricPowerCompany_2_broker reply` XML document

ently very complex and adding quality elements to them is not a trivial task. However, being the requirements that such applications need to satisfy more and more exacting, people do not yet be satisfied by a simple “response”, they want the *best* available one.

The aim of the brokering service is just to make accessible best available data. Once gathered all available copies of the same data, improvement actions can (and should) be enacted. In future work, features related to improvement actions will be more deeply studied. Specifically, algorithms to improve the quality of cooperative data will be considered, on the basis of the quality data that are transferred together with data items. Reconciliation and merging algorithms can be investigated by using quality data.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE QueryResult SYSTEM "http://www.x.y.z/Provider_2_Broker.dtd">
<QueryResult>
    <Data>
        <DataElement>
            <Literal label="ResidenceAddress"
                        SynAccuracy="o01" Completeness="o02" Currency="o03">
                <Attribute label="Number" value="27"
                    SynAccuracy="o04" Completeness="o08" Currency="o12"/>
                <Attribute label="Street" value="Giuseppe Mazzini"
                    SynAccuracy="o05" Completeness="o09" Currency="o13"/>
                <Attribute label="City" value="Rome"
                    SynAccuracy="o06" Completeness="o10" Currency="o14"/>
                <Attribute label="Country" value="Italy"
                    SynAccuracy="o07" Completeness="o11" Currency="o15"/>
            </Literal>
        </DataElement>
    </Data>
    <QualityData>
        <QualityElement>
            <SynAccuracy>
                <SynAccuracyLiteral label="SynAccuracyResidenceAddress " degree="9" OID="o01">
                    <SynAccuracyAttribute label="SynAccuracyNumber" degree="9" OID="o04"/>
                    <SynAccuracyAttribute label="SynAccuracyStreet" degree="10" OID="o05"/>
                    <SynAccuracyAttribute label="SynAccuracyCity" degree="9" OID="o06"/>
                    <SynAccuracyAttribute label="SynAccuracyCountry" degree="8" OID="o07"/>
                </SynAccuracyLiteral>
            </SynAccuracy>
            <Completeness>
                <CompletenessLiteral label="CompletenessResidenceAddress" degree="7" OID="o02">
                    <CompletenessAttribute label="CompletenessNumber" degree="1" OID="o08"/>
                    <CompletenessAttribute label="CompletenessStreet" degree="1" OID="o09"/>
                    <CompletenessAttribute label="CompletenessCity" degree="1" OID="o10"/>
                    <CompletenessAttribute label="CompletenessCountry" degree="1" OID="o11"/>
                </CompletenessLiteral>
            </Completeness>
            <Currency>
                <CurrencyLiteral label="CurrencyResidenceAddress" degree="10" OID="o03">
                    <CurrencyAttribute label="CurrencyNumber" degree="10" OID="o12"/>
                    <CurrencyAttribute label="CurrencyStreet" degree="10" OID="o13"/>
                    <CurrencyAttribute label="CurrencyCity" degree="10" OID="o14"/>
                    <CurrencyAttribute label="CurrencyCountry" degree="10" OID="o15"/>
                </CurrencyLiteral>
            </Currency>
        </QualityElement>
    </QualityData>
</QueryResult>
```

(a) XML document

Accuracy	**Currency**	**Completeness**
9	10	7

(b) Quality values

Fig. 7. The `DepartmentOfFinance_2_broker reply` XML document

Moreover, methods to derive quality values of classes and literals starting from atomic quality values of attributes are under investigation. For each quality dimension, the purpose is to derive global quality values for transferred data structures (i.e., classes, literals, list of classes and list of literals).

Source reliability management also needs to be further considered. In this paper, we have simply supposed that source reliability values are stored in a repository. As regards the evaluation of such a dimension, a possibility is to consider a certification organization, assessing source reliability based on objective parameters. In future work, source reliability evaluation in *e*-Business scenarios will be considered, thus also addressing the new requirements of the "Web of Trust".

Acknowledgments

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

The author would like to thank Barbara Pernici for important discussions about some issues addressed in this paper.

References

1. Ariba, Microsoft, and IBM, *Web Services Description Language (WSDL) 1.1*, W3C Note. Available on line (link checked October, 1st 2001): `http://www.w3.org/TR/2001/NOTE-wsdl-20010315`, March 2001.
2. C. Batini, M. Lenzerini, and S.B. Navathe, *Comparison of Methodologies for Database Schema Integration*, ACM Computing Surveys **18** (1986), no. 4.
3. C. Batini and M. Mecella, *Enabling Italian* e-*Government Through a Cooperative Architecture*, IEEE Computer **34** (2001), no. 2.
4. P. Bertolazzi, M.G. Fugini, M. Mecella, B. Pernici, G. Plebani, and M. Scannapieco, *Supporting Trusted Data Exchanges in Cooperative Information Systems*, Technical Report 30-2001, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Roma, Italy, 2001, submitted to international journal.
5. P. Bertolazzi and M. Scannapieco, *Introducing Data Quality in a Cooperative Context*, Proceedings of the 6th International Conference on Information Quality (IQ'01), Boston, MA, USA, 2001.
6. D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati, *Information Integration: Conceptual Modeling and Reasoning Support*, Proceedings of the 6th International Conference on Cooperative Information Systems (CoopIS'98), New York City, NY, USA, 1998.
7. F. Casati, D. Georgakopoulos, and M.C. Shan (eds.), *Proceedings of the 2nd VLDB International Workshop on Technologies for* e-*Services (VLDB-TES 2001)*, Rome, Italy, 2001.
8. F. Casati and M.C. Shan, *Dynamic and Adaptive Composition of* e-*Services*, Information Systems **6** (2001), no. 3.
9. J. Castro, M. Kolp, and J. Mylopoulos, *Towards Requirements-Driven Information Systems Engineering*, To appear in Information Systems, 2002.
10. R.G.G. Cattell and D.K. Barry (eds.), *The Object Database Standard: ODMG 2.0*, Morgan Kaufmann Publishers, 1997.
11. T. Dasu, T. Johnson, S. Muthukrishnan, and V. Shkapenyuk, *Mining Database Structure or, How to Build a Data Quality Browser*, Proceedings of the 2002 ACM/SIGMOD Conference, Madison, WI, USA, 2002.

12. U. Dayal, M. Hsu, and R. Ladin, *Business Process Coordination: State of the Art, Trends and Open Issues*, Proceedings of the 27th Very Large Databases Conference (VLDB 2001), Roma, Italy, 2001.
13. G. De Michelis, E. Dubois, M. Jarke, F. Matthes, J. Mylopoulos, M.P. Papazoglou, K. Pohl, J. Schmidt, C. Woo, and E. Yu, *Cooperative Information Systems: A Manifesto*, Cooperative Information Systems: Trends & Directions (M.P. Papazoglou and G. Schlageter, eds.), Accademic-Press, 1997.
14. L. English, *Improving Data Warehouse and Business Information Quality*, Wiley & Sons, 1999.
15. H. Galhardas, D. Florescu, D. Shasha, and E. Simon, *An Extensible Framework for Data Cleaning*, Proceedings of the 16th International Conference on Data Engineering (ICDE 2000), San Diego, CA, USA, 2000.
16. M.A. Jeusfeld, C. Quix, and M. Jarke, *Design and Analysis of Quality Information for Data Warehouses*, Proceedings of the 17th International Conference on Conceptual Modeling (ER'98), Singapore, Singapore, 1998.
17. M. Lenzerini, *Data Integration Is Harder than You Thought*, Invited Talk at the 9th International Conference on Cooperative Information Systems (CoopIS 2001), Trento, Italy, 2001.
18. S. Madnick, *Metadata Jones and the Tower of Babel: The Challenge of Large-Scale Semantic Heterogeneity*, Proceedings of the 3rd IEEE Meta-Data Conference (Meta-Data '99), Bethesda, MA, USA, 1999.
19. M. Mecella and B. Pernici, *Designing Wrapper Components for* e-*Services in Integrating Heterogeneous Systems*, VLDB Journal **10** (2001), no. 1, (A preliminary version also in Proceedings of the 1st VLDB International Workshop on Technologies for *e*-Services (VLDB-TES 2000)).
20. M. Mecella, M. Scannapieco, A. Virgillito, R. Baldoni, T. Catarci, and C. Batini, *Managing Data Quality in Cooperative Information Systems*, submitted to international conference, 2002.
21. G. Mihaila, L. Raschid, and M. Vidal, *Querying Quality of Data Metadata*, Proceedings of the 6th International Conference on Extending Database Technology (EDBT'98), Valencia, Spain, 1998.
22. R.C. Morey, *Estimating and Improving the Quality of Information in the MIS*, Communications of the ACM **25** (1982), no. 5.
23. T. Pilioura and A. Tsalgatidou, e-*Services: Current Technologies and Open Issues*, Proceedings of the 2nd VLDB International Workshop on Technologies for *e*-Services (VLDB-TES 2001), Rome, Italy, 2001.
24. T.C. Redman, *Data Quality for the Information Age*, Artech House, 1996.
25. G. Shankaranarayan, R.Y. Wang, and M. Ziad, *Modeling the Manufacture of an Information Product with IP-MAP*, Proceedings of the 5th International Conference on Information Quality (IQ'00), Boston, MA, USA, 2000.
26. J.D. Ulmann, *Information Integration using Logical Views*, Proceedings of 6th International Conference on Database Theory (ICDT '97), Delphi, Greece, 1997.
27. P. Vassiliadis, M. Bouzeghoub, and C. Quix, *Towards Quality-Oriented Data Warehouse Usage and Evolution*, Proceedings of the 11th International Conference on Advanced Information Systems Engineering (CAISE'99), Heidelberg, Germany, 1999.
28. Y. Wand and R.Y. Wang, *Anchoring Data Quality Dimensions in Ontological Foundations*, Communications of the ACM **39** (1996), no. 11.
29. R.Y. Wang, *A Product Perspective on Total Data Quality Management*, Communications of the ACM **41** (1998), no. 2.

30. J. Yang, W.J. Heuvel, and M.P. Papazoglou, *Tackling the Challenges of Service Composition in* e-*Marketplaces*, Proceedings of the 12th International Workshop on Research Issues on Data Engineering: Engineering E-Commerce/E-Business Systems (RIDE-2EC 2002), San Jose, CA, USA, 2002.

Appendix

DTD of the `requester_to_broker` Request

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT Q-Query (Query, Q-Requirements, Constraints)>
<!ELEMENT Query (#PCDATA)>
<!ELEMENT Q-Requirements (SyntacticAccuracy, SemanticAccuracy, Completeness, InternalConsistency, Currency)>
<!ELEMENT SyntacticAccuracy EMPTY>
<!ATTLIST SyntacticAccuracy
          min_value (1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10) #IMPLIED
          weight CDATA #REQUIRED
>
<!ELEMENT SemanticAccuracy EMPTY>
<!ATTLIST SemanticAccuracy
          min_value (1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10) #IMPLIED
          weight CDATA #REQUIRED
>
<!ELEMENT Completeness EMPTY>
<!ATTLIST Completeness
          min_value (1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10) #IMPLIED
          weight CDATA #REQUIRED
>
<!ELEMENT InternalConsistency EMPTY>
<!ATTLIST InternalConsistency
          min_value (1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10) #IMPLIED
          weight CDATA #REQUIRED
>
<!ELEMENT Currency EMPTY>
<!ATTLIST Currency
          min_value (1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10) #IMPLIED
          weight CDATA #REQUIRED
>
<!ELEMENT Constraints (maxResults?)>
<!ELEMENT maxResults (#PCDATA)>
```

DTD of the `provider_to_broker` Reply

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT QueryResult (Data, QualityData)>
<!ELEMENT Data (DataElement)*>
<!ELEMENT DataElement (Class | Literal | Attribute)>
<!ELEMENT Class (Attribute* | Literal*)+>
<!ATTLIST Class
          label CDATA #REQUIRED
          SynAccuracy IDREF #IMPLIED
          SemAccuracy IDREF #IMPLIED
          Completeness IDREF #IMPLIED
          IntConsistency IDREF #IMPLIED
          Currency IDREF #IMPLIED
>
<!ELEMENT Literal (Attribute* | Literal*)+>
<!ATTLIST Literal
          label CDATA #REQUIRED
          SynAccuracy IDREF #IMPLIED
          SemAccuracy IDREF #IMPLIED
          Completeness IDREF #IMPLIED
          IntConsistency IDREF #IMPLIED
          Currency IDREF #IMPLIED
>
<!ELEMENT Attribute EMPTY>
<!ATTLIST Attribute
          label CDATA #REQUIRED
          value CDATA #IMPLIED
          SynAccuracy IDREF #IMPLIED
          SemAccuracy IDREF #IMPLIED
          Completeness IDREF #IMPLIED
          Currency IDREF #IMPLIED
```

```
<!ELEMENT QualityData (QualityElement)*>
<!ELEMENT QualityElement (SynAccuracy?, SemAccuracy?, Completeness?, IntConsistency?, Currency?)>
<!ELEMENT SynAccuracy (SynAccuracyClass | SynAccuracyAttribute | SynAccuracyLiteral)>
<!ELEMENT SemAccuracy (SemAccuracyClass | SemAccuracyAttribute | SemAccuracyLiteral)>
<!ELEMENT IntConsistency (IntConsistencyClass | IntConsistencyLiteral)>
<!ELEMENT Completeness (CompletenessClass | CompletenessAttribute | CompletenessLiteral)>
<!ELEMENT Currency (CurrencyClass | CurrencyAttribute | CurrencyLiteral)>
<!ELEMENT SynAccuracyClass (SynAccuracyAttribute* | SynAccuracyLiteral*)>
<!ATTLIST SynAccuracyClass
          label CDATA #REQUIRED
          degree (1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10) #REQUIRED
          OID ID #REQUIRED
>
<!ELEMENT SynAccuracyLiteral (SynAccuracyAttribute* | SynAccuracyLiteral*)>
<!ATTLIST SynAccuracyLiteral
          label CDATA #REQUIRED
          degree (1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10) #REQUIRED
          OID ID #REQUIRED
>
<!ELEMENT SynAccuracyAttribute EMPTY>
<!ATTLIST SynAccuracyAttribute
          label CDATA #REQUIRED
          degree (1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10) #IMPLIED
          OID ID #REQUIRED
>
<!ELEMENT SemAccuracyClass (SemAccuracyAttribute* | SemAccuracyLiteral*)>
<!ATTLIST SemAccuracyClass
          label CDATA #REQUIRED
          degree (1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10) #REQUIRED
          OID ID #REQUIRED
>
<!ELEMENT SemAccuracyLiteral (SemAccuracyAttribute* | SemAccuracyLiteral*)>
<!ATTLIST SemAccuracyLiteral
          label CDATA #REQUIRED
          degree (1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10) #REQUIRED
          OID ID #REQUIRED
>
<!ELEMENT SemAccuracyAttribute EMPTY>
<!ATTLIST SemAccuracyAttribute
          label CDATA #REQUIRED
          degree (1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10) #IMPLIED
          OID ID #REQUIRED
>
<!ELEMENT CompletenessClass (CompletenessAttribute* | CompletenessLiteral*)>
<!ATTLIST CompletenessClass
          label CDATA #REQUIRED
          degree (1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10) #REQUIRED
          OID ID #REQUIRED
>
<!ELEMENT CompletenessLiteral (CompletenessAttribute* | CompletenessLiteral*)>
<!ATTLIST CompletenessLiteral
          label CDATA #REQUIRED
          degree (1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10) #REQUIRED
          OID ID #REQUIRED
>
<!ELEMENT CompletenessAttribute EMPTY>
<!ATTLIST CompletenessAttribute
          label CDATA #REQUIRED
          degree (0 | 1) #REQUIRED
          OID ID #REQUIRED
>
<!ELEMENT IntConsistencyClass (IntConsistencyClass | IntConsistencyLiteral*)>
<!ATTLIST IntConsistencyClass
          label CDATA #REQUIRED
          degree (1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10) #REQUIRED
          OID ID #REQUIRED
>
<!ELEMENT IntConsistencyLiteral EMPTY>
<!ATTLIST IntConsistencyLiteral
          label CDATA #REQUIRED
          degree (1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10) #REQUIRED
          OID ID #REQUIRED
>
<!ELEMENT CurrencyClass (CurrencyAttribute* | CurrencyLiteral*)>
<!ATTLIST CurrencyClass
          label CDATA #REQUIRED
          degree (1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10) #REQUIRED
          OID ID #REQUIRED
>
<!ELEMENT CurrencyLiteral (CurrencyAttribute* | CurrencyLiteral*)>
<!ATTLIST CurrencyLiteral
          label CDATA #REQUIRED
          degree (1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10) #REQUIRED
          OID ID #REQUIRED
>
<!ELEMENT CurrencyAttribute EMPTY>
<!ATTLIST CurrencyAttribute
          label CDATA #REQUIRED
          degree (1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10) #IMPLIED
          OID ID #REQUIRED
>
```