A FLEXIBLE AND GENERIC DATA QUALITY METAMODEL

(Research-in-Progress)

David Becker
The MITRE Corporation
dbecker@mitre.org

William McMullen
The MITRE Corporation
mcmullen@mitre.org

Kevin Hetherington-Young
The MITRE Corporation
khetherington@mitre.org

Abstract: DQ metadata can be stored in a Metadata Repository (MDR). The structure of the MDR should be carefully defined to ensure a maximum amount of flexibility, generality and ease of use. It should also implement the major aspects of DQ management that are represented in the architecture of DQ. The basic DQ Metamodel should first provide representation of information products and data objects. It should then model DQ metrics, DQ measurements, DQ Requirements, DQ assessments and finally DQ actions. Extensions to the DQ metamodel may be needed to cover topics such as DQ aggregation, user feedback and confidence metrics, pedigree tracking, parameterized metric definitions, and enterprise data validation. There are a number of interesting applications of the DQ metamodel in areas such as decision support dashboards and service oriented architectures. There also a number of other future areas of investigation for potential incorporation into the DQ metamodel.

Key Words: Data Quality, Information Quality, Metadata, Metamodel

INTRODUCTION

Metadata is commonly defined as "data about data". A piece of metadata provides descriptive information about a particular piece of data or a collection of data. There are many different types of descriptive information that can be measured or assigned: name, subject, size, format, category, author, source, date/time stamps (creation, update, access, etc.), location, classification level, authorization, etc. The type of descriptive information of interest here has to do with the data's accuracy, precision/certainty, completeness/brevity, timeliness, consistency, lineage/pedigree, and reliability. These types of descriptive information together constitute an entire category of metadata called data quality (DQ). DQ information is clearly a type of metadata.

DQ metadata can be generated from many different sources (manual annotation, community and marketplace voting systems, data profiling tools, DQ measurement tools, ETL systems, application generated error reports and log files, human generated deficiency reports, data sampling and analysis, internal controls and security audits, etc.). These sources all help to reveal information about the quality of the information products (IPs) and data objects that are their subjects. Once the DQ metadata has been

generated, it can be used for a number of different purposes (data cleansing, operations control, financial reporting, business intelligence, analytics, decision support, data improvement initiatives, business process reengineering, six-sigma activities, business alerts and notifications, etc.).

Frequently the DQ metadata generation source is tightly coupled to its immediate use. Unfortunately this tight coupling of source and use, while providing immediate convenience and efficiencies to the involved areas, does not lend itself to reuse. DQ information that could greatly improve the operations of many other consumers of a particular set of IPs is not readily available because it is locked up inside the tightly coupled source/use area. To address this situation, the DQ metadata must be exposed. In this way the information can be made available for any consumer to access. And furthermore, to ensure maximum availability, the exposure of the metadata must span variable time periods during which the different potential consumers might need access. These objectives are most readily accomplished by storing the DQ metadata in a persistent data store. A database for storing metadata is typically called a metadata repository (MDR). A metadata repository (also called a metadata catalog) is simply a standard database that contains metadata. So, a database for storing DQ metadata is called a DQ metadata repository.

A database typically has a design that specifies the structure of the database, its tables and their relationships. This specification is called a data model. The data model for a metadata repository is called a metadata model or metamodel. A metamodel for a DQ metadata repository is called a DQ metamodel.

Modeling tools can be used to represent a metamodel at a high conceptual level, a more specific logical level, and a lower detail physical level. Many of these modeling tools can export the physical model in Data Description Language (DDL) SQL language format, which can in turn be used to automatically construct the tables in the DQ metadata repository.

An organization will normally generate and use many different types of IPs. These different types of IPs will comprise different data objects each of which may have different DQ dimensions. The DQ metamodel must be flexible enough to accommodate the various DQ metadata associated with these many different IPs, yet generic and simple enough to provide an overall common structure to support as many different potential usage scenarios as easily as possible.

For over a year, a MITRE team has been working on a Mission Oriented Investigation and Experimentation (MOIE) research project for their AF Electronic Systems Command (ESC) sponsor that deals with data quality. The title of the MOIE is “Improving Trustworthiness of Enterprise Data for Decision Making”. To date, the MOIE project has resulted in a number of specific learnings, one of which is the DQ metamodel described in this paper. This DQ metamodel is specifically intended to address the objectives of generality, flexibility and ease of implementation and use. The DQ metamodel had to be generic enough to support the large number of situations in which DQ information could be used. It had to be flexible enough to accommodate the many different DQ characteristics of interest to the widest variety of information producers and consumers. Finally, it had to be relatively simple and easy to implement and use to facilitate its ready acceptance and adoption.

THE ARCHITECTURE OF DATA QUALITY

There is a generally applicable architecture for DQ that identifies the major operational components and their relationships to one another. This architecture serves as the basis for understanding and describing most of the key architectural concepts that must be properly considered and addressed. The definition, capture and use of data quality information as part of an overall systems architecture is depicted in Figure 1.

There are several specific objectives to be served by this architecture. The first is to highlight that data quality processing is a side activity to the primary information flow of the information manufacturing system within the enterprise. The second is to show that data quality information about a specific IP can be used in a number of different areas, and, as such, must be captured and managed externally from the data quality measurement application in a reusable form. Because data quality information is a type of metadata, the place where it is stored is called a metadata repository. The third specific objective of the architecture is to represent and support the situation where the same data asset can be used by multiple consuming applications. And furthermore, each of these consuming applications might have different requirements for the quality of the data.

This systems architecture has been used to evaluate various case study DQ implementations to determine how much of the architecture they have implemented, and how well. It has also been used to guide the construction of various prototypes that attempt to build out the case study architectures with missing or inadequate architectural components.

The different components to the systems architecture are organized into four areas as follows:

- Information Manufacturing System
- Data Management Tools & Quality Aware Processing
- Metadata Repository
- DQ Metamodel / Ontology

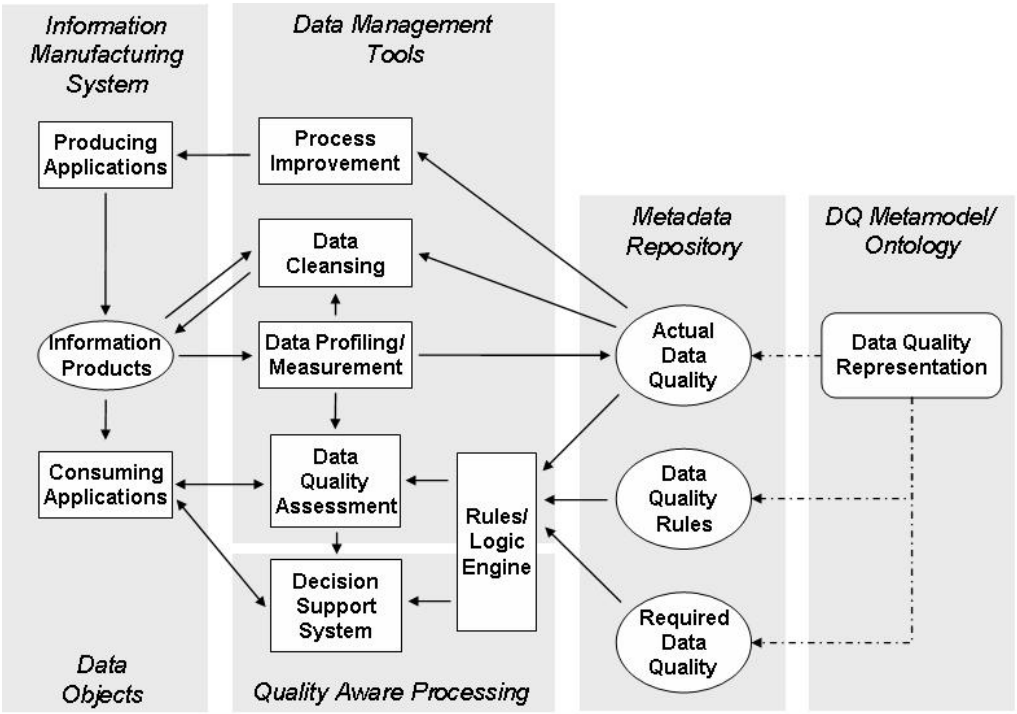


Figure 1 – The Architecture of Data Quality

Information Manufacturing Systems – The foundations of data quality are rooted in the IPs and the producing and consuming applications of the information manufacturing systems within which the IPs exist. The IPs and the data that comprises them are characterized as data objects for which DQ will be measured and assessed. A key observation is that any activity associated with the measurement and assessment of data quality is a side activity to the primary information flow – i.e., it is overhead, and thus,

considerations for how much it costs (time and resources) and how much is acceptable must also be factored into the business case for each DQ project.

Data Management Tools – The starting point for DQ is data profiling and data quality measurement. These functions are provided by a maturing collection of commercial off-the-shelf (COTS) DQ software tools. These tools originally emerged primarily in support of quickly and efficiently loading data into data warehouses as well as supporting feeds into Enterprise Resource Planning (ERP) and Customer Relationship (CRM) systems. However, the profiling and measurement functions of these tools are often tightly coupled to the DQ assessment and data cleansing functions, and frequently the DQ information they collect is maintained in a proprietary and difficult to access format.

Metadata Repository – As mentioned above, metadata is data about data. DQ metadata is data about IPs and data objects. So, DQ management is fundamentally a metadata management problem. As such a metadata repository (MDR) must be a central component of the architecture. And, the data profiling and data quality measurement tools must be able to expose the DQ information they are collecting and then use it to populate the MDR. If DQ information is externally available through an MDR, it can be used for many different purposes such as driving data cleansing, data operations management, continuous process improvement / Six Sigma initiatives, data improvement initiatives, as well as to better inform the decision support and decision making activities of an organization.

DQ Ontology/Metamodel – The structure of the metadata in the MDR is defined by a conceptual model of the metadata, i.e., a metamodel. This metamodel, which consists of the DQ entities and their relationships, essentially constitutes an ontology of data quality. Modeling tools can be used to represent a physical model of the metamodel, and can actually export the physical model in Data Description Language (DDL) format, which can in turn be used to automatically construct the tables in the MDR database.

In this paper we discuss the development and structure of the DQ metamodel.

RELATED WORK

A number of efforts have been undertaken to provide metadata models as standards that have incorporated DQ into their structural definition, or can do so with extensions and modifications. Because data quality is a type of metadata, a primary criterion for consideration of these efforts is whether they have addressed the task of defining a separate structure for DQ metadata. Other criteria are how well they model the overall architecture of data quality and support goals of generality, flexibility and ease of use.

One important generic metamodel to be considered is the Common Warehouse Metamodel (CWM) [7]. CWM is an interoperability standard of the Object Management Group (OMG) that defines a common language and interchange mechanism for metadata in the data warehousing and business analysis domains. CWM provides a set of very generic and detailed modeling concepts. It is attractive for our project primarily because it is the foundation for a tool which is currently being used to implement Extract, Transform and Load (ETL) capabilities and Metadata Repository capabilities within the GCSS-AF Data Services data warehousing environment. Unfortunately, the CWM was too large, cumbersome and abstract to be useful for our immediate project purposes. There were also a host of political and contractual issues when attempting to explore using it within the context of GCSS-AF Data Services.

A second metamodel effort is part of the European Union Data Warehouse Quality (DWQ) project. The DWQ project has as its aim the establishment of a foundation for data warehouse quality through the linking of semantic models of data warehouse architecture to explicit models of data quality. Its intent is

to develop logic-based knowledge representation and reasoning techniques to control accuracy, consistency and completeness via advanced computational modeling techniques for source integration, data reconciliation, and multi-dimensional aggregation. While this initially looked very promising, critical information needed to to make progress was ultimately inaccessible.

A number of DQ software vendors also construct a database to capture data they generate during their processing activity. However, for the most part these vendors were hold their metamodels close as proprietary product, and will not share. In general DQ metadata is available for extraction from these databases in the form of SQL driven reports, but databases are not amenable to general population or access outside the tightly coupled confines of the DQ software suite, or through a controlled list of other vendors with whom the DQ vendors have established a supported interface relationship.

Another major metamodel effort is the Universal Meta Data Model [10] developed and promoted by David Marco and Michael Jennings as part their proposed Managed Metadata Environment (MME). MME establishes a way for corporations and government entities to provide a systematic enterprise solution for metadata management. The book that describes the metamodel comes with a CD-ROM which contains the metamodel in Erwin file format for analysis and modification. It contains data models specifically targeted at areas of interest for DQ: the generic structure of data, data quality management, data profiling, data movement and transformation associated with lineage, software and systems environments, deployments and management associated with pedigree, data stewardship, and business rules management.

While the Universal Meta Data Model provides a good starting point for exploring the capture, storage manipulation and use of DQ metadata, in many areas it provides much more definition than needed for simple DQ management, and in other areas it is incomplete. For example, it provides a three level hierarchy of data definition involving data elements, data groups and data packages. This approach does not adequately provide for the multidimensional ways that data will be measured and assessed for DQ purposes. Instead, a single generic and dynamically defined “data object” construct has proved better suited for these purposes. Nor does the Universal Meta Data Model provide for the classification of DQ metrics according to the generally desired quality classification schemes that different organizations might want to employ. It is also very difficult in many cases to distinguish between definitional classes and operational classes.

Data Quality Taxonomies

Another important area of knowledge to incorporate into the DQ metamodel has to do with data quality ontologies and taxonomies. These ontologies and taxonomies should identify the key data quality characteristics that are considered most important not only in industry, but also to our primary sponsor, the AF Operations Support (OS) community. These data quality characteristics should permit focus on automated capture, while taking advantage of COTS tools to automate the capture as much as possible.

A number of individuals and organizations have explored this area, including the seminal Wang & Strong study [18]**Error! Reference source not found.**, related to the identification and categorization of different data quality dimensions. Ballou [4] and Shankaranarayanan [15] investigated the application of manufacturing system concepts to the management of data quality in IPs. Pipino and Lee’s work [13] explored design patterns for defining data quality metrics and their use in performing data quality assessments. Significant work has also been conducted by Widom [6] on the Stanford TRIO project related to uncertainty and lineage. Other studies included work by Taylor [16] on error analysis, precision and error propagation, Stonebreaker [20] on lineage in very large data collections, Ballou [2] and Jensen [9] on timeliness, Toone [17] on completeness as part of trust models for distributed information systems, Kaomea [8] and Ballou [3] on data quality valuation and usage in methodologies for allocating resources

to data quality projects. Additional works on data quality by Redman [14], Olson [12], Wang [19], and Adelman also provide significant insight in this area. Relevant practical applications of data quality work have been performed at the US EPA and the US and Canadian Census Bureaus related to accuracy and completeness. Finally Simons' has provided an excellent survey of taxonomies of uncertainty, including reviews of work by Smithson, Smets and Bonissone and Tong.

This project has used the 80/20 rule to focus our metamodeling efforts on the set of characteristics that are most prevalent concerns to our sponsor base in the AF Operations Support community, while trying to ensure that we are addressing those entities and relationships that are most amenable to automated generation and processing.

THE BASIC DQ METAMODEL

The basic DQ metamodel created for this project is summarized at a high level as a conceptual data model in Figure 2. This model presents the semantics of DQ management, i.e., the core set of entities and relationships that should be constructed and populated to properly support the fundamentals of DQ processing.

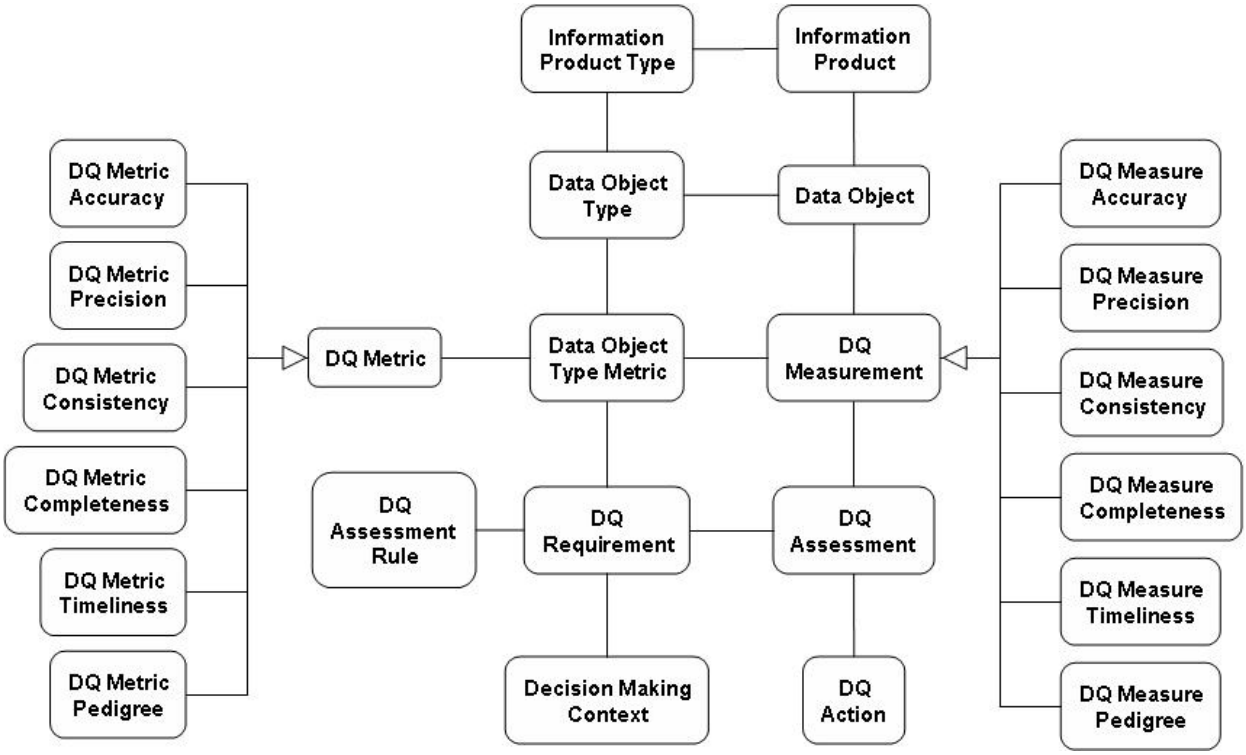


Figure 2 – The Basic DQ Metamodel

These entities are addressed in six major areas: 1) IPs and data objects, 2) DQ metrics, 3) DQ measurements, 4) DQ requirements, 5) DQ assessments, and 6) DQ actions. Each of these areas is detailed below with some examples of tables that would be constructed from the DQ metamodel.

Types Versus Instances

Another key idea implemented as part of this metamodel is the inclusion of one set of entities that

represent abstract concepts called types, and another set of entities that represent actual instances of the types. Those entities that are types, the left side of Figure 2, are generally definitional in nature and would be specified during set up of the system. Those entities which represent instances, the right side of Figure 2, are operational in nature and will be derived from their corresponding types. The operational tables will be populated and used during processing of data through the information manufacturing system during production.

Sample Baseline Test Data Set

A simplified version of the information manufacturing system for our sample baseline test data is depicted in Figure 3. The data in the sample tables below has been drawn from a baseline test data set of invoice transaction records that were collected over a two month period. Each invoice transaction record represents the basic invoicing information related to an invoicing activity from a parts supplier at one of three separate maintenance, repair and overhaul (MRO) facilities ("OO", "OC" and "WR"). Each invoice is either received electronically from external sources, or is manually entered into the system. Individual invoice transaction records are grouped into files that are processed as a batch on a daily basis at the 3 separate locations.

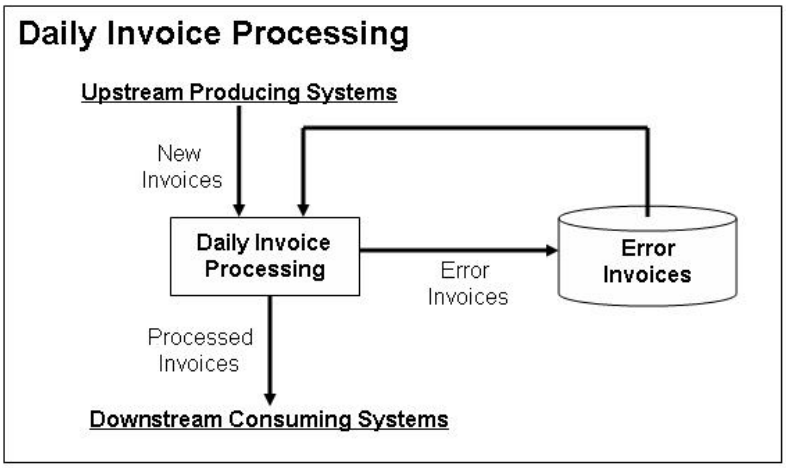


Figure 3 - Sample Baseline Test Data

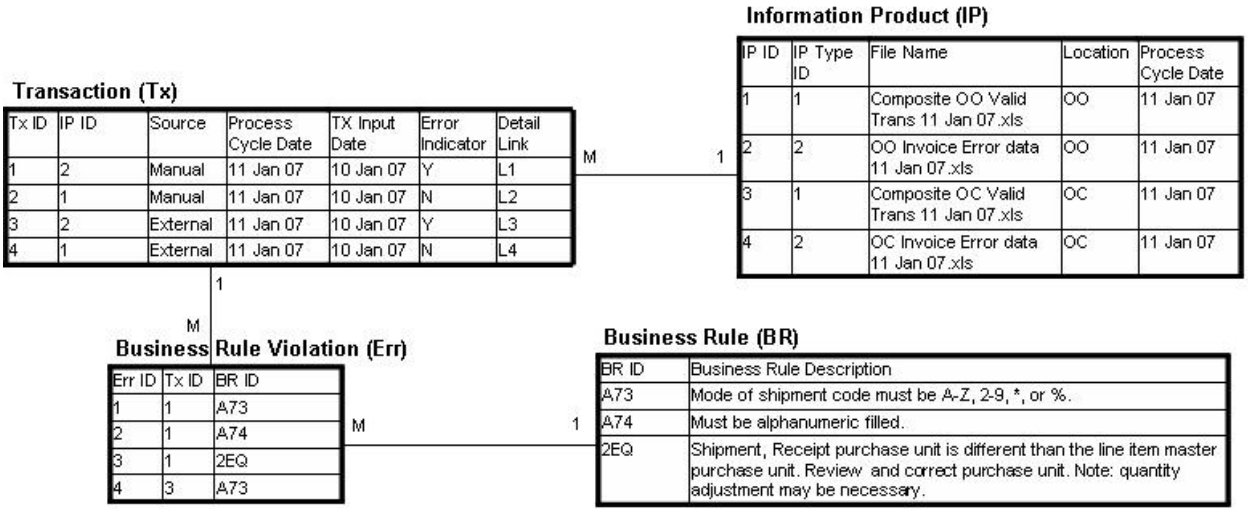
The data quality monitoring point has been set to evaluate the output of the daily invoice processing activity, and the information products generated out of this activity. There are a number of information processing and data quality management problems associated with this set of data. So, it is an ideal candidate to use in our research.

Information Products and Data Objects

The first problem that has to be addressed is how to represent the data, both as information products (IPs) and data objects. An IP represents how data is physically packaged for movement through an information manufacturing system. It is the data items that comprise the IP and the quality dimensions of each data item that provide the foundations for data quality measurements. The IP must therefore be identified in terms of the data items used to compose it. On the other hand, a data object is any set of data for which quality measurements and assessments will be made. Data objects are different than IPs. This is because collections of data in whose quality consumers have an interest are usually different than the way data is packaged for purposes of efficient operations. Data objects will typically be built up from the IPs' basic data items. So, the IPs need to be decomposed into their component data items, and the data items will then be used to build up the data objects.

The IPs for our baseline test data set are depicted in Figure 4. There are two types of IPs: a file of valid transactions, and a file of error transactions. A pair of these files will be received from each of the MRO facilities. The component data items for these IPs are the individual invoice transactions. While the actual invoice transactions themselves will typically be stored in some other shared information space, metadata about each of the constituent transactions that will be needed in subsequent DQ processing activities will be stored in the Transaction metadata table. The IP Type table is not shown, but could conceivably be a link to almost any sort of descriptive information related to the files: file naming conventions, schemas or record layouts, timing/scheduling info, producing or consuming system information, storage locations, transmission/communication protocols and parameters, etc.

Transactions can be in error when they violate certain business rules. These business rule violations are the primary contributors to many types of data quality problems. An extremely valuable but typically underused source of DQ information is the error reports and records which are generated by almost all systems and which can be transformed into DQ measurements if properly captured. In Figure 4 we show how the business rules driving the error report line items might be represented in a Business Rule table, and related to individual error transactions in the Transaction table through entries in the Business Rule Violation table.



IP ID	IP Type ID	File Name	Location	Process Cycle Date
1	1	Composite OO Valid Trans 11 Jan 07.xls	OO	11 Jan 07
2	2	OO Invoice Error data 11 Jan 07.xls	OO	11 Jan 07
3	1	Composite OC Valid Trans 11 Jan 07.xls	OC	11 Jan 07
4	2	OC Invoice Error data 11 Jan 07.xls	OC	11 Jan 07

Tx ID	IP ID	Source	Process Cycle Date	TX Input Date	Error Indicator	Detail Link
1	2	Manual	11 Jan 07	10 Jan 07	Y	L1
2	1	Manual	11 Jan 07	10 Jan 07	N	L2
3	2	External	11 Jan 07	10 Jan 07	Y	L3
4	1	External	11 Jan 07	10 Jan 07	N	L4

Err ID	Tx ID	BR ID
1	1	A73
2	1	A74
3	1	2EQ
4	3	A73

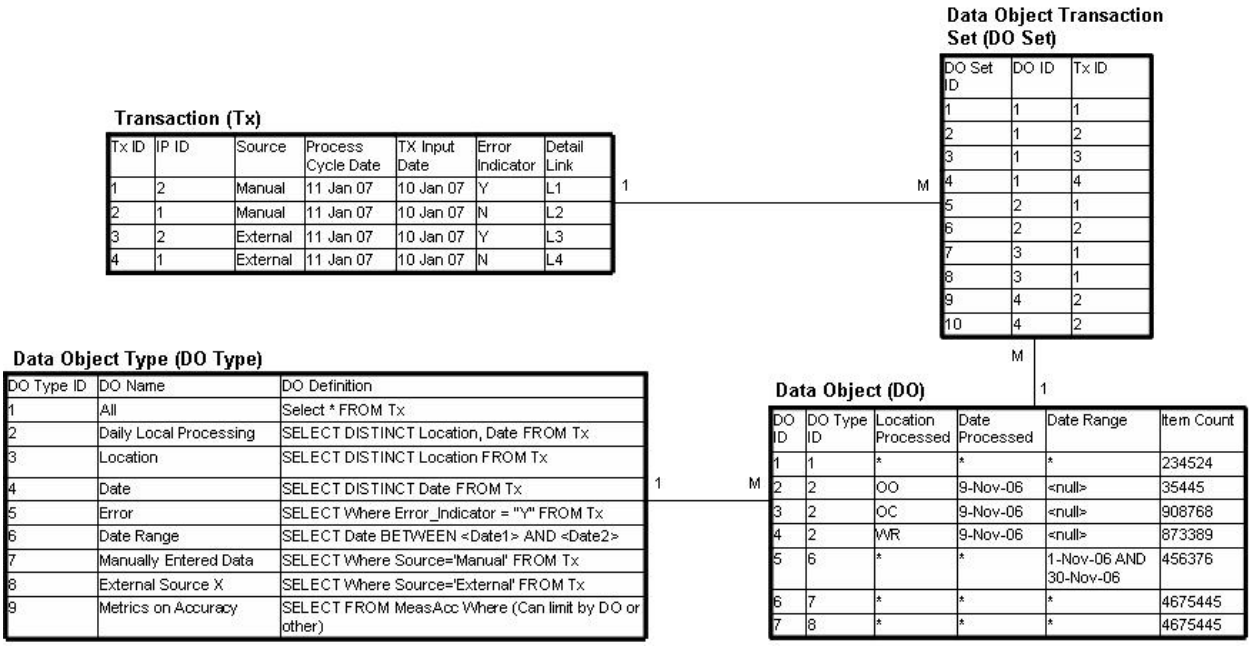
BR ID	Business Rule Description
A73	Mode of shipment code must be A-Z, 2-9, *, or %.
A74	Must be alphanumeric filled.
2EQ	Shipment, Receipt purchase unit is different than the line item master purchase unit. Review and correct purchase unit. Note: quantity adjustment may be necessary.

Figure 4 – Information Products & Transactions

The data objects for our baseline test data set are depicted in Figure 5. The data objects are built up from the transaction data items. There are many ways that we might conceive of slicing the transaction data in order to look at different dimensions of the DQ problem. For example we might want to measure the data quality of all transactions, regardless of location or data. Or we might want to look at the data quality of transactions for a given date from a given location or from all locations across a particular data range. To do this we would simply define other types of data objects. To accommodate this, we have included a Data Object Type table where SQL definitions of the different types of data objects are defined. The DO Definition field provides SQL-like descriptions of how the data object would be constructed from the Transaction table.

Data objects can also be higher level constructs. The last entry in the DO Type table is doing a selection from the accuracy measurement table to permit a consistency measurement to be performed among the accuracy calculations from different periods or locations to identify inconsistencies or trends based on mean and standard deviation calculations.

Because a particular transaction may now be included in many different data objects, and a data object may have many different transactions, we have established the conditions for a many-to-many relationship. To handle this we have introduced an association table called Data Object Transaction Set.



Transaction (Tx)

Tx ID	IP ID	Source	Process Cycle Date	TX Input Date	Error Indicator	Detail Link
1	2	Manual	11 Jan 07	10 Jan 07	Y	L1
2	1	Manual	11 Jan 07	10 Jan 07	N	L2
3	2	External	11 Jan 07	10 Jan 07	Y	L3
4	1	External	11 Jan 07	10 Jan 07	N	L4

1 — M

Data Object Transaction Set (DO Set)

DO Set ID	DO ID	Tx ID
1	1	1
2	1	2
3	1	3
4	1	4
5	2	1
6	2	2
7	3	1
8	3	1
9	4	2
10	4	2

M — 1

Data Object Type (DO Type)

DO Type ID	DO Name	DO Definition
1	All	Select * FROM Tx
2	Daily Local Processing	SELECT DISTINCT Location, Date FROM Tx
3	Location	SELECT DISTINCT Location FROM Tx
4	Date	SELECT DISTINCT Date FROM Tx
5	Error	SELECT Where Error_Indicator = "Y" FROM Tx
6	Date Range	SELECT Date BETWEEN <Date1> AND <Date2>
7	Manually Entered Data	SELECT Where Source='Manual' FROM Tx
8	External Source X	SELECT Where Source='External' FROM Tx
9	Metrics on Accuracy	SELECT FROM MeasAcc Where (Can limit by DO or other)

1 — M

Data Object (DO)

DO ID	DO Type ID	Location Processed	Date Processed	Date Range	Item Count
1	1	*	*	*	234524
2	2	OO	9-Nov-06	<null>	35445
3	2	OC	9-Nov-06	<null>	908768
4	2	WR	9-Nov-06	<null>	873389
5	6	*	*	1-Nov-06 AND 30-Nov-06	456376
6	7	*	*	*	4675445
7	8	*	*	*	4675445

Figure 5 – Data Objects

DQ Metrics

The next problem that has to be addressed is how to define the DQ metrics and relate them to the data objects we have identified. A Data Quality Metric is generally defined as a quantifiable audit criterion for the measurement and assessment of a given data object. There are six major types of data quality (see Table 1) we have focused on as part of Basic DQ Metamodel that appear to address the majority of DQ issues encountered by the AF Operations Support community. There are potentially many more dimensions that have been proposed and could be considered, but these six seemed most relevant for our purposes. Once these basic six are accommodated, other types of DQ can be introduced as needed.

Accuracy:	Degree to which the reported information value is in conformance with the true or accepted value
Precision/Certainty:	Exactness or confidence in value (vs. imprecise, uncertain, approximate, probabilistic, or fuzzy)
Consistency/Validity:	Degree of freedom from variation or contradiction Degree of satisfaction of constraints (including format/syntax/structure)
Completeness/Brevity:	Degree to which values are present in the attributes that require them Degree to which values not needed for decision making are excluded
Timeliness:	Degree to which specified data values are up to date
Lineage/Pedigree:	History of data origin (also called lineage or provenance) and subsequent transformation

Table 1 - Types of Data Quality

Figure 6 shows how the data quality of a type of data object will be calculated. A DQ metric must be specified for each type of data object that is determined to be of interest. The metric must be categorized

as one of the 6 types of data quality discussed above. All metrics are specified in the DQ Metric table with the metric calculation formula details for each type of DQ metric specified as a form of business rule in the individual DQ Metric <Subtype> tables. For example, accuracy is specified as a percentage calculation of transactions with a specific error type in the DQ Metric Accuracy table, whereas, timeliness is specified as a percentage calculation of transactions whose processing or delivery times don't fall within specific day allowances. Note that there can be multiple DQ metrics defined for each DQ type.

Consistency is interesting because it seems you cannot measure the consistency of a single data object. Instead, you must always have a comparison among several different data objects. I.e., is this data object consistent with similar (same?) data from another source? Or, is it consistent over time (with historical data from the same source)? Consistency usually involves the calculation of a mean and a standard deviation.

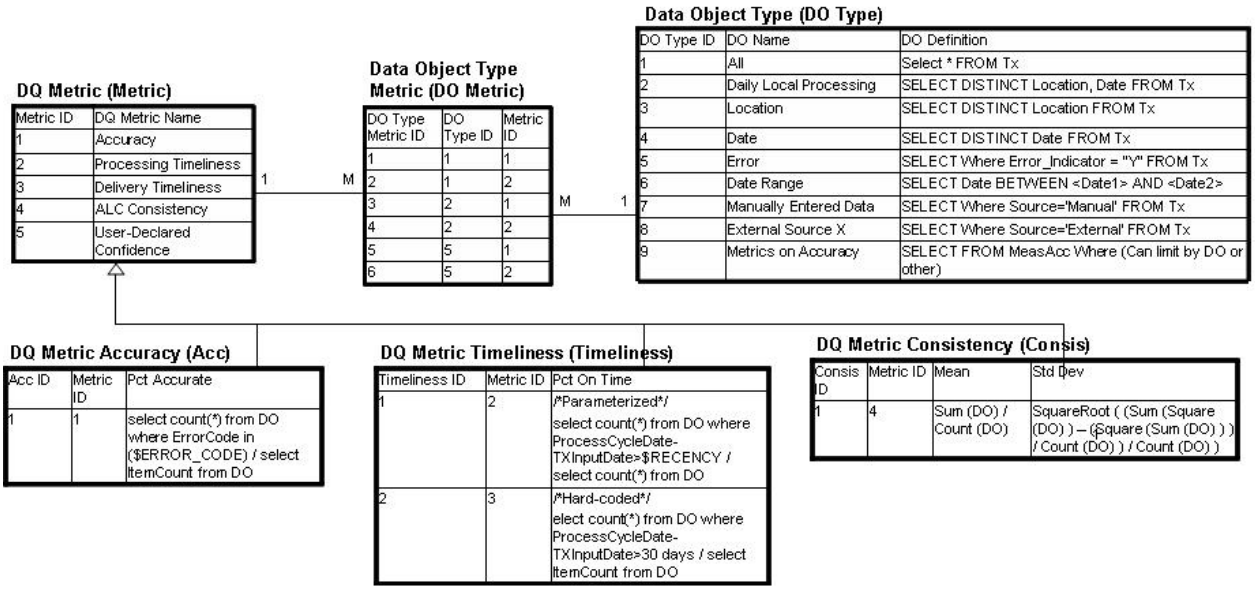


Figure 4 – DQ Metrics

Because a particular metric may now be used for many different types of data object, and because a type of data object may need to be measured by several different DQ metrics, we have again established the conditions for a many-to-many relationship. To handle this we have introduced an association table called Data Object Type Metric. This table will come in very handy later on.

DQ Measurements

Once the data objects are defined and the DQ metrics for those types of data objects are defined, you are ready to begin capturing measurements of the data quality for individual instances of the data objects. A DQ measurement is a value that is the result of applying a DQ Metric to a given data object. How we handle this is depicted in Figure 7.

Each data object instance will have an entry created in the Data Object table. A measurement will be created in the DQ Measurement table for each data object type metric in the Data Object Type Metric table. The individual measurement values will be stored in DQ Measure <Subtype> tables that correspond to the different DQ types whose definitions were included in the DQ Metric <Subtype> tables.

Because the same metric may be measured multiple times (say at different times) on the same data object,

the relationship between Data Object Type Metric and DQ Measurement may be one-to-many. This allows for periodic measurement or sampling.

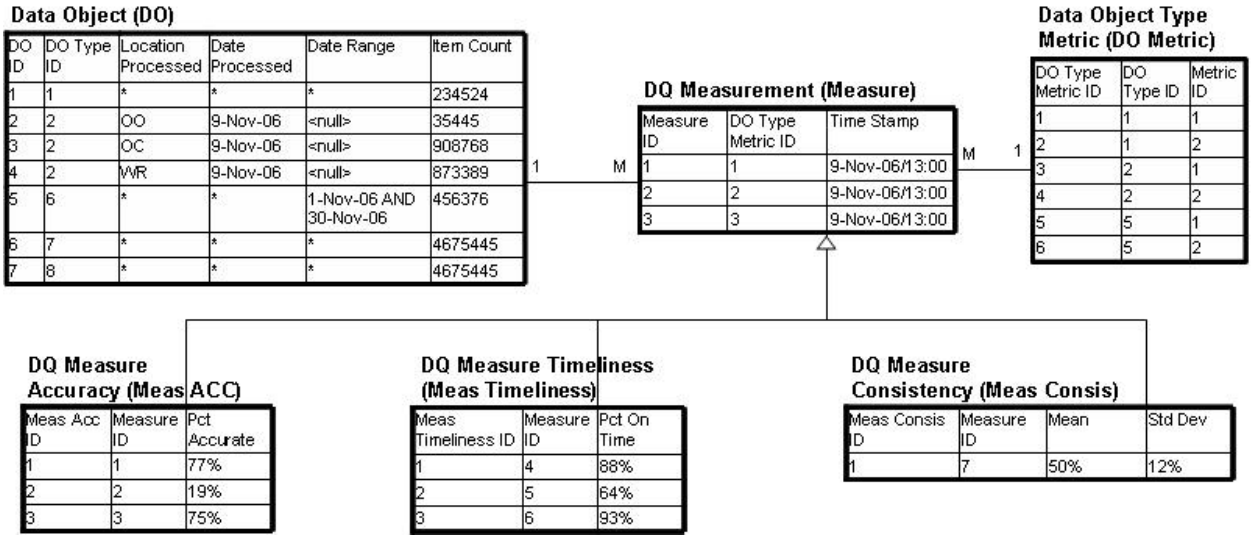


Figure 5 – DQ Measurements

DQ Requirements

A data quality requirement specifies the level that a DQ metric should have for a data object in a given decision making context. It typically provides a set of thresholds that permit an assessment of the acceptability of the quality of the data for an end user. How this is handled is depicted in Figure 8. In this way we will be able to accommodate different users or decision making contexts that have different criteria for assessing the data quality levels that are appropriate for their particular situations.

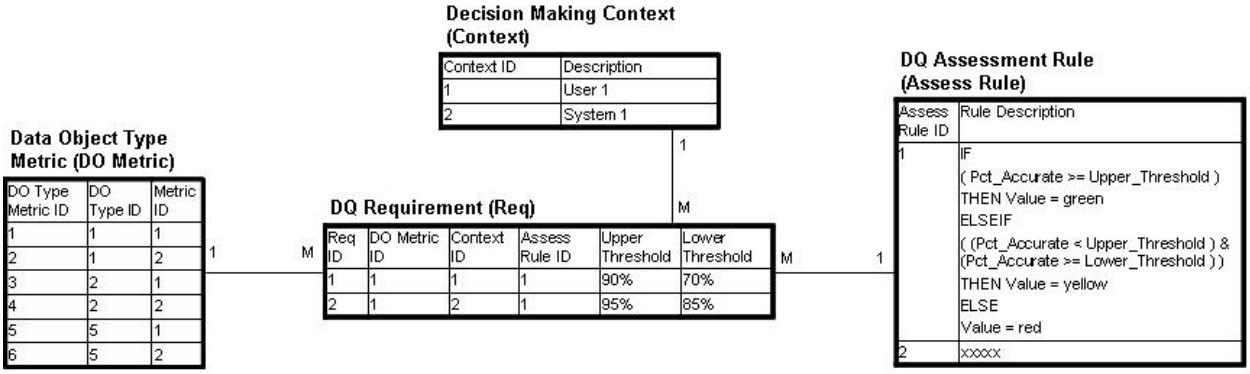


Figure 6 – DQ Requirements

DQ Assessments

Given that data quality measurements have been generated, and data quality requirements have been specified, we are ready to apply our assessment criteria. A DQ Assessment is a value that is the result of applying a data quality assessment rule that compares a DQ Measurement to a DQ Requirement as depicted in Figure 9.

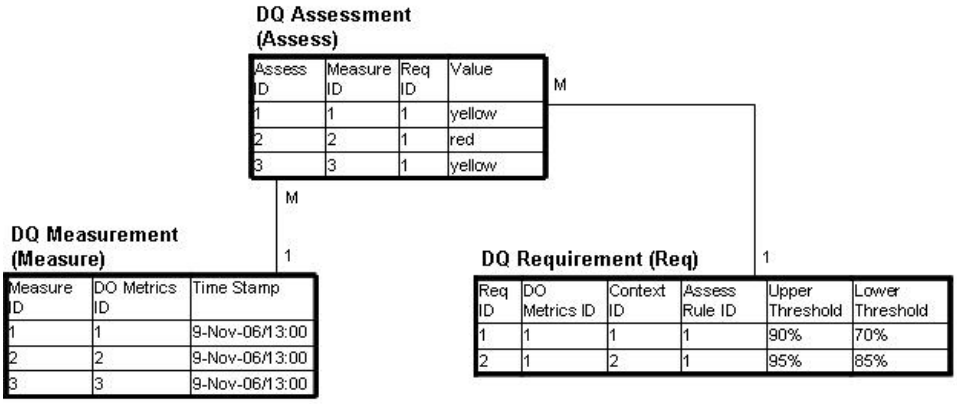


Figure 7 – DQ Assessments

One of the important concepts of the Architecture of DQ is the separation of actual DQ from required DQ. Actual data quality consists of measurements of those DQ metric characteristics of data objects that are intrinsic to the data objects. Whereas, required data quality consists of those DQ measurement levels/thresholds that are deemed necessary for specific usage or decision making contexts, and thus, are intrinsic to the user. A DQ measurement of a data object that is perfectly acceptable for one user might not be adequate for another user's purposes. Different users have different DQ requirements. DQ assessments then consist of comparisons of the DQ measurements for a given data object to the DQ requirements for that data object for each decision making context.

DQ Actions

Once a DQ assessment has been performed, appropriate actions can then be undertaken. These will be accomplished by triggering events or raising event flags that can be recognized as part of an orchestrated business process or work flow instance, and processed by a process/workflow management engine.

Many different types of actions are possible depending entirely on the needs of the operating environments in which the quality of the data is an important consideration. These could include analytics, business intelligence and support decision, operations monitoring and data management, and continuous process improvement through lean or six sigma methodologies.

IMPLEMENTATION OF THE DQ METAMODEL

The method we used to build our MDR was to model it in the ProVision architecture modeling tool as a UML class diagram - minus the methods/behaviors. We created both a logical data model and then a physical data model. Provision has the ability to export the physical model in Data Description Language (DDL) SQL format. The DDL can then be used by most DBMSs to automatically generate a database. Careful attention must be paid to naming conventions to ensure that the generated DDL will actually operate correctly, particularly with regard to relationships and foreign keys. We have generated MDRs in both DB2 and MySQL.

EXTENSIONS TO THE BASIC DQ METAMODEL

Our current work involves exploring a series of refinements and extensions to the basic DQ Metamodel. In particular we are looking at 5 different areas of interest:

Data Quality Aggregation – Data Quality Aggregation has to do with techniques to combine separate data quality assessments across different data sources, or across several different DQ dimensions for the same data object. Aggregates of any or all of the six basic DQ types for a given data object can be generated typically as a weighted sum. Data Quality Aggregation allows us to look at the big picture regarding data quality. By aggregating data quality measures across data objects we can track how data quality varies across such things as time and source. Our metadata model stores and tracks data quality measures allowing us to connect to OLAP-like reporting interfaces to better track quality and compare across different data objects, time, data lineage, etc.

User Feedback & Confidence Metrics – The User Confidence Metric is unique among the DQ metrics in that it is user-driven. The concept is to solicit users to indicate their level of confidence in a particular data object. The confidence metric, therefore, is really a user-feedback scheme. Users that consume certain data objects with low-confidence despite other high-quality indicators may indicate a problem with those data objects that the system has otherwise yet to reveal. Likewise, high-confidence community indicators mapped with otherwise low- data quality indicators may indicate that the users are either unaware of a data quality problem or that that investing in higher-quality data for those particular data objects would have a low return on investment. Since this is user-driven, we can also look at how the confidence metrics vary by decision-making context. The point of this is really to get at the level of acceptability of imperfect data.

Pedigree Tracking – Pedigree tracking is a little different in that we're explicitly tracking data objects' relationships with particular upstream data producers and production systems. The intention is to allow us to track any number of upstream systems or business processes that touch the data in any way from data production/manufacture, to cleansing, filtering, copying, vetting, and transformation. Path analysis of the resulting tree structure enables creation of a specific pedigree metric that provides a percentage of the data object derived from trustable or authoritative sources. This tree structure can also enable other measurement and assessment types such as source expiration & deprecation, source conflict and corroboration analysis, impact analysis, what-if and forensic analyses, and feedback analysis.

Parameterized Metric Definitions - Parameterization allows us to more easily manage and re-use our metric definitions. Each Metric definition can contain multiple parameters. Calling functions do not need to know how to handle these parameters.

Enterprise Data Validation – There is a major class of data quality problems that arise when attempts are made to integrate or interoperate multiple data objects or IPs generated by different systems in an enterprise context. Often, records from individual systems cannot be properly matched up, and large portions of the data are either rejected, or loaded with major voids in their keys and reference structures. It is also possible that records may be incorrectly matched due to an improper mapping of the relevant semantics of the underlying data. These integration or interoperation data quality problems only become apparent on a larger, enterprise scale. However, they should be measured, assessed and properly managed just like any other more localized DQ problem [11].

CONCLUSION & FUTURE WORK

A formally defined DQ metamodel can be valuable for ensuring that DQ management techniques adequately accommodate the flexibility, generality and ease of use requirements of potential usage situations. In this paper we have developed a DQ metamodel to meet these objectives, as well as to address the basic objectives of an overall architecture for data quality. Such a DQ metamodel should be constructed in a particular way. It should first represent information products, then data objects, DQ metrics, DQ measurements, DQ requirements, DQ assessments, and finally DQ actions. Extensions to the

DQ metamodel can be defined to cover topics such as DQ aggregation, user feedback and confidence metrics, pedigree tracking, parameterized metric definitions, and enterprise data validation. There are a number of interesting applications of the DQ metamodel in areas such as decision support, data management, and continuous process improvement.

Work still remains to permit measurement of the quality of DQ metadata. This is a second order metadata management problem that has to do with tracking the pedigree of the DQ metadata, and identifying & dealing with false negatives and false positives in data quality measurements and assessments. Another area of investigation is to explore how the DQ metamodel can be incorporated into a Services Oriented Architecture (SOA). We are currently building an open source SOA prototype [5] to characterize the architecture of data quality and the DQ metamodel as a set of data quality services in a quality aware publish & subscribe SOA that uses orchestration models to invoke the DQ services and process activities.

REFERENCES

[1] Adelman, S., Moss, L., Abai, M., *Data Strategy*, 2005, Addison-Wesley, Upper Saddle River, NJ.

[2] Ballou, D. P., Pazer, H.L., "Designing Information Systems to Optimize the Accuracy-Timeliness Tradeoff", *Information Systems Research*, 1995, Vol. 6, No. 1, pp 51-72.

[3] Ballou, D. P., Tayi, G. K., "Enhancing Data Quality in Data Warehouse Environments", *Communications of the ACM*, January 1999, Vol. 42, No. 1, pp 73-78.

[4] Ballou, D., Wang, R., Pazer, H., Tayi, G. K., "Modeling Information Manufacturing Systems to Determine Information Product Quality", *Management Science*, April 1998, Vol. 44, No. 4, pp 462-484.

[5] Becker, D. K., "Information Quality & Service Oriented Architecture", *Proceedings of the MIT 2007 Information Quality Industry Symposium*, July 2007, Cambridge, MA, pp 592-622.

[6] Cui, Y., Widom, J., "Lineage Tracing for General Data Warehouse Transformations", *Proceedings of the 27th VLDB Conference*, 2001, Roma, Italy,.

[7] Gomes, P., Farinha, J., Trigueiros, M. J., "A Data Quality Metamodel Extension to CWM", *Proceedings of the Fourth Asia-Pacific Conference on Conceptual Modeling*, 2007, Ballarat, Australia, Vol. 67, pp 17-26.

[8] Kaomea, P.,, "Valuation of Data Quality: A Decision Analysis Approach", Working Paper *TDQM-94-09, Sloan School of Management, Massachusetts Institute of Technology*, Cambridge, MA. 1994.

[9] Li, P., Ravindran, B., Jensen, E. D., "Adaptive Time-Critical Resource Management Using Time/Utility Functions: Past, Present, and Future", *28th International Computer Applications and Software Conference*, 2004.

[10] Marco, D., Jennings, M., *Universal Meta Data Models*, 2004, Wiley Publishing, Indianapolis, IN.

[11] McMullen, B., "Enterprise Data Validation Architecture (EDVA): Fixing Data Quality for Enterprise Interoperability", *Proceedings of the MIT 2007 Information Quality Industry Symposium*, July 2007, Cambridge, MA, pp 798-822.

[12] Olsen, J. E., *Data Quality – The Accuracy Dimension*, 2003, Morgan Kaufmann Publishers, Boston, MA.

[13] Pipino, L. L., Lee, Y. W., Wang, R. Y., "Data Quality Assessment", *Communications of the*

ACM, April 2002, Vol. 45, No. 4ve, pp 211-218.

[14] Redman, T. C., *Data Quality – The Field Guide*, Digital Press, Boston, MA, 2001.

[15] Shankaranarayanan, G., Wang, R. Y., Ziad, M., "IP-MAP: Representing the Manufacture of an Information Product", *Proceedings of the 2000 Conference on Information Quality*, Cambridge, MA, 2000.

[16] Taylor, J. R., An Introduction to Error Analysis – The Study of Uncertainties in Physical Measurements, Second Edition, University Science Books, Sausalito, CA., 1997.

[17] Toone, B., Gertz, M., Devanbu, P., "Trust Mediation for Distributed Information Systems", *UC Davis Computer Science Technical Report CSE-2002-35*, November 2002.

[18] Wang, R. Y., Strong, D., "Beyond Accuracy: What Data Quality Means to Data Consumers", *Journal of Management Information Systems*, Spring 1996, Vol. 12, No. 4, pp 5-34.

[19] Wang, R. Y., Ziad, M., Lee, Y. W., *Data Quality*, 2001, Kluwer Academic Publishers, Norwell, MA.

[20] Woodruff, A., M. Stonebraker, M. ,"Supporting Fine-Grained Data Lineage in a Database Visualization Environment", *Report No. UCB/CSD-97-932*, January 1997, Computer Science Division (EECS), University of California, Berkeley, CA.