

Data Quality Notification in Cooperative Information Systems

Carlo Marchetti*, Massimo Mecella*, Monica Scannapieco*[†], Antonino Virgillito*, Roberto Baldoni*

*Università di Roma “La Sapienza”

Dipartimento di Informatica e Sistemistica

Email: {marchet, mecella, monscan, virgi, baldoni} @dis.uniroma1.it

[†]Consiglio Nazionale delle Ricerche

Istituto di Analisi dei Sistemi ed Informatica (IASI-CNR)

Abstract— Current approaches to the development of cooperative information systems are based on services to be offered by cooperating organizations, and on the opportunity of building coordinators and brokers on top of such services. The quality of data exchanged and provided by different services hampers such approaches, as data of low quality can 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, we consider the problem of disseminating quality information. In particular, we describe the Quality Notification Service, whose objective is to provide cooperating entities with information concerning the quality variations of exchanged data. By exploiting the information supplied by the Quality Notification Service, other applications and services can be built or modified to benefit of notifications of quality variations.

I. INTRODUCTION

A *Cooperative Information System (CIS)* is a large scale information system that interconnects several heterogeneous systems belonging to different and autonomous organizations that are geographically distributed and share common objectives. Among the different resources shared by organizations, data are fundamental; in real world scenarios, an organization A may not request data from an organization B if it does not “trust” B data, i.e., if A does not know that the quality of the data that B can provide is sufficiently high for its purposes.

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.

CIS’s designed according to a service-based approach [13] 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* [11], are exported by different organizations as well

defined operations that allow users and applications to access and perform tasks offered by back-end business applications.

In previous works [23], [22], we proposed (i) a data model to export data with associated quality information and (ii) a service-based architecture for managing data quality in CIS’s. These instruments provide support for data quality diffusion and improvement, and to avoid dissemination of low qualified data through the CIS. In particular, the overall architecture comprises several services including a Data Quality Broker, a Quality Notification Service and a Rating Service. In this paper we focus on the Quality Notification Service (QNS). The main objective of QNS is to provide users with information concerning variations of the quality of data stored within the CIS. Through QNS, users can declare the information they are interested to receive, and the service guarantees users to receive the information for which they expressed interest. The design of this service passes through the definition of a language suitable to declare the information of interest, and through the analysis of the possible implementations. In this work, we first define a simple language and then we show how a QNS supporting this language can be implemented basing on a *publish/subscribe* middleware platform [15]. Due to the heterogeneity of both capabilities and limitations of existing publish/subscribe platforms, we analyze several possibilities of implementing QNS.

The remainder of the paper is organized as follows. Section V discusses the related research work. Section II describes the overall architecture addressing quality related issues, and summarizes the data quality model on which the system is based on. Section III describes the Quality Notification Service. Finally, Section VI concludes the paper and outlines some future research directions.

II. BACKGROUND

In [23], an architecture for the management of data quality in CIS’s is proposed. This architecture aims to avoid dissemination of low qualified data through the CIS, by providing support for data quality diffusion and improvement. The basic assumption of this architecture is that all cooperating organizations export their application

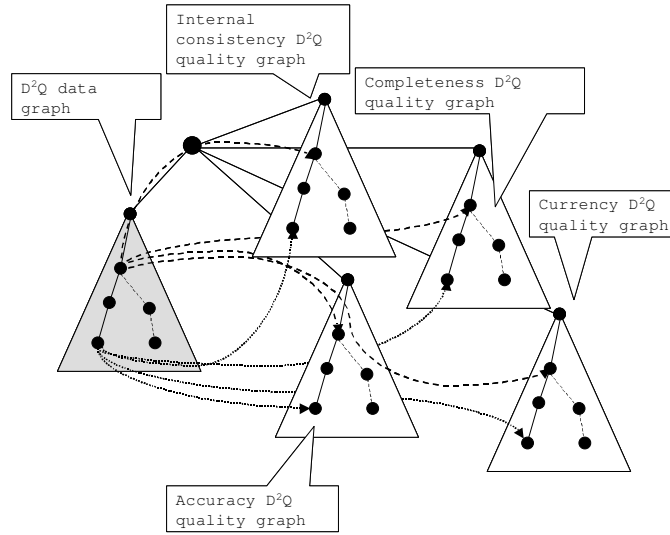


Fig. 1. The generic structure of a D^2Q XML document, returned as result by a service operation

data and quality data according to a data model common to all cooperating organizations, by using a Global As View schema approach [7], [34]. Exported data and quality data can be accessed by other organizations by means of services that each cooperating organization makes available to the others. The model for exporting data and quality data is referred to as *Data and Data Quality (D^2Q) model*. Such a model 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. In order to produce data and quality data according to the D^2Q model, each organization holds a *Quality Factory*, that is the architectural component responsible for evaluating the quality of its own data.

In this section we summarize the quality dimensions on which the data model is based on and the data model itself, then we give an overview of the overall architecture for data quality management.

A. 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 [28].

In this section, we propose and outline some data quality dimensions to be used in CIS's, stemming from real requirements of CIS's scenarios that we experienced [3]. 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:

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

- **Completeness.** It is the degree to which values of a schema element are present in the schema element instance.
- **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. Therefore, currency can be defined as the “age” of a value. Namely, 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.
- **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 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.

The reader should refer to [22] for complete definitions, examples and possible evaluation methods related to each of them.

B. The D^2Q Model

The D^2Q model is inspired by the data model underlying XML-QL [14]. 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 II-A.

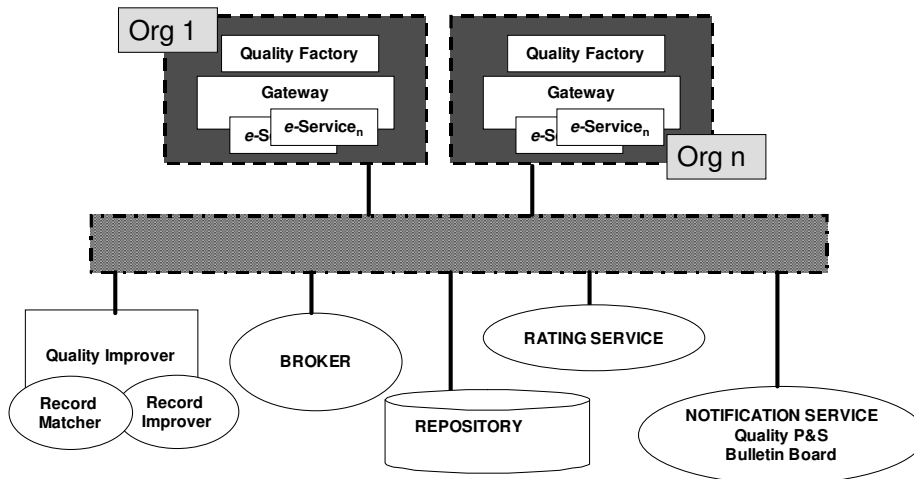


Fig. 2. An architecture for data quality diffusion and improvement

Specifically, 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, and quality classes and objects are represented as D^2Q quality graphs, as detailed in [23].

Each node of the D^2Q data graph represents a single data item and it is linked to the corresponding ones of the D^2Q quality graphs through links, as shown in Figure 1.

C. Architecture for quality management in CIS's

The overall architecture for data quality management in CIS is depicted in Figure 2. In the following we give a short description of each component:

- **Data Quality Broker:** the broker performs, on behalf of a requesting organization, a data request in which a set of quality requirements that the desired data have to satisfy (*quality brokering function*) is specified. 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 Knowledge Repository:** it consists of a knowledge base used by the other components in order to perform their functions. For example, it maintains (i) the interschema knowledge representing the relationships among schemas that allow to determine intensional and extensional equivalence of data in different organizations [12], and (ii) historical quality knowledge, including statistics related to data quality ensured by organizations in the past, that is also used by the Rating Service to measure reliability of organizations in quality evaluation.
- **Quality Notification Service:** this service allows quality-based subscriptions for organizations to be notified for quality changes in data. For example, an organization may want to be notified if the quality of a data

it uses degrades below a certain acceptable threshold, or when high quality data are available.

- **Rating Service:** it associates reliability values to each data source in the CIS. These are used by the Data Quality Broker to determine the credibility of the quality evaluation made by organizations. Reliability values are calculated on the basis of feedbacks from users of data, that can submit to the Rating Service itself their own evaluation of the data received from the Data Quality Broker.
- **Quality Improver:** the quality improver comprises a collection of tools that allow organizations to improve the quality of their data. As an example, it contains the **Record Matcher** and a **Record Improver** that periodically compare and reconcile data spread throughout the system.

The architectural components represent the software support of a process for the overall improvement of data quality in a CIS. This process can be divided in three phases, corresponding to increasing levels of improvement and realized by different architectural components. We describe them below:

- **Off-line improvement:** implemented by the Quality Improver component. It consists of a periodical, off-line execution of a record matching algorithm over different databases of the CIS. The Quality Improver identifies non-matching duplicates and tries to reconcile them. This leads a first level of overall quality improvement in the whole CIS.
- **On-line improvement:** implemented by the Data Quality Broker. When answering to user queries, the Broker identifies the best quality data among all the duplicates present in the CIS. After the query, it propagates the best data through the CIS, thus organization hosting lower quality duplicates have the opportunity to improve their data. Being more frequent than the Quality Improver and acting on data already handled by the record matching algorithm, this process leads to a further, more refined

improvement of data quality.

- Quality Maintenance: implemented by the Quality Notification. A user can be notified about changes in the quality of data. This gives the opportunity to monitor quality of data and allows to maintain the overall quality in the CIS at an acceptable level.

In [23], we consider the Data Quality Broker. Work on other components is ongoing in the context of the DaQuinCIS project¹. In the remainder of this paper, we focus on the Quality Notification Service.

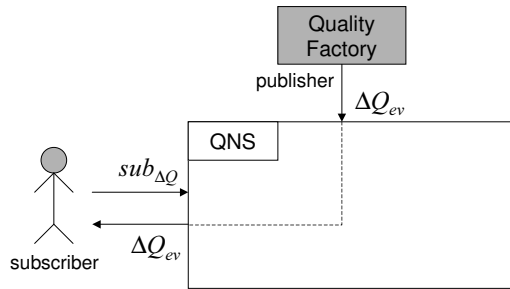


Fig. 3. Quality Notification Service

III. THE QUALITY NOTIFICATION SERVICE

The Quality Notification Service (QNS) is used in the context of our architecture to inform users when changes in quality values occur within the CIS. The interaction between QNS and its users follows the publish/subscribe paradigm [15]: a user willing to be notified for quality changes *subscribes* to the QNS by submitting the features of the events to be notified for, through a specific subscription language. When a change in quality happens, an event is *published* by the QNS i.e., all the users which have a consistent subscription receive a notification.

The QNS can be used by end-users or software systems in the CIS that use critical data, in order to keep track of its quality changes and be always aware when quality degrades under a certain threshold, which makes it no longer suited for the use it is devoted to. The QNS is also exploited by other architectural components to maintain the information they use to perform their service always updated. As an example, the Quality Improver could be triggered when the quality of specific sample data falls under a certain threshold.

A. Requirements

The design of QNS presents aspects related to a general purpose event service and also domain-dependent issues, that determine the interaction of the QNS with users in the CIS and architectural components.

First of all, the subscription language has to allow quality-based subscriptions where one of the quality dimensions is

¹“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/>).

constrained within a threshold value. The data granularity should be sufficient to subscribe for changes in quality either of a single property or of a whole data object. Finally, the language should allow more articulated subscriptions through comparison and aggregation operators. On the architectural point of view, a quality-related feature is defining how QNS notifications are produced i.e., the mechanism through which the change of a data object produces a new event in QNS. We simply let the QNS interact with the Quality Factory components inside each organization. Each Quality Factory (QF) reports to QNS the production of a new quality values or a change in an existing one.

From the point of view of notification diffusion, the QNS should face various problems in order to scale to the large size we can expect in a CIS context. For example, it is reasonable to assume a high number of possible participants, each issuing several unrelated subscriptions. This, together with the fine-grained granularity of the subscription language leads to frequent events and subsequent production of notifications. In general, the QNS has to be based on an efficient communication infrastructure, guaranteeing high notification throughput even under large service load.

B. Specification

Figure 3 depicts the high-level architecture of the QNS. Users interact with QNS by issuing subscriptions, using a subscription language that defines subscriptions of the form:

$$sub_{\Delta Q} (\mathbb{O}, \mathcal{D}, expr)$$

where:

- \mathbb{O} is the organization that maintains the desired data; if \mathbb{O} is equal to \perp , the subscription refers to all organizations in the CIS;
- \mathcal{D} is a data object; if \mathcal{D} is not specified (i.e., \perp), all data objects of a specific data class, specified in *expr* are considered;
- finally *expr* has the form $\langle \text{quality_dimension} \rangle \langle \text{operator} \rangle \langle \text{value} \rangle$, with $\langle \text{quality_dimension} \rangle$ referring to a property of \mathcal{D} ; if \mathcal{D} is \perp the entity to which the property belongs to must also be included; if *expr* is equal to \perp all changes for all properties are considered.

At least one of the parameters needs to be specified (i.e., not \perp). By exploiting the proposed language, it is possible to express a large number of different kinds of subscriptions, ranging from very specific ones (e.g., $sub_{\Delta Q} (\text{Ministry of Finance}, \text{Massimo Mecella} : \text{Citizen}, \text{name.Accuracy} \geq 0.7)$, which subscribes when the accuracy of the name of the data object Massimo Mecella is greater / equal to 0.7), to very generic ones (e.g., $sub_{\Delta Q} (\perp, \text{Massimo Mecella} : \text{Citizen}, \perp)$, which subscribes for all the changes regarding the data object Massimo Mecella, independently of the generating organization and the affected dimension, or $sub_{\Delta Q} (\perp, \text{Citizen.name.Accuracy} \geq 0.7)$, which subscribes

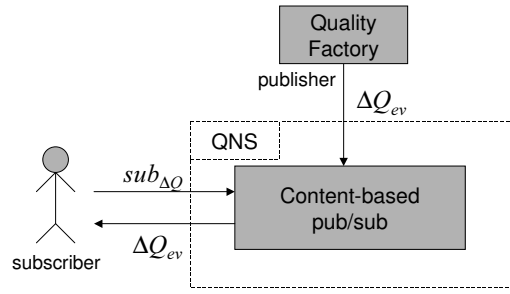


Fig. 4. Implementing QNS through a content-based pub/sub system

for all the changes in accuracy of all data objects of a single data class, independently of the organization).

The Quality Factory components in each organization push events to the QNS in a form that intuitively matches user's subscriptions:

$$\Delta Q_{ev} = \langle \mathcal{O}, d, (\text{quality_dimension}, \text{value}) \rangle$$

As an example, an event might be $\langle \text{Ministry of Finance}, \text{Massimo Mecella} : \text{Citizen}, (\text{name.Accuracy}, 0.9) \rangle$

IV. IMPLEMENTING QNS

From both the requirements and the specification of the QNS presented in previous section, it stems that the QNS functionality is analogous to that of a publish/subscribe communication system (pub/sub). It is then straightforward to implement QNS exploiting a COTS-based pub/sub middleware system. Existing pub/sub middleware can be classified as follows:

- *Topic-based* ([25], [33], [19]). Events are grouped in topics (or subject) i.e., a subscriber declare its interest for a particular topic and will receive all events related to that topic. Each topic corresponds to a logical *event channel* ideally connecting each possible publisher to all interested subscribers. That is, it exists a static association between a channel and all its subscribers, then when an event is published, the system does not have to calculate all the receivers. While this mechanism can drive to very efficient implementations, it limits expressiveness of topic-based pub/sub to a static scheme.
- *Content-based* ([10], [2], [32]). In this case, subscriptions are specified through constraints on properties of each event. In other words, each subscription is a filter composed by a set of constraints on the values of attributes of the event (e.g. “StockName = ‘IBM’ and change < -3”). Lacking each possibility of determining a priori the set of receivers for each notification, this has to be computed on a per-event basis. Then, the higher expressive power of content-based pub/sub comes at the price of the resource consumption needed to calculate for each published event the set of interested subscribers [9].

It is easy to see that a topic-based scheme can be possibly emulated through a content-based one, while the vicev-

ersa it is not true. In particular, the channel abstraction in the topic-based scheme cannot represent flexible features of the content-based scheme such as comparison operators or complex conjunctive subscriptions. In the remainder of this section we evaluate the feasibility of the two variants in the implementation of the QNS.

A. Content-based solutions

Due to the high expressive power of content-based pub/sub, implementing QNS with such a tool it is straightforward (Figure 4): it is sufficient to adapt the subscription language of QNS to the language of the specific system used i.e., rewrite each possible subscription expressible with QNS in terms of the language of the pub/sub system.

Concerning efficiency, let us note again that the generality of these tools imposes a performance penalty in terms of network overhead and storage requirements to hold subscriptions. As pointed out above, the content-based pub/sub has to (i) calculate for each message the set of receivers basing on all the subscriptions (“matching”) and to (ii) propagate notifications, possibly avoiding to disseminate events to parts of the network with no interested subscribers (“routing”). Due to this, research related to content-based systems has focused on efficient and scalable algorithms for matching [1], [17], [8] and routing [2], [10]. However, all these contributions assume that all subscriptions are propagated to the entire system. This can be not trivial in a CIS with a large number of organizations and users.

Content-based systems are inherently more costly that topic-based ones and their scalability remains an issue, especially in contexts where such high expressive power is not required. As a final remark, we point out that there are few available implementations of content-based pub/sub middleware. This also limits the possibility of building the global pub/sub system as an integration of the notification services used by each organization in the CIS, a common practice in real-world CIS implementations.

Given the above analysis, we argue that, despite the first impression, the suitability of content-based pub/sub systems to the CIS environment has to be further investigated.

B. Topic-based solutions

Many topic-based pub/sub tools are available on the market, featuring high efficiency in wide-area, large-scale environ-

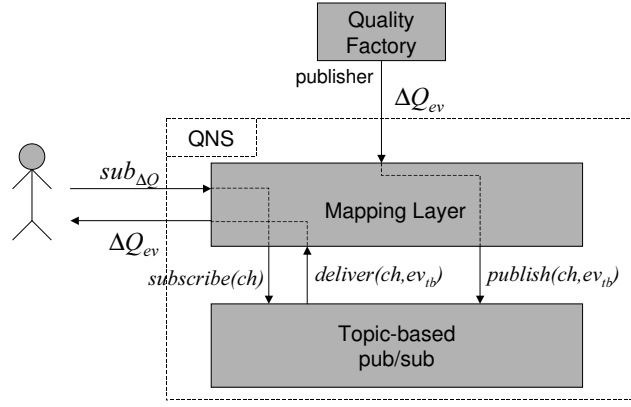


Fig. 5. Implementing QNS through a topic-based pub/sub system

ments, such as the communication infrastructure underlying a CIS.

However, implementing QNS through a topic-based system requires additional processing to emulate the content-based semantics. In particular, the subscription language of QNS must allow comparison operators that cannot be implemented using a topic-based solution. As explained in previous section, topic-based systems can only implement the abstraction of channels connecting all possible destinations of messages to a particular topic. Then, the problem is establishing a correspondence between each subscription and a channel.

Our proposal considers an intermediate layer in the QNS, the Mapping Layer (ML, Figure 5), logically placed between the clients of the service (subscribers and QFs) and the topic-based pub/sub system (TBPS). The Mapping Layer then behaves as a client for the pub/sub system and, interacting with it through the operations indicated in the Figure, it performs the following functions:

- 1) mapping QNS subscriptions $sub_{\Delta Q}$ to TBPS channels ch ;
- 2) deciding on which TBPS channels ch each QNS event ΔQ_{ev} should be published as a TBPS event ev_{tb} ;
- 3) delivering events ev_{tb} from TBPS to interested QNS subscribers in the form of QNS events ΔQ_{ev} (this point includes the implementation of the comparison operators on quality dimensions).

Different policies can be considered for structuring the channels with respect to the QNS subscription language. As an extreme solution, we can consider the TBPS as maintaining a single channel, serving all subscribers and on which all events are published. In this case, the ML receives all events from the TBPS and each event will have to be matched against all subscribers. If no subscriptions exist for an event, this will be not purged by the TBPS, generating “useless” network traffic. On the other hand, this solution requires low memory and computational power on the TBPS.

On the opposite side, we can consider each single subscription as mapped to a different channel. In this case, the ML performs no mapping at all on the sending and delivery, as each subscriber directly corresponds to one channel and

each QNS notification can be directly mapped to one or more channels. While no matching is required, the TBPS layer has to maintain an unpredictably high number of channels, with a subsequent high resource usage.

These two trivial solutions, though evidently unfeasible, allowed us to reason about the general problem of emulating a content-based system through a topic-based one, with respect to a specific application context. It turns out, as a very coarse-level intuition, that with a lower number of channels we obtain lower “precision”, defined in terms of the number of messages the ML receives from the TBPS and has to filter out before delivering. On the other hand, a high number of channels easily cloaks the TBPS level. The idea is optimizing the management of subscriptions in a hierarchical fashion in order to distribute the load among ML and TBPS.

The discussion above leads to identifying two cost functions, that can be used to evaluate a mapping policy: the number of channels (nc), measuring the cost due to the mapping policy, and the non-precision (np), measuring the cost due to the transit of useless traffic and matching at ML level. In the following we consider three different policies for structuring TBPS channels and give an intuitive evaluation of the cost functions. We also present examples of scenarios where each policy allows to obtain the best trade-off between nc and np .

- Channel-per-Entity: each channel is associated to an entity in the global schema. If a subscription exists considering a specific data object (e.g., $sub_{\Delta Q}(Ministry\ of\ Finance, Massimo\ Mecella : Citizen, name.Accuracy \geq 0.7)$), ML will receive, and will have to filter out, all the notifications related to all other instances of the entity, for each quality attribute. Independently from the subscription scenario, we expect nc to be high since it is equal to the number of entities in the CIS schema. np is as low as more subscriptions exist that are aggregated at entity level ($sub_{\Delta Q}(\perp, Citizen.name.Accuracy \geq 0.7)$). This choice is then suited for situations where few fine-grained subscriptions are present.
- Channel-per-Quality Dimension: each channel is associ-

ated to a different quality dimension of the D^2Q model. Being the number of dimensions in the model fixed to 4, nc is fixed and very low, but resulting in a very low precision (i.e., np can be very high). That is, the subscription in the previous point will provoke ML to receive all notifications related to all data objects of all entities. This solution is clearly under-optimized, not exploiting the features of the pub/sub system. Then, it can only make sense in situations where a principle of locality can be applied i.e., many subscribers are connected to a same ML requiring all variations on a same dimension.

- Channel-per-Organization: this solution differs from the other two in the sense that it is not based on a property of data, but on the topology of the CIS. A different channel is associated to each organization, and notifications for a data are published to the channel of the organization the data belongs to. This can be seen as a general purpose approach that is independent from the structure of data. nc is simply equal to the number of organization in the CIS. In this case, np does not depend on subscriptions but exclusively on the frequency of notification production: the finer-grained example subscription of previous points results in receiving all data changes from a same organization, then the smaller is the set of data that changes, the lower are the unwanted notifications received. Note also that in this case ML may have to know where each entity can be found in order to perform a correct mapping of subscriptions and publications.

V. RELATED WORK

Data Quality has been traditionally investigated in the context of single information systems; only recently, a methodological framework for data quality in cooperative systems has been proposed, consisting of five phases (i.e., definition, measurement, exchange, analysis and improvement) [4].

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, e.g., [18], [20]. Methods and techniques for exchanging quality information (ii) and for improvement (iii) have been only partially addressed in the literature. When considering the issue of exchanging data and the associated quality, a model to export data and quality data needs to be defined. In this work, we consider the one proposed in [23], in which novelties over previous approaches are also discussed.

For what concerns publish/subscribe communication systems, research has focused on algorithms to face large-size scenarios, considering for example efficient information diffusion ([16], [27], [29]) or efficient matching in content-based

systems ([1], [17], [8], [2]), while other works take into account QoS parameters such as reliable message delivery ([5]). Descriptions of practical experiences with pub/sub systems can be found in [6], [26].

VI. 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. In the context of an overall framework to support data quality management in CIS's, in this paper we discussed the design of the Quality Notification Service, a publish/subscribe-like communication service that notifies variations in quality of data in the CIS.

The reference architecture of the QNS presented in this paper considers an underlying publish/subscribe communication system used for diffusing quality notifications. We presented several solutions based on the two different variants of the publish/subscribe paradigm (content-based and topic-based), and gave an informal, qualitative analysis of their suitability to different application contexts. A more accurate, quantitative study is currently in progress and will be subject of future work.

ACKNOWLEDGMENTS.

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] M. K. Aguilera, R. E. Strom, D. C. Sturman, M. Astley, and T. D. Chandra. Matching Events in a Content-Based Subscription System. In *Proceedings of The Symposium on Principles of Distributed Computing*, pages 53–61, 1999.
- [2] G. Banavar, T. Chandra, B. Mukherjee, J. Nagarajarao, R.E. Strom, and D.C. Sturman. An Efficient Multicast Protocol for Content-based Publish-Subscribe Systems. In *Proceedings of International Conference on Distributed Computing Systems*, 1999.
- [3] C. Batini and M. Mecella, "Enabling Italian e-Government Through a Cooperative Architecture," *IEEE Computer*, vol. 34, no. 2, 2001.
- [4] 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.
- [5] K.P. Birman. "The Process Group Approach to Reliable Distributed Computing,". *Communications of the ACM*, vol. 12, no. 36, 1993.
- [6] K.P. Birman. "A Review of Experiences with Reliable Multicast,". *Software - Practice & Experiences*, vol. 9, no. 29, 1999.
- [7] A. Cali, D. Calvanese, G. De Giacomo and M. Lenzerini. On the Expressive Power of Data Integration Systems. In *ER-02*, 2002.
- [8] A. Campailla, S. Chaki, E. M. Clarke, S. Jha, and H. Veith. Efficient filtering in publish-subscribe systems using binary decision. In *Proceedings of The International Conference on Software Engineering*, pages 443–452, 2001.
- [9] A. Carzaniga, D.S. Rosenblum, and A.L. Wolf. Challenges for distributed event services: Scalability vs. Expressiveness. In *Engineering Distributed Objects '99*, Los Angeles CA, USA, May 1999.
- [10] A. Carzaniga, D.S. Rosenblum, and A.L. Wolf. Design and Evaluation of a Wide-Area Notification Service. *ACM Transactions on Computer Systems*, 3(19):332–383, Aug 2001.
- [11] 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.

- [12] T. Catarci and M. Lenzerini, "Representing and Using Interschema Knowledge in Cooperative Information Systems," *Journal of Intelligent and Cooperative Information Systems*, vol. 2, no. 4, 1993.
- [13] U. Dayal, M. Hsu, and R. Ladin, "Business Process Coordination: State of the Art, Trends and Open Issues," in *Proceedings of the 27th Very Large Databases Conference (VLDB 2001)*, Roma, Italy, 2001.
- [14] 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.
- [15] P.Th. Eugster, P. Felber, R. Guerraoui and A.M. Kermarrec, The Many Faces of Publish/Subscribe. Technical Report ID:2000104, EPFL, DSC, Jan 2001.
- [16] P. Eugster, S. Handurukande, R. Guerraoui, A. Kermarrec, and P. Kouznetsov. Lightweight Probabilistic Broadcast. In *Proceedings of The International Conference on Dependable Systems and Networks (DSN 2001)*, July 2001.
- [17] F. Fabret, F. Llirbat, J. Pereira, and D. Shasha. Efficient matching for content-based publish/subscribe systems. Technical report, INRIA, 2000.
- [18] 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.
- [19] R. E. Gruber, B. Krishnamurthy, and E. Panagosf. The architecture of the READY event notification service. In *Proceedings of The International Conference on Distributed Computing Systems, Workshop on Middleware*, Austin, Texas, 1999.
- [20] M. Jarke, M. Lenzerini, Y. Vassiliou, and Panos Vassiliadis, Eds., *Fundamentals of Data Warehouses*, Springer Verlag, 1999.
- [21] H.B. Kon, R.Y. Wang 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.
- [22] M. Mecella, M. Scannapieco, A. Virgillito, R. Baldoni, T. Catarci, and C. Batini, "Architectural Support for Data Quality in Cooperative Information Systems," Technical report of the DaQuinCIS project, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Roma, Italy, 2002.
- [23] M. Mecella, M. Scannapieco, A. Virgillito, R. Baldoni, T. Catarci, and C. Batini, "Managing Data Quality in Cooperative Information Systems," in *Proceedings of the 10th International Conference on Cooperative Information Systems*, Irvine, CA, 2002.
- [24] 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.
- [25] B. Oki, M. Pfluegel, A. Siegel, and D. Skeen. The Information Bus - An Architecture for Extensive Distributed Systems. In *Proceedings of the 1993 ACM Symposium on Operating Systems Principles*, December 1993.
- [26] R. Piantoni and C. Stancescu. Implementing the Swiss Exchange Trading System. In *Proc. of the 27rd IEEE International Symposium on Fault-Tolerant Computing*, June 1997.
- [27] R. Preotiu-Pietro, J. Pereira, F. Llirbat, F. Fabret, K. Ross, and D. Shasha. Publish/Subscribe on the Web at Extreme Speed. In *Proc. of ACM SIGMOD Conf. on Management of Data*, Cairo, Egypt, 2000.
- [28] T.C. Redman, *Data Quality for the Information Age*, Artech House, 1996.
- [29] A. I. T. Rowstron, A. Kermarrec, M. Castro, and P. Druschel. SCRIBE: The design of a large-scale event notification infrastructure. In *Networked Group Communication*, pages 30–43, 2001.
- [30] M. Scannapieco, "Data Quality in Cooperative Information Systems," Doctoral Poster at the 28th Very Large Databases Conference (VLDB 2002), Hong Kong, 2002.
- [31] M. Scannapieco, V. Mirabella, M. Mecella, and C. Batini, "Data Quality in e-Business," in *Proceedings of the Workshop on Web Services, e-Business, and the Semantic Web: Foundations, Models, Architecture, Engineering and Applications, in conjunction with CAiSE 2002*, Toronto, Ontario, Canada, 2002.
- [32] B. Segall, D. Arnold, J. Boot, M. Henderson, and T. Phelps. Content Based Routing with Elvin4. In *Proceedings of AUUG2K, Canberra, Australia*, June 2000.
- [33] Softwired Inc. `ibus//messagebus`. Web site - <http://www.softwired-inc.com>, 2002.
- [34] J.D. Ullman. Information Integration using Logical Views. In *ICDT-97*, 1997.