Improving Data Quality in Practice: A Case Study in the Italian Public Administration

Article *in* Distributed and Parallel Databases · March 2003

DOI: 10.1023/A:1021548024224 · Source: DBLP

6 authors, including:

Paolo Missier
Newcastle University
47 PUBLICATIONS **590** CITATIONS
SEE PROFILE

Vassilios S. Verykios
Hellenic Open University
196 PUBLICATIONS **6,009** CITATIONS
SEE PROFILE

Improving Data Quality in practice: a case study in the Italian Public Administration.

P.Missier[*], G.Lalk[*], V.Verykios[¥], F.Grillo[§], T.Lorusso[§], P.Angeletti[¶]

Abstract

Assessing and improving the quality of data stored in information systems are both important and difficult tasks. For an increasing number of companies that rely on information as one of their most important assets, enforcing high data quality levels represents a strategic investment aimed at preserving the value of those assets. For a public administration or a government, good data quality translates into good service and good relationships with the citizens. Achieving high quality standards, however, is a major tasks because of the variety of ways that errors might be introduced in a system, and the difficulty in correcting them in a systematic way. Problems with data quality tend to fall into two categories. The first category is related to inconsistency among systems such as format, syntax, and semantic inconsistencies. The second category is related to inconsistency with reality as this is exemplified by missing, obsolete and incorrect data values and outliers.

In this paper, we describe a real-life case study on assessing and improving the quality of the data in the Italian Public Administration. The domain of study is set on taxpayer's data maintained by the Italian Ministry of Finances. In this context, we provide the Administration with a quantitative reckoning of such specific problems as record duplication and address mismatch and obsolescence, suggest a set of guidelines for setting precise quality improvement goals, and illustrate analysis techniques for achieving those goals. Our guidelines emphasize the importance of data flow analysis and of the definition of measurable quality indicators. They are generic and can be used to describe a variety of data quality problems, thus representing a possible reference framework for practitioners. Finally, we investigate ways to partially automate the analysis on the causes for poor data quality.

1. Introduction.

The case study presented in this paper is set in the context of consulting work, done for the Italian Ministry of Finances, and performed in collaboration with SO.GE.I., the Italian company responsible for the entire management of taxpayers' data on behalf of the Ministry. For the purpose of this work, out of all the data maintained by the Administration for each taxpayer (individual as well as business), we are only interested in the individual's vital (name, gender, date and place of birth) and current residence information. These data have been accumulating over many years in a number of legacy databases. Various official and unofficial sources contributed to the database at different times, by inserting and updating data in a way that depended more on the intricacies of tax laws rather than on a rational design for data acquisition. The bureaucratic structure of the Administration, a central hub with many peripheral offices, is reflected in the presence of separate local networks in the offices with connections to the central systems, either as clients or as mainframe terminals. The central hub is interconnected in complex ways with a number of other administrative entities (banks, the Chamber of Commerce, public utility companies, and so forth),

[*] Applied Research, Telcordia Technologies, Morristown, NJ, USA.
[¥] Drexel University, Philadelphia, PA, USA.
[§] Italian Ministry of Finances.
[¶] SO.GE.I, Roma, Italy.

both as a consumer and as a supplier of information. Data is exchanged over mostly private networks using X.25 and other protocols, and is sent to the Central Administration through a large number of phases that include manual data entry, batch and online transaction processing.

A number of problems have been hampering the quality of the Administration's data. First, due to the nature of the administrative flows, taxpayers' address data tends to become stale and not be updated for long periods of time. This happens because it is often impractical to obtain updates from the local municipalities that maintain the official residence data. Also, errors may occur when individuals' personal data are recorded. Some of these errors are not corrected and a sizeable fraction of them can not be detected. Furthermore, addresses that are supplied through various data sources differ in formats, following local conventions that change in time and resulting in multiple versions for the same real address. Personal data also follows local conventions, with multiple spellings of the same names. Finally, many of the records currently in the database, were entered over the years using legacy processes that included one or more manual data entry steps. This usually indicates that they may not have been correctly cross-referenced to related information in other databases, resulting in integrity problems, and also that they may contain spelling and other errors.

Within this context, two specific Data Quality problems were given top priority by the Administration. The first problem was to identify the incorrect and obsolete address records by using two alternate address sources. Official and accurate address data can, in principle, be obtained from the local municipalities. However, in many cases providing access to this information requires ad hoc and expensive processing, since online transactions are not always available. Thus, the Administration first wanted to assess the actual benefits, in terms of quality improvements on the address information, of using the municipalities' data as a reference source. The second source is the address information provided by taxpayers on their tax returns. Because of the fact that until 1999 the processing time for returns has been measured in years, an address change information may be already obsolete[1] by the time it is acquired. Additionally, there is no guarantee that it is accurate and correct. Thus, our second task was to assess the reliability of the tax returns as a source of address information. Finally, the problem of assessing accuracy and consistency was complicated by the absence of a suitable, reliable database of official Italian addresses.

In order to understand the second problem, it is necessary to refer to the structure of the key used to identify taxpayers' records in the database. In Italy, taxpayers are uniquely identified using a "Codice Fiscale" (referred to as CF throughout the paper). The CF string is computed using the first and last name, gender, date and place of birth. Consider for instance Ms. Amoruso Anna, born October 10, 1968, in Bari (town code A662). Her CF is "MRSNNA68L43A662", as is read as follows. The first and second three characters are the first and second three consonants of the last and first name, respectively (vowels are used if not enough consonants are available). "68" is the year of birth, "L" encodes the month (October), "43" encodes both the day of birth and the gender: by convention, 40 is added to the birthday for females. Finally, "A662" is the code for the town of birth. Potential collisions (say, two twins with first names having the same first three consonants), which are infrequent, are dealt with by changing one of the first characters.

[1] The situation is dramatically improving with the recent introduction of Internet-based solutions for tax return submissions by individuals, slated for 2001. Electronic submission is expected to affect the majority of taxpayers, reducing the processing time to a few months.

The main difficulty with using the CF as a key is that errors in the record fields propagate to the key. Thus, when these errors are corrected, a new key must be issued, to replace the old one. In the database, the two keys are linked together, so that can both still used in practice. Problems arise when the new key is not correctly linked to the old one, but rather, an entire new record, identified only by the new key, is created for the individual whose data has been corrected. This may happen for a number of reasons that can be traced to the nature of the processes that feed the database. The net effect is that multiple records for the same individual may exist in the database, each carrying slightly different values for the fields that contribute to the computation of the CF. If this situation may sound extreme, consider that, according to the Administration's estimates, this problem affects 5% of the database, or a few million records. Thus, our second problem is to identify data analysis techniques for the automatic detection of pairs of records that represent the same individual.

Our work had two main general goals. First, we recognize that no fully automated cleaning procedure can in general provide sufficient confidence in the correct association of records to the corresponding real-world entities. Thus, our first goal was to define data analysis procedures and data cleaning techniques that minimize the number of records for which a matching decision cannot be reached with sufficient confidence, that is, the records for which a costly manual inspection is needed. The second goal was to provide some insight into the causes of poor quality, by classifying the types of errors encountered and relating them to the business and system processes that manipulate the data. We formulated a comprehensive Data Quality plan, consisting of the following steps:

- Assess the current state of the data, with respect to a number of quality dimensions – definitely precisely in Section 2.1;
- Determine precise and quantitative target Data Quality improvement goals, for various types of data items (Section 3);
- Determine the causes of poor data quality, and suggest modifications to the processes that manage (store, manipulate) the data;
- Based on repeated, periodical assessments and on the causal analysis, try to predict the future state of the data (what data becomes dirty, and how fast);
- Identify priorities (which data gives me the highest return on investment after cleaning? which has the most potential for sustaining target data quality levels in time?)

Data cleaning activities can then be designed as a part of this data quality plan, and are instrumental to the quality targets defined in the course of the analysis.

In the rest of the paper we describe in details the techniques and tools used to tackle these problems, and then present the quantitative results of our analysis (Section 5). First, however, we propose a simple model for concise data flow analysis, introduce the notions of Data Stewards and Data Users and discuss their implications on quality planning. Finally, in Section 3 we provide an informal description of quality indicators and quality targets. The paper concludes with related work on matching algorithms.

2. Data Flow and Process Analysis

A correct investigation on the causes of poor data quality hinges upon the analysis of the processes that manipulate the data. The analysis should focus on the system features that are relevant to the flow of data through a system, and it should highlight how specific process features impact the quality of the data being manipulated by the process.

There is general agreement in the area of data quality [18][17][20][21], that data cleaning without data flow analysis is only a temporary solution at best, and often a complete waste of money. This is going to be the case, unless the processes that supply poor quality data are analyzed and re-engineered in order to avoid contaminating the freshly cleaned system. Data flow analysis should be able to answer a number of relevant questions about the way data is handled by a system. For instance, suppose two data flows, F_1 and F_2, carry logically related data. In the context of our study, these could represent the acquisition of tax returns from taxpayers, and the collection of the associated balance payments. It matters to know at which point and when in the system the flows meet, that is, when related entries are matched against each other: When is a payment checked against the corresponding tax return? What system procedures are involved? What happens if either the return or the payment are not there? How late is the discrepancy discovered? If there is a long delay in return processing, then by the time the two flows meet for a given taxpayer, the address information indicated on the return (or in the main database) may no longer be current. In this case, should any discrepancy arise, it becomes difficult to even communicate with the taxpayer, let alone reconcile the accounting. How much would it cost to quickly pre-process the returns in order to provide faster problem resolution?

Of course, these considerations do assume knowledge of the specific domain under consideration. Data flow issues, however, can be stated in more generality:

- Which entity is responsible for which data? the distinction between Data Stewards and Data Users is introduced in Section 2.3;
- How is data supplied? Several sources may contribute data to a system. In general, different levels of data quality are expected, depending on the type of source (is it an individual taxpayer or a certified accountant, is data input manually, has it being copied over from system to system, etc.);
- How do duplicates propagate? Data is often duplicated either due to poor definition of the business logic (or bad bureaucratic processes, as one may view it), or for technical reasons, or simply for convenience. In many cases, the logic used to synchronize and reconcile the copies is buried in the details of obscure business processes or system procedures, and the effects of duplications are difficult to analyze;
- Are related entries cross-validated? Different pieces of data are often logically related, but that relationship is not defined explicitly in the data model. This makes it difficult to enforce integrity constraints, and therefore, to guarantee value correctness.

In the following, we first briefly review the traditional dimensions used for measuring the quality of data, and then introduce a simple abstract model for the description of relevant process features that help answer some of these questions.

2.1. Traditional Classification of DQ Dimensions

The dimensions listed here are widely used, see for instance [17]. They help describe the data quality problems in precise terms.

- *Correctness* refers both to the syntactic and semantic correctness of data, and is defined with respect to the range of values for a data type. For instance, a CF encodes information about a taxpayer and is computed according to a well-defined algorithm. In addition, it contains a checksum digit, like a credit card number. Thus, both its correctness with respect to the associated personal data, and its internal consistency can be verified. Correctness can also be defined with respect to other reference values, by means of integrity constraints. For instance, a full name may be considered incorrect if it does not appear in a related reference table. Notice that missing values for optional fields are considered correct.
- *Format* and *Value Consistency* measure the degree of homogeneity, in format and value respectively, among all of the representations of values that correspond to the same real-world entity. Format consistency indicates how closely the values for a given data type adhere to a common format. For instance, when multiple formats are used to represent the same date, format consistency is lower than when one single format is accepted. Value consistency measures the number of ways in which one single entity can be expressed. Typically, for instance, a physical place in a town can be known through completely different addresses, all semantically correct, or it can be subject to acceptable and known spelling variations. The more variations, the less consistent the value for that address.
- *Accuracy* generalizes correctness to the case where discrepancies from a set of canonical data values are measurable. For instance, the correctness of an address can sometimes be validated against a reference database of known addresses. In this case, the accuracy of an address would be defined in terms of the types and number of spelling deviations from the canonical value, and although the address value can often still be recognized, it is not considered accurate.
- *Value Completeness and Precision*: these dimensions both refer to the quantity of information expressed in a data value. For numerical data, the standard definition for numerical precision applies. More generally, though, completeness may also refer to the presence of various optional elements of a compound data value. Values for which some of the optional elements are omitted are less complete and less precise than values that carry all of those values. For instance, an address in which all parts are specified is more precise than one in which only the street name is present, although both may be valid addresses. Notice that a value can be precise (specified in full detail), but incorrect, and vice versa. In particular, for optional data fields, a missing value can still be correct.

In addition to the non-temporal dimensions mentioned above, value *currency* is the most commonly used temporal dimension. Currency is informally defined as the interval between the time at which a data value becomes available to a system, and the time that value is actually acquired by the system. This notion includes the case of real-world values that change in time. In this case, the currency is defined as the time it takes for the system to acquire the most recent value resulting from a sequence of updates.

2.2. Process Modeling

The model we propose for the description of relevant process features, consists of three domains in which events occur: (a) the real world, (b) the abstract representation of the real world as viewed by some administrative entity, and (c) the representation of the real world as viewed by an Information System (IS). Each of these three domains is described by its state. The situation is depicted in Figure 1. Real-world events, of course, change the state of the world. To keep with our Public Administration example, we are interested in events that have an effect on the Administration, in this case the Revenue Service, on the IS that maintains data about the taxpayers. In this scenario, the birth of a baby is an example of interesting event that changes the state of the real world, say from *A* to *A'*. In the Administration, paperwork is produced and a workflow is typically triggered as a consequence of that event. The effect of the workflow represents a state change in the Administration domain (the new birth is recorded officially). In turn, the workflow produces new events, often in the form of paperwork or other procedures mandated by the law, which cause a state change, say from α to α', in one or more of the databases that are part of the IS. For instance, a new CF is issued and a new entry is made into the database of (current and future) taxpayers'. This abstraction attempts to capture the essential information about the propagation of a real-world event all the way into the IS, which is where the data is ultimately to be used, and where data quality is measured.

Different quality levels for the data result from different data flows. The essential flow and quality information we want to capture, include the following:

- The relevant events, called triggers, how they originate and the time at which they occur;
- The Administrative flow triggered by the event, and the official paperwork it produces (obviously, this is domain-specific);
- Events in the Administrative domain that trigger IS procedures;
- IS procedures triggered by Administrative events (online transactions, batch jobs, system workflows), and their input data;
- State changes (at a macroscopic level) on the System as a consequence of the procedures being executed.

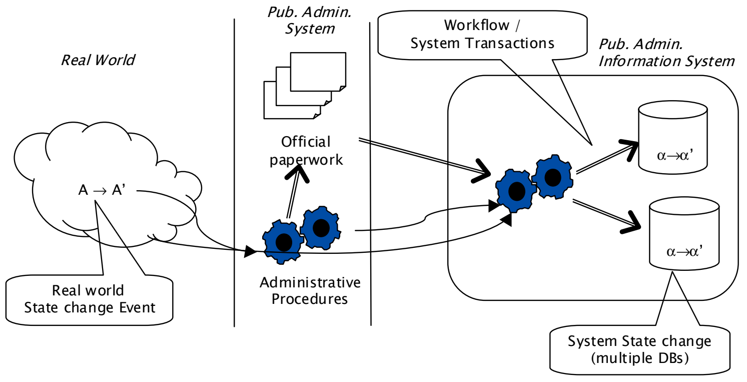
Real World
Pub. Admin. System
Workflow / System Transactions
Pub. Admin. Information System
Official paperwork
A → A'
α→α'
α→α'
Real world State change Event
Administrative Procedures
System State change (multiple DBs)

Figure 1. Model for data flow description.

Table 1 and lists the possible data flows that cause a new CF to be assigned to a citizen. In the first real-world trigger, when a new baby is born, the IRS is informed through the local municipality. In the second case, the parents inform the local IRS office directly. The third flow is triggered when a citizen shows up with an explicit request to have a CF assigned to her, and the last flow accounts for periodical batch re-alignments of local municipalities data with the IRS systems. In this case, missing or wrong CFs are assigned by the system directly, and only later are the citizens informed.

For each of the flows, Table 2 lists the input data resulting from the corresponding administrative action (the workflow itself is omitted for brevity), the system procedures triggered by those actions, and finally, the expected data quality levels, with respect to various quality dimensions, namely Correctness, Format and Value Consistency, and Temporal Currency. Quality levels can be directly related to the nature of the system procedure (online, periodic/batch, offline through paper or fax), the number of expected manual data entry steps, and the certification level of the source for the original input data (e.g., in Italy "self-certification" of personal information is commonly used).

Flow Number	Trigger	Input	System Procedure
1	Birth / registration with municipality alone	Newborn's personal data, from municipality	Online CF assignment procedure (integrated for some of the municipalities)
2	Birth / registration with municipality and with IRS office	Parents' request / self-certification	Terminal procedure from IRS office
3	Taxpayer's request to IRS office	Taxpayer's (self-certified) data with official ID	Terminal procedure from IRS office
4	Periodic re-alignment with Muni. data.	Muni batch data, ad hoc format	Ad hoc validation procedure

Table 1 Example Data Flows.

Flow Number	Expected Data Quality levels		
	Correctness	**Currency**	**Format and Value Consistency**
1	High if procedure is integrated with Muni's workflow Medium if additional copy/data entry required	Max 3 months lag, by law	High if Muni applies its own consistency rules.
2	Low if data cannot be cross-referenced with muni data (common case). 1 data entry required; High if certified by muni.	Determined by parents' delays. System process time is negligible	Low. Determined mainly by parents' consistency
3	Determined by accuracy of ID. 1 data entry required.	Determined by taxpayer (arbitrary). System process time is negligible.	Determined by ID/ muni data
4	High for validated data. Highly reliable correctness info No data entry required	Arbitrarily low. Data expected to lose alignment eventually.	High for validated data.

Table 2 Expected Quality levels for the example flows.

In practice, this simple data flow analysis proved sufficient to effectively derive data quality targets and to pinpoint simple causes for poor data quality, forming the base for the data quality plan outlined in the following sections.

2.3. Data Stewards and Data Users

For each of the relevant pieces of data (entities, records, fields) mentioned in the data flows, the appropriate Data Stewards must be identified. These are informally defined as the management, administrative and technical authorities who are responsible for the values assigned to data, and for their quality. Data Stewards serve as reference points during the lifetime of the data across various systems. For instance, as the information about a newborn is propagated through the

Administration and into the system, the authority that certifies its correctness at any point in time, i.e., the local municipality, does not change. The definition of the Data Steward for each item mentioned in the data flow plays an important role in managing different versions of the data: when responsibilities are not clearly assigned, inconsistent values may be assigned to the different versions.

Data Stewards are often defined based on the set of legislative constraints that indicate the certification authority responsible for a data item. In Italy, for instance, only the Ministry of Finances can assign a new CF or modify an existing one. While this sounds quite reasonable, the nature of the CF is such that, as we mentioned earlier, in practice it can be computed by anyone using an individual's personal data. Because the CF is required for many business and administrative transactions, but few people carry the official card in their pocket, a value for the CF is in fact often computed on the fly. The problem with using this shortcut, even assuming it is used in earnest, is that the computed CF may be *wrong*, i.e., *different from the one assigned by the Data Steward*[2]. However, because online validation or lookup of a CF is not normally available when the transaction takes place, it can be used without control. The overall result is that multiple versions may exist, for which the quality levels become difficult to assess.

Legislation helps define Data Stewards for data that flows into an Information System, but, in practice, it is not sufficient to guarantee quality levels. Keeping with the taxpayers address example, local municipalities are ultimately responsible for the correctness of the address information. However, because data exchange between the central Administration and the over 8,000 Italian municipalities is technically difficult[3], other sources are used, primarily the address indicated in the tax returns. When the returns are obtained on paper and data entry is done manually, the result is address data that is at most as current as the processing time (often measured in years), and only as correct and format-consistent as the taxpayer sees fit. In general, technical and architectural constraints limit the way in which Data Stewards can enforce their institutional authority.

To summarize, Data Stewards must *provide* for correct, accurate and current data values, as defined through a system of business or administrative rules. Their role is often assigned by the legislation or, in other data domains, as part of a business contract. Information System management must *enforce* data quality levels by making sure that data values conform to those defined by the Data Stewards.

Some comments are in order. First, the system management often faces a trade-off between the ease of access to an unofficial version of the data, versus its expected quality levels. It often makes sense for the system to supply data to outside consumers that is only approximately correct, rather than going through the pain of contacting the relevant Data Steward. Much of the business

[2] This may happen for at least two reasons. First, the information used to do the computation may itself not be official, e.g. proper official ID may not be required (as long as the transaction goes through, everyone is happy). Second, the computed CF may collide with one assigned to someone else (the coding is not guaranteed to be injective over the domain of present and future taxpayers). Without online validation, these collisions cannot be detected.

[3] Today, pilot projects are in place to enable two-way online data exchange with a number of selected municipalities. Full coverage of the country is expected in the near future.

negotiation about the cost and benefits of a data quality plan, revolves around the reality of this trade-off.

Second, there are situations in which data is *generated* by the system. For instance, let us assume that the indication of inconsistencies between the balance on a tax return and that computed by the Administration, is generated by a system procedure. Because the procedure does nothing more than implementing a business rule defined by the Administration, the system is only responsible for the correct implementation of the rule. In other words, the Data Stewards remains the administrative authority outside of the system.

Finally, notice that we assumed that a Data Steward implements its role of provider of good data, by using another Information System. Thus, we reduced the problem of enforcing quality levels to one of technical information exchange across systems, through networks and protocols. It is important to realize that, in general, each system can have both roles: as a supplier of correct data to a Steward, it must correctly manage the Steward's data; and as a user of data defined by other Stewards, it must enforce the corresponding quality levels.

In conclusion, we again emphasize the importance of this key distinction of roles. When roles are not clearly defined, it becomes very difficult to produce a quality plan in which all the entities involved understand and agree on their respective responsibilities.

2.4. Internal and External Consistency of Data

The definition of Data Stewards is relevant for data flow analysis because, as data cross system boundaries during its lifetime, the responsibility for their quality rests on different administrative entities at different stages. For instance, the System management for the Finances data may claim that it cannot be responsible for the poor levels of correctness of data that is accepted into its systems without proper certification from the municipal authorities. While our task is not to try to resolve inter-administrative disputes directly, we can help restoring the order by suggesting a further classification of data quality dimensions, to help Data Stewards and data users understand their respective roles with respect to data and processes. This "system-oriented" classification is designed to highlight the roles and responsibilities of system managers in data management.

The main distinction is between *internal* and *external consistency*. Internal consistency is traditionally defined in the context of relational databases both for individual schema entities, and across entities. On a single entity, the distinction is between consistency on a single attribute – usually referring to simple value completeness, and consistency on a set of attributes in a relation, expressed for instance by constraints such as "the value in attribute A is present if and only if attribute B (a flag) is set to true". When considered across entities, internal consistency is usually defined at different levels. At the schema level, it takes the form of inter-relational integrity constraints (cardinality, values). At the implementation level, it is defined by architectural constraints for the implementation of entities on multiple DBMS, and for entity duplicate management (constraints between duplicates).

External consistency refers to the quality of data before it enters the system: it attempts to capture the level of noise and the nature and frequency of data entry errors, and the currency of input data, enforcing a distinction depending on the "speed" at which data reaches the system.

In general, internal consistency levels are associated with the quality of System design, implementation and management, and thus its responsibility rests with system designers and architects. External consistency indicates the quality of information that flows into the system from outside sources. The responsibility for quality levels associated with external consistency rests with the managers of the systems that supply the data.

The two classifications express similar information from two different perspectives, as Table 3 shows:

		Value accuracy, precision	**Currency**	**Value Completeness**	**Format Consistence**	**Value Consistence**
Internal Consistency	**Integrity over single logical entity, single field**	Value is present and semantically correct	Info is up-to-date	Value is complete	Field or record-level format consistency	Field or record-level format consistency
	Integrity over single logical entity, entire record	(e.g. both values for two fields are required)				
	Integrity over multiple entities, ER Logical Schema	Validation of integrity constraints among entities				
	Integrity over multiple entities, Architectural (DB level)	Correct physical level representation	Timely Replica propagation		Format consistent across replicas	Correct replica propagation
External Consistency	**Input data quality ("noise")**	Correct input values	Complete input values		Format consistent over subsequent inputs	Value consistent over subsequent inputs
	Currency		Data acquisition is within time constraints			Consistent data acquisition times

Table 3. Traditional and system-oriented quality dimensions.

3. Data Quality planning: Indicators and Targets

In the previous section, we argued that data cleaning without adequate process analysis can only yield short-term benefits. By the same token, we view the data analysis and quality assessment described in this paper as a part of a comprehensive data quality plan. The plan defines quantitative quality targets, expressed in terms of precise quality indicators. Indicators are quantities that describe the quality levels of a data set with respect to the various quality dimensions, e.g. the fraction of a population of data values that is incorrect, or obsolete. For the domain of the Fiscal Administration, examples of indicators include the fraction of duplicate CFs, the fraction of missing or null values for a field, and the fraction of residence data that is known to be out-of-date, or that is *probably* obsolete, with some given probability. Thus, the dimensions defined in Section 2.1 are used to express indicators.

Data quality targets are predicates expressed using the values of the indicators, and provide precise guidelines for defining procedures that enforce data quality levels. An example target is: "the fraction of known obsolete residence data must be less than X at the end of each month". This target's one parameter, X, provides a degree of freedom to the quality planners for the definition of realistic plans. In practice, its setting often reflects business decisions about the cost/benefits trade-off of enforcing a quality plan.

3.1. Meta Attributes

In order to define significant indicators, it is convenient to introduce a small number of *meta-attributes*, i.e., attributes that describe the quality of the attributes values in the data domain. These attributes provide useful information on the history of a value, such as the time it was last updated, the source data used to update the value, the entity (system or administrative) that authorized the update, and so forth. This meta-information, together with data flow analysis, is used primarily to investigate the causes for poor data quality, but is also very important for quality monitoring. For instance, one can use it to compute derived quantities such as the average currency of a data set. More importantly, currency information can be aggregated over a number of other meta-attributes, such as the town of residence, the processing center for that segment of the taxpayers population, and so forth, so that if a particular type of systematic address error is identified, we can attempt to correlate the error with the meta-information. Finally, suppose that a new procedure is implemented to automate some of the data entry activity: the meta-information collected through monitoring makes it possible to prove quantitatively that the claimed improvement in the quality levels over a more traditional method can actually be achieved.

A core list of domain-independent meta-attributes is presented in Table 4.

Generic Meta-attribute	**Example**
Who: Data source	Consider address information. Sources can include the Central Administration, the municipality through an occasional batch job, the taxpayer through a tax return, etc.
When: Latest update date, plus history of all corrections to the data.	Date of latest update. Obviously, gives information on currency. Full history of updates also indicates pattern of update frequency.
What:: Operation performed on the data.	In addition to the usual insertion, deletion, it also includes *correction of an incorrect value*, and *validation of a value*. Notice that, by law, some incorrect data can be flagged as invalid, but not automatically corrected.
Status: Not validated, verified correct, verified incorrect, corrected.	The state indicates whether the value has been validated against the Data Steward or other source as indicated above, the result of the validation (correct/incorrect), and whether it has been corrected as a result of the validation.

Table 4. Meta-attributes.

In practice, meta-attributes values are generated by data testing and cleaning procedures, which are often performed in a spotty and occasional fashion. When a systematic data quality monitoring plan is in place, meta-values can be accumulated at regular intervals, facilitating temporal analysis of quality improvements over time. The data testing activity is best reflected by the *status* meta-value, which indicates the type and result of quality testing performed on a specific data item. Valid status values are as follows. *Validated* indicates that the data value is consistent with a reference value, and is also correct. For a record that has been tested against a reference value and has been found invalid, the status is *tested incorrect*. These values are known to be of low quality, but for a variety of reasons, they could not be fixed. *Corrected* records, on the contrary, have been tested, found invalid, and fixed.

From a system's perspective, notice that the meta-attributes reflect the typical information found in the system operations transaction logs, such as a transaction's timestamp, the system procedure used for the operation, and so forth. This suggests that a general mechanism for the design of a database of meta-information may consist in "promoting" part of the information present in the logs, at the level of the database schema.

3.2. Indicators and Targets

Both quality indicators and targets can be expressed precisely in terms of the meta-attributes. Specifically, the *accuracy* of an entire dataset with respect to a reference dataset is the fraction of validated values in the dataset, while the fraction of non-tested values provides a measure of coverage for the testing procedure itself. Likewise, the fraction of tested incorrect values gives a lower bound on the fraction of values that are actually incorrect (this is because some of the non-tested values are expected to be incorrect)[4].

Here is an example quality target: "Given a time period of [Y] days, at the end of each period [Z%] of the population must be in the tested state, and at least [Y%] of that fraction must be in the validated state". In this target, the fraction of tested population and the fraction of validated values represent the measurable indicators. Parameters X, Y, and Z provide the degrees of freedom that are necessary in practice for the various data managers to negotiate realistic levels of effort and investment. Other examples of indicators include the fraction (of a sample of defined size) of CFs that are found to be duplicated (correctness), the fraction of obsolete residence data in the taxpayers database (currency), the fraction of truncated values in a given data field (accuracy), the fraction of null values in a field (completeness), the fraction of records in which the gender value is inconsistent with the CF (consistency), and so forth. Examples of quality targets that predicate over these indicators include the following: "the fraction of obsolete residence data must not exceed X% at the end of each month", "the fraction of inconsistent gender values must not exceed X% for values stored before 12/31/1997 (some domain-specific milestone date)", etc. Additionally, aggregated indicators are often considered, for instance given a set of administrative regions identified in attribute A, with values $\{a_1,\ldots,a_n\}$, one can aggregate the indicators expressed above

[4] Given sufficient historical information on the result of testing procedures, one could produce statistics that attempt to estimate the portion of incorrect non-tested values. However, this discussion goes beyond the scope of our present work.

over A, e.g. "the fraction of duplicated CFs in region a_i, for each region", yielding a set of indicators, one for each group. Corresponding targets follow naturally.

Significant *temporal indicators* are of two main types. First, with reference to Figure 1, one can measure the delay from the time an event triggers a workflow in the administrative world, to the time the data reaches the Information System (time t2 – t1 in the figure). In our case study, this time lag represents the sum of administrative delays and processing delays (due to backlog, offline transactions, etc.). For instance, depending on the specific data flow, the real-world event "new birth" may take months to translate into actual data entry into the taxpayers database. A typical quality target over such indicator can be stated as: "Given a data sample covering at least X% of the entire dataset, the max time lapse from the day a tax return is submitted, to the day it is processed, should be no larger than [Y] days for [Z%] of the values". The second type of temporal indicator records the time necessary to detect and correct errors, using the dates associated with changes in the status of values. These numbers provide an indication of monitoring and control effectiveness over the quality of the data.

The most valuable information produced by data cleaning is perhaps the insight it provides into the causes for poor data quality. After indicators are computed, quality problems are highlighted in the form of unsatisfied targets. Although organizations are often tempted to blame failure to meet the quality targets on insufficient data cleaning efforts, relating the results to the data flows may reveal more useful explanatory information.

We conclude this section by mentioning the potential use for data mining and machine learning techniques to automate the analysis of the causes of poor data quality. A discussion on this interesting line of research, currently being pursued by this Data Quality group, goes beyond the scope of this paper.

4. Analysis and Improvement of Data Quality for the Public Administration

Armed with the common terminology and the context for data analysis presented so far, we can now describe our case study in more detail. The study focused on two main problems. The first is a classical address reconciliation problem, where (taxpayers') addresses in one database need to be normalized and then cleaned both for correctness (check correct spelling and verify that the address actually exists), and for currency (identify and possibly update obsolete addresses). In the second, a duplication analysis problem, taxpayers who are assigned more than one CF are to be identified, in those cases where the Administration is unaware of the duplication.

In this section, we first provide a summary of the main issues in the object identity problem, searching and approximate matching, and then we illustrate our approach and the tool used to tackle the problems. Details of the experiments and of their results are presented in Section 5.

4.1. Record Matching and the Object Identity Problem

Data quality improvement relies mainly upon techniques for record matching. Record matching is the process of identifying records in a database, that refer to the same real world entity or object. The decision, as to the matching status of a pair of records, is based on the comparison of common characteristics, for instance relevant parts of an address, between the particular pair of records.

These common characteristics are related to the similarities in the schema of the corresponding records. Similar information is generally represented differently in two database schemas. Despite the differences in the representation of the two schemas, however, there must be some overlapping information which can be used for identifying matches of records from different databases that refer to the same customer. In our example, address information is represented in slightly different, but overlapping ways in the three data sources used for the analysis.

There are three different types of linking or matching. The first one is called one-to-one matching and refers to the identification of a one-to-one mapping among the records of two files or databases. The second one is one-to-many, and refers to the process of matching one record from one file or database to one or more records from the second file or database. The last one, which is called many-to-many, refers to the process of matching each one of the records from both files or databases with one or more records from their corresponding files. Both our case study problems are concerned with the first type of matching.

The record linking or matching process consists of a searching step and of a matching step. In the searching step, the record linking process explores the space of record pairs in order to identify records that are prospective matches (there is high probability that the compared records are actually a match). We expect that only a very small percentage of the records that exist in a file or database will actually be a match. In the matching step, the record linking process compares the records which were identified as prospective matches during the searching phase, and assigns to them one of the following labels: link, non-link or possible link. The comparison of the two records depends on the nature of the common characteristics in the two databases.

Two main types of errors may result from the record linking process. The first error occurs when a pair of records is a match while the record matching algorithm assigns the non-link status. This type of error is called a false match or a false positive and it is known as Type I error in statistics. The second type of error occurs when a pair of records is not actually a match, but the record linking process assigns the match status to this pair. This type of error, is known as a false non-match or false negative and it is known as Type II error in statistics. These errors depend on the actual matching status of a pair of records which is usually not known in advance. In reality, two records either they match or they do not. Their actual matching status depends on the agreement of their common characteristics, not in an absolute way, but in a rather probabilistic one. For example, assume that we have two records which they agree in one characteristic while they disagree in a second one. In practice, it is usually the case that the same agreement and disagreement pattern can appear in two pairs of records in which the first is a match while the second is not.

Whenever there is not enough evidence to choose one of the decisions over the other, the record pair is assigned the "possible link" status. All the record pairs that have been assigned the possible link status, should be manually reviewed in order a final decision to be taken for the pair. Because manual investigation is recognized as the single most expensive operation in data quality analysis, a matching algorithm that assigns too records to the "possible link" class should be avoided. For this reason, by using either some cost metrics or some levels of acceptable error in the record linking process, we can explicitly control how many of the record pairs can be assigned the possible link status. In Section 5, these notions are reviewed and applied to our specific case study.

4.2. The Searching Process

The searching process must be intelligent enough to exclude from comparison record pairs that completely disagree with each other (not prospective matches). This is quite reasonable based on the fact that the matching process has a very high application cost and must be applied economically. Often times, we need to make a compromise between the number of record pairs that are compared and the accuracy of the matching process. For example, assume that we have developed a matching algorithm which is error free. Clearly, such an algorithm is very difficult to build, and we expect that its complexity will be high. Given such a matching algorithm, the only way to be sure that we have found all the matching record pairs is to apply an exhaustive pair-wise comparison of all records in both data sets. The complexity of such an approach is quadratic in the number of records in one database given that the databases have similar sizes. If we also assume that a very small percentage of the pairs of records that will be compared, will actually be matches, the result will be an extremely high process overhead. The way out of this problem is to identify only those record pairs which have a high probability of matching, leaving uninspected those pairs that look very different.

Several techniques have been developed in the past for addressing the problem of searching the space of record pairs for matching records. The first one which appeared early on in a paper by Newcombe [13] is called blocking. This technique splits the entire set of records into segments which are called blocks. Each block contains records that have the same value for a subset or part of the common characteristics. These characteristics are known as blocking variables. The idea behind blocking is to compare records from the two data sets that belong to blocks which agree on the so-called blocking variables. This decreases the overall complexity of matching but may lead to leaving some of the matching records uninspected.

A second approach for searching is very common in un-duplicating a database. This approach includes the sorting of the database, which is followed by a second step for identifying duplicate or approximately duplicate records. This approach leaves the matching process with the task of comparing only those pairs of records that lie next to each other in the final ordering. The subset of characteristics that are used for sorting the data sets is known as the sorting key. The role of the sorting key is very similar to that of the blocking variable. In this approach, it is clear that the data sets should be merged into one set, before the sorting starts.

A very similar approach to the sorting technique, which increases the completeness of the matching process, is called windowing. In this approach, the data sets are merged, and then sorted, like in the sorting approach. The only difference is that instead of comparing records that lie next to each other, we compare records that fall within a constant size window from each other. If the window size is very large, this process is very similar to the exhaustive search method. For small values of the window size, the process is very efficient and gives better results than the sorting method.

Because of the many errors that exist in the data sets that are compared, it is very common that the information selected for blocking or sorting the data sets contain errors. If this happens, then some records are clustered or sorted far way from those records that they should be compared with. For this reason, a multi-pass approach can be used for increasing the completeness of the matching process. In this approach, a number of different blocking variables, or sorting keys, can be used. The combined data set is scanned as many times as the number of different keys. Results from independent passes are combined to give the final set of matching records. An extension of the multi-pass approach has been implemented by these authors. In the multi-pass approach, the transitive closure of the results of independent passes are combined. For example, if one record matches with a second one in one pass, and the second one matches with a third one in the second pass, then the first and the third records are also compared to each other. The same solution has been proposed independently by other researchers who made use of an algorithmic technique which identifies the connected components of a graph. The connected components represent the record clusters in the record matching process. Both groups of researchers presented very similar results.

4.3. The Matching Process

The matching process entails making a correct decision about the matching status of a pair of records by comparing their characteristics. Usually, the characteristics to be compared include strings of characters, and for this reason string comparison algorithms play an important role in the record matching process. Some kinds of character strings, like names, exhibit a certain error behavior, and for this reason, some coding schemes have been devised that are able to detect and neutralize these errors. These coding schemes, mostly phonetic in nature, like the Soundex code, can be used to extract some components from the name string that are very prone to errors, so as the string matching process will be more robust and reliable. The same approach can be used for blocking as well. String matching algorithms like the edit distance or the Jaro algorithm [10], or the *n*-grams, have been used in different research and experimental record matching models or systems giving very promising and accurate results.

The simplest way to solve the record matching problem is to build a linkage rule by using a string matching algorithm. The algorithm considers the entire record as a string with or without blank characters and decides upon the matching status based on the distance of the strings corresponding to the records under comparison. This approach for solving the record matching problem has been called field matching and its performance depends primarily on the selection of a smart string matching algorithm that accounts for different types of errors. In our case study, it turned out that string matching algorithms, independently applied to different pairs of fields in the records, were sufficiently accurate to provide a satisfactory analysis. In Section 5.3.1, we review some of the more sophisticated matching techniques known in the literature.

4.4. Analysis Tool

The prototype data quality analysis toolkit used for this case study is designed around the idea of pushing the data through a pipeline of pre-defined processing blocks, which can be quickly and cheaply connected together to solve a specific analysis problem. Figure 2. Although the toolkit comes with basic general-purpose processing blocks, new, user-defined and domain-specific blocks can be easily added, providing a very flexible and extensible analysis environment.

Typical functional blocks include pre-processing, filtering, and pairwise approximate matching. Pre-processing and filtering includes the standard elimination of stopwords, special characters, and blanks, and the reduction of known, common words to a canonical form. This processing is usually highly domain dependent. For instance, in the context of Italian street addresses, the notation “vle” is expanded to “Viale” (“avenue”), “p.zza” to “Piazza”, and so forth. Approximate matching may encapsulate a variety of edit distance functions. The specific definitions used in the case study are described in Section 5.2.

User-defined blocks can be designed from scratch, or they can be derived from existing blocks, using standard object-oriented programming techniques. Processing blocks can accept and produce multiple data streams. Thus, a pre-processing block would be a 1-input, 1-output block, while a pairwise approximate matching would use two data sources, and produce one data stream consisting of the matching pairs. Blocks can also perform more complex tasks, such as clustering records from one single input stream into a number of classes, according to some domain-specific criteria, e.g. the value of an enumerated type, or a threshold value. One such block would have one single input and produce one output stream for each record class identified by the clustering algorithm.

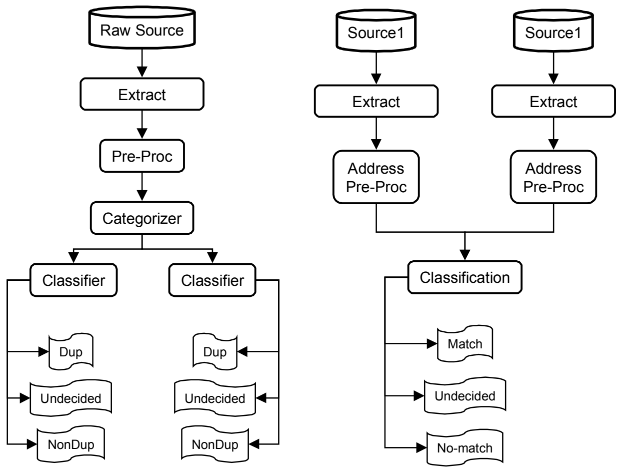

Figure 2. Data processing flows.

Figure 2 shows how several types of blocks can be connected into a data flow, using the multiple i/o feature. The flow on the left side contains a categorizer block, which partitions the input data stream into two categories, based on some exclusive condition. Each of the two categories is then processed independently in order to assign each record to one of three possible classes: “duplicate”, “non-duplicate”, and “Undecided”. The flow on the right illustrates the typical matching process in which records from two sources are pairwise compared, and record pairs are assigned to the “match”, “non-match” and “undecided” classes.

5. Results from the Address Matching and Duplicate Detection

5.1. Data Analysis Goals and Classification Techniques

Both the address matching and the duplicate problems were tackled using the same general technique, namely pairwise approximate record matching with fine-tuning of the matching algorithm. In addition, in the spirit of systematic monitoring of data quality, pairs of reconciled records are classified according to the type of mismatch encountered, in order to provide a map of the known problems and of their relevance. Both address matching and duplicate detection are instances of the more general supervised classification problem: given a set of labels, a classification algorithm, or classifier, assigns one the labels to each data item in a dataset, according to some measure of fitting. In practice, labels often represent real-world categories, or “classes”. In the address matching problem, given two data sources for address data, the dataset consists of its cartesian product. Each pair of addresses is to be assigned by the classifier to one of only three classes: “matching”, “non-matching” and “undecided”. In the second problem, the dataset consists of the cartesian product of the set of taxpayers records with itself, and the ideal outcome of the classification for each pair of records is “duplicate”/”non-duplicate”.

In this context, a classifier is defined as a collection of processing blocks organized into a data flow, or a “pipeline”, as described in the previous section. The decision logic is encapsulated in the individual blocks. For instance, one block may perform pairwise matching between two records, based on some definition of the edit distance between some of the records’ fields, while the next block makes the actual classification decision based on a threshold value for the computed distance. Thus, parameters in this classification scheme include a proper choice of edit distance function, as well as a choice of threshold value.

The accuracy of the classifier depends critically on the choice of the training set used to fine-tune its processing blocks. A training set is a limited-size sample from the full dataset, with the property that the labels associated to its data items are known. Not all datasets come with a nice training set, however. While in the duplicates problem for this case study the training set was given, in the case of address matching a *surrogate* training set had to be constructed. A surrogate consists of a random sample from the dataset, which is simply analyzed in a semi-manual fashion, with the objective of adjusting the parameters of the classifier until they fit the training set data. The most widely used measure of accuracy is the number of misclassifications produced by the classifier when a test dataset is processed, often presented in the form of a confusion matrix[5]. When only two classes exist, this notion reduces to the fraction of *false positives* and *false negatives* produced by the classifier, i.e., items incorrectly classified as a match and non-match, respectively. The practical consequence of a large number of misclassification, e.g. in our duplicates study, is that the gravity of the problem is overestimated or underestimated, respectively.

The approach followed in our study has been to conservatively accept some fraction of false positives while minimizing the number of false negatives (because we did not want to miss any of the potential problems). In practice, after tuning the distance functions, we fine-tuned the decision threshold using the training set, so that we could estimate the fraction of *expected* false positives, as well as provide an upper bound for it. We then introduced one additional class of “probably matching” record pairs, to account for the uncertainty around the region of expected false positives (a neighborhood around the threshold). The rationale for this setup is that manual investigation of individual records is the single most expensive step in data quality analysis, hence it makes sense to accurately identify the (hopefully small) class of record pairs for which a sound classification decision could not be reached.

[5] For each pair of classes A and B, a cell (A,B) in the confusion matrix contains the number of samples from the dataset that belong to class A but were instead incorrectly assigned to class B.

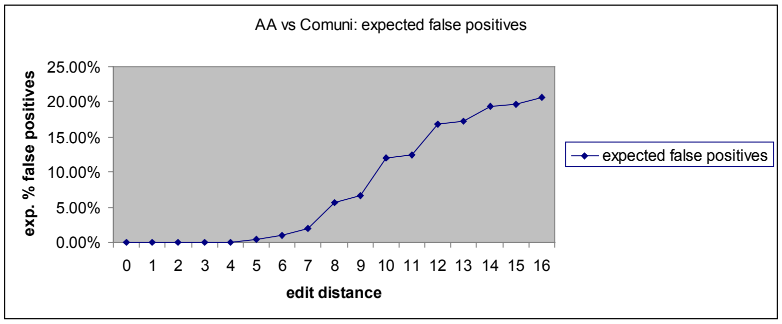

Figure 3. False positives vs. edit distance values.

The choice of training set in the accuracy of the classification is known to be critical [1][9][15]. We only mention one specific trade-off between using a natural (when available) and a surrogate training set: in the former, there is no control over the sampling phase, and hence no guarantee that the sample reflects the characteristics of the population. In fact, in our study this was not the case, because the training set was composed of records that had been previously identified as potentially problematic. Thus, the distribution of the *types* of problems in the training set is not indicative of the distribution of problem types in the overall population. However, this training set carries perfect class labeling. Conversely, a surrogate can be randomly sampled, but then the accuracy of the labeling is limited by the available domain expertise – point queries are often necessary to decide on difficult cases. In our work, we decided to create a surrogate in the address matching problem, and to use the existing training set in the duplicates problem.

5.2. Address Matching

The data for the address matching analysis includes three sets, each consisting of 275,000 taxpayers' records (name, CF and address only). The size was mainly determined by the availability of records from the municipalities, which formed the first of two reference sets. As we mentioned in Section 2.3, municipal administrations are the Data Stewards for the current residence data. Because the majority of those administrations are still unable to provide their data or to validate the Central Administration's data on a periodic basis, the availability of such recent addresses was our most stringent constraint for data collection. The second source of address information, used by the Administration to update its own address information using tax returns, once a year, is the taxpayer's own indication on the tax return. This source can therefore be used as a good reference to derive quality information about address currency. The final dataset, of course, is the one under analysis, namely the address data in the main taxpayers' database. Data selection was based on the CF alone, which is used as a primary key[6].

Two analyses were conducted, matching the target set against the two references sources separately, in an attempt to determine which of those sources is more reliable and would be more effective for improving the quality of addresses. In each of the two analyses, the first task has been to generate a random sample of only 500 records, to be used as the surrogate training set. The choice of size was obviously dictated by the resources available to do manual classification. After

[6] Selection relied on the system's own selection procedures, so that in the case of multiple or replaced CFs, the proper and official search mechanism would be used.

using a pre-processing block for address normalization specialized for specific Italian-style addresses, the approximate matching processing block was tuned. A number of algorithms for computing edit distance between two addresses were tested and compared with a baseline definition (minimum number of inserts and deletes required to map one entire address string to the other). We settled on the group edit distance algorithm, which compares strings based on evaluation of the matching state of individual words within the string, regardless of order. An exact match is assigned a value of "0" and a mismatch is assigned a value of "1". For example, Table 5 shows the comparison of the strings "F.ROSSITTO 6" to "FELICIANO ROSSITTO 6 SCALA D".

String 1	String 2	Distance
F	FELICIANO	2
ROSSITTO	ROSSITTO	0
6	6	0
SCALA		1
D	6	1
Group distance		4

Table 5. Group Distance.

Next, we determined a threshold for a binary decision (match/no-match) based on the number of false positives in the training set. As Figure 3 shows, using edit group distance ≤5 achieves the best matches/false positives ratio. Specifically, the set of record pairs that are within an e.d. of 5, called the experimental matching set, represent 74.8% of the training set. The question is then, what is the expected number of false positives if the decision rule simply accepts as matches all and only the elements of the matching set? As it turns out, manual inspection reveals that 74.4% of the matching set is actually correct, yielding an expected 0.4% rate for false positives. As for the false negatives, one again finds that 78.8% of the entire training set is actually a match. Because the classifier will only match 74.4% of the training set, the expected ratio of undetected matches (false negatives) is 4.4%. Figure 4 details the false positives for various distance values.

Application of the tuned classifier to the entire data set yields matching ratios for various e.d. values that are very close to those obtained on the training set, as Figure 5 shows. This allowed us to conclude that the surrogate training set was actually a representative set, and that the overall confidence in the classifier was ±2%. In the end, we only included the records with e.d. equal to 5 and 6 in the "probable match" class of records to be further investigated.

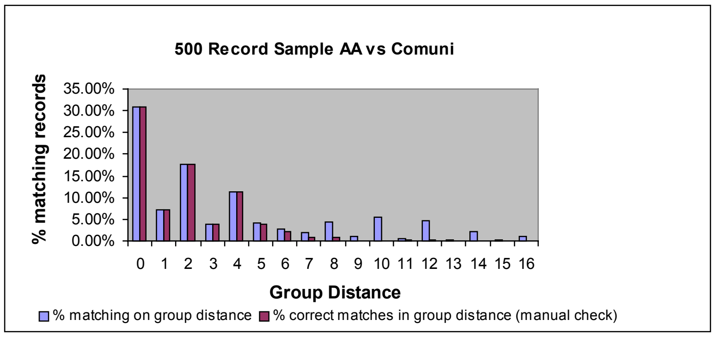

Figure 4. False positives by distance values.

The second analysis, using the taxpayers returns as reference data source, leveraged the tuning performed on the first. In fact, as it turned out, applying the same "edit distance 5" rule yielded a similarly small proportion of false positives on a 10,000 test sample. Furthermore, the proportion of matches within distance 5 on the test set and on the full dataset are remarkably close (90.29% and 89.93%, respectively), indicating that the rule was equally applicable at different scales on a homogeneous data set.

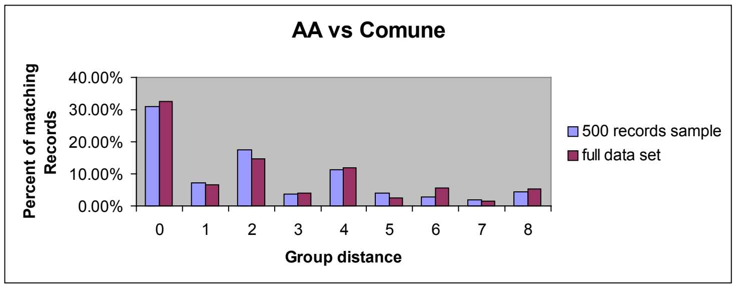

Figure 5. Matching distributions -- training and full set.

A more detailed analysis of the types of mismatches among the approximate matching record pairs, omitted here for brevity, revealed expected differences in the spelling of many street addresses. Following are a few examples of the types of mismatches that were correctly classified as matches (remember that we did not make use of specific domain information, e.g. a reference street atlas): "TRAVERSA IOVINE 18" matches "TRAV. IOVINO 18", "U.DELLA RAGGIOLA 3" matches "UGUCCIONE DELLA FAGGIUOLA 3.", "S.SCRIBANO 1" matches "G NNA SORTINO SCRIBANO 1", and "F ROSSITTO 6" matches "FELICIANO ROSSITTO 6 SCALA D". Most of the unresolved mismatches, however, refer to completely different addresses. This analysis thus provided useful insight into the low currency of address data in the main database: in the context of the framework we have adopted, it provided values for our currency quality indicators.

5.3. Duplicate Analysis

The second analysis consists, as mentioned in previous sections, of detecting the pairs of records that identify the same taxpayer using two slightly different CFs. Records consist of a CF and of the fields used by the Administration to compute the CF, namely first and last name, gender, date and place of birth. To illustrate the encoding used, consider the CF "MRSNNA68L43A662" assigned to Ms. Amoruso Anna. The first and second three characters are the first and second three consonants of the last and first name, respectively (vowels are used if not enough consonants are available). "68" is the year of birth, "L" encodes the month (October), "43" encodes both the day of birth and the gender: by convention, 40 is added to the birthday for females (thus, Anna is a female, born on 10/3/68). Finally, "A662" is the code for the town of birth.

The size of the full dataset is 600,000 records, representing a sample over the entire database of about 70M records. Sampling was based on the selection of a few municipalities, of various sizes, in which a particularly high rate of duplication problems was either known from previous studies, or suspected (based on the presence of very popular and common last names that are shared among a disproportionate fraction of the local citizens). For each such municipality, all citizens known to the Administration were selected, to maximize the chance that both elements of each duplicate pair were included[7]. A final consideration was the availability of training set elements for the chosen areas. Unlike the previous case study, for this analysis a training set of 13,100 known duplicate pairs was given. These duplicates had been identified over the years, either in the course of ad hoc cleaning initiatives, or because external events triggered the appropriate administrative correction flows.

Because the duplication problem manifests itself in the form of slightly different CFs, we first tried to correlate the types of mismatches observed between pairs of sub-fields of the CFs, with the edit distances observed between the fields in the records. Using the processing blocks and the tool described in Section 4.4, we assigned records pairs to categories based on the type of mismatch between individual CF sub-fields. For instance, one class would contain all pairs that differ *only* in the gender sub-field of the CF (all other sub-fields being identical), another would contain all pairs for which *only* the first name sub-fields are at most *n* characters apart, and so forth. Thus, in each class, every pair has identical CF, except for exactly one of the sub-fields.

Next, it is natural to try to use this information to answer questions like: " how likely is it to have two distinct individuals for which only the gender information changes in the CF?". We started by computing the distribution of the types of mismatches for each of the categories, using different edit distance functions on the rest of the fields, for each pair of records. For instance, for the set of pairs that only differ in the gender sub-field, the distribution of mismatches on the first and last names is listed in Table 6:

Category		Number of CF pairs	
		Training Set (known duplicates)	Full Dataset (classified as duplicates)
Exact match in first name and last name, after removing punctuation marks		1660 (84%)	263 (68%)
Exact match in first name and last name, after removing punctuation marks, but middle name missing in one of the two records		20 (1%)	8 (2%)
Exact match in last name only after removing punctuation marks	Typo in first name	21 (1%)	3 (1%)
	Male versus Female name and vice-versa in the first name[8]	258 (13%)	106 (27%)
Exact match in first name only after removing punctuation marks, ed = 1 in last name		28 (1%)	9 (2%)

Table 6 - Distribution of mismatches for the "wrong gender" category.

The specific sub-categories used to characterize one particular main category are determined heuristically. For instance, the "mismatching year-of-birth" category includes one specific sub-

[7] To be more accurate, one would factor in people's mobility rate in Italy, which is about 10% annually – no local statistics were available.

[8] EG "Mario" vs. "Maria", "Francesco" vs "Francesca", etc.

category for the common case where the two digits that form the year are swapped. The distribution information can now be used to determine whether two records in the full dataset are duplicated. Similarly to the address matching problem, analyzing this data amounts to inducing a classifier for which only three classes are of interest, namely "duplicate", "non-duplicate", and "unknown". Furthermore, the classifier should produce a small "unknown" class – the class of pairs that would have to be manually analyzed.

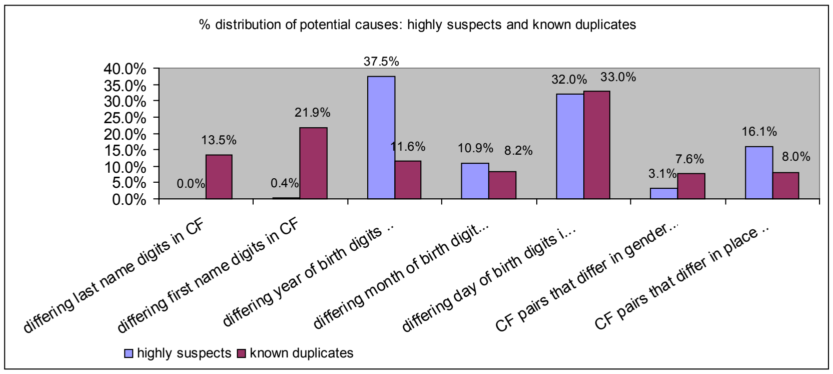

Figure 6. Distribution of causes for suspect duplicates.

The definition of the classifier is heuristic, and is based on the observation that, for the training set, the distributions can be interpreted as rules of the form: "if a pair belongs to category C (say, gender sub-field mismatch) and to sub-category SC (both first and last names match exactly), then the pairs are duplicates". One such rule can be defined for each category/sub-category pair. In fact, these rules already provide a simple decision criteria for each pair in the full dataset. Clearly, however, not all the rules are equally supported by the data. One way to select the most relevant rules is to follow the Data Mining approach of association rules, and to characterize each rule by its confidence and its support[9]. Rules could then be ranked and only the "strongest" rules would be adopted for the decision. However, in reality this approach proved difficult to automate, because of the observed mismatch between the distribution of sub-categories in the training set, and that in the full dataset. As it turned out, our data violated one implicit assumption behind the association rules approach, namely that rules enjoy a similar support in the training and in the full dataset. These discrepancies are illustrated in Figure 6 as well in Table 6 for the gender category only (last column).

In practice, the classification rules were determined manually, based on the direct observation of the distribution, but without too much regard for the objective support values. In the end, the results were consistent with the expectations of the data owners: in addition to the known 4.4% of duplicates over the 600,000 records dataset, 1.5% records were classified as duplicates, while only .3% were undecided.

[9] Confidence is a measure of statstical significance for the rule. In this case, it can be expressed as the number of pairs in the TS that either are in C, or satisfy the condition for SC (both first and last names match). Support is a measure of strength for a rule. In this case, it equals |SC|.

5.3.1. Related Work

Estimating and/or improving the quality [21] [22] of data stored in real life databases are difficult problems because of the data volume and the variety of ways errors might be introduced in a system. Consequently, heuristic ad-hoc solutions are often sought to balance the computational load and the predictive accuracy.

The areas of approximate matching has been investigated in some detail. The first general solution to the matching problem was a probabilistic one. An optimal linkage rule was developed by Fellegi and Sunter [5] based on the matching and non-matching probabilities of the comparison results and on the basic observation that some comparison patterns had different matching and non-matching probabilities. By dividing these probabilities for each pattern and sorting the ratios, they generated a linear ordering of these agreement and disagreement patterns. The next issue was to identify appropriate threshold values for the ratio, so as by looking at a ratio value the rule could designate a pair as a link, or a non-link or a possible link. The authors computed these thresholds by using the probabilities of errors, and bounding them by the error levels specified by the user. The optimality of the proposed linkage rule relies on the fact that it minimizes the number of record pairs that are assigned a possible link status. The reasoning behind this selection is that pairs designated by the linkage rule as possible links need to be manually inspected. But as we mentioned earlier on, manual inspection costs a lot, and should be avoided if possible.

Another solution to the record matching problem was the facilitation of an equational theory which was developed based on human expertise [8]. This approach is known as the knowledge acquisition approach of knowledge engineering. An equational theory is actually a set of rules that captures the semantics of the record linkage process. This set of rules is built based on an interview of a domain expert with a knowledge engineer. Although the researchers presented good results, we should notice that such techniques have certain limitations (i.e., knowledge acquisition bottleneck) and should not be used occasionally.

Concerning the data quality of database answers, Motro and Rakov [12] proposed a methodology in which they rely on an error model, specified a priori, for representing possible errors in the database. They apply machine learning and statistical techniques to identify homogeneous areas in the database with respect to errors. After separating the database into segments of similar quality, they can measure the quality of an answer to a query or a view as a combination of the quality of the basic segments. In this way, they can compute the total quality of any area in the database that represents the answer to a selection, projection, or join query.

Finding the errors that exist in a database--a problem overlooked in the previous approach--is by itself an even harder problem than simply estimating the quality of data by assuming a known error model. An example is the duplicate elimination problem discussed in this paper. This problem is also known as the “merge-purge” problem, the “field-matching” problem, data reconciliation, or in general approximate matching record detection.

An early method devised for addressing the problem of exact duplicate elimination uses a sorting technique which takes an entire record and uses it as the comparison key, in order for identical records to be positioned close to each other. After this step, the database is scanned sequentially and a small window of records is searched for exact duplicate records. An improvement over this technique is presented in [2] based on a modified merge-sort procedure. If the size of the window is small enough, then the complexity of this approach is equivalent to the complexity of the sorting algorithm.

Calling it the “sorted neighborhood method”, Hernandez and Stolfo [7] enhance the traditional approach through the use of multiple passes (different parts of a record are used as keys in each pass for sorting) and then combining the results of all separate passes in a transitive closure step. Along the same lines, Monge and Elkan [10] use the entire record as a key, processing it both from left to right and from right to left, for sorting the database out. In their approach, they use a small pre-specified number of clusters (in union/find structures) that temporarily keeps sets of approximately duplicate records. Then they check whether the currently processed record is a member of one of the existing clusters by applying a similarity checking algorithm between the current record and the representatives of the existing clusters. If the algorithm finds that the record belongs to a cluster, it inserts the record to that cluster. Otherwise a new cluster is created, and the record is inserted in the newly created cluster. If the number of clusters exceeds the maximum amount, then an old cluster is deleted. Although the two techniques outlined above show great improvement over their predecessors with respect to both the reduction in the number of record comparisons and the efficiency of the selected structures utilized, they have two basic flaws: they lack an automatic methodology which generates the equational theory for similarity searching and they suffer from the increased complexity of applying the manually identified theory for similarity.

The work described in [6] is concerned with the reconciliation of the contents of multiple autonomous data sources in the context of materialized views for data warehousing. The three problems addressed in the paper are data errors (mis-typing), the object identity problem [7] and inconsistencies across records that represent the same real-world object. The authors propose a framework which offers three basic services: data transformation, duplicate elimination and multi-table matching. These services are defined as macro-operators and implemented as SQL extensions. By appropriately combining the use of the operators, analysts can describe complex data transformations. The framework explicitly accounts for human intervention in the analysis process, and permits a number of performance optimizations that are specific to data cleaning applications.

Finally, Cochinwala et. al. [4] demonstrate a technique to automatically generate an equational theory for record similarity detection through machine learning and statistical techniques as well as record sampling for reducing the complexity of inducing the model.

6. Summary and further work.

In this paper, we have presented a case study in data quality analysis and improvement for the Italian Ministry of Finances, performed in collaboration with SO.GE.I, the company responsible for the entire data management for the Ministry. Our work focused on taxpayer’s data, and was ultimately successful in addressing two of the Administration’s main concerns. First, we provided a quantitative estimate of the currency and consistency levels of the address data recorded in the main taxpayers’ database, which suffers from currency and format inconsistency problems. Second, some of the taxpayers’ have multiple entry in the database, each with slight differences in the field values, because of a peculiarity in the way the taxpayers are identified in the database. By leveraging previous studies performed by SO.GE.I., we were able to estimate the number of record pairs that are very likely to be duplicates.

These experiments represented the final phase of a work that started by providing the Ministry with guidelines on how to best attack their perceived data quality problems. We began by proposing a simple framework for data quality planning. Starting from a simple data flow analysis, we derived quantitative data quality indicators and suggested how precise quality targets can be expressed in terms of predicates over the indicators. Within this context, our subsequent data analysis effort proved the feasibility of our approach, by suggesting ways to compute values for the indicators in practice.

A few lessons were learnt from this experience. First, we found that dealing with data in unfamiliar domains, such as that of the Ministry of Finances, requires great flexibility in the definition of ad hoc analysis techniques. Although the problems often include the same approximate record matching issues, in reality their setting varies, making a monolithic tool unsuitable for the different tasks. Using our data analysis toolkit greatly increased our effectiveness, allowing us to quickly experiment with different ideas for specific analysis problems. Second, we are now convinced that data cleaning initiatives are bound to be a wasted effort unless they are set in the context of a comprehensive quality plan, and they are aimed at meeting a specific set of long-term goals. Finally, we realized that data flow analysis often requires performing reverse process engineering, not just at the system level, but also on the administrative and business processes that produce and consume the data. Because these are often defined implicitly through regulations and conventions, tight collaboration with the Administration and the system personnel proved invaluable.

This effort may continue in several directions, after this initial assessment, depending on the Administration's priorities. Practical work ranges from the application of the current analysis to the entire database (about 70 million records), to its extension to other realms in the Fiscal Administration. As for research directions, we believe that some of these analysis tasks can be at least partially automated, despite their peculiarities, in a way that goes beyond the current capabilities of our tool. Our current line of research in this area, still at a premature stage, contemplates using automatic classification techniques, a form of machine learning , and association rules to help tracing the causes of poor data quality in large data samples. Specifically, we believe that simple rules of the form "records *A* and *B* match with confidence *c* if conditions $C_1, \ldots C_n$ are true" can be automatically derived from a sufficiently rich training set. Notice that such rules may then be collected into an equational theory of the type described in [8]. This way, we may hope to derive rules that can be used to establish a causal relationship between the observed mismatches among pairs of records in a training set, and their match/no-match classification. One benefit of this approach would be to reduce the space of the features that a classification algorithm must explore to induce an accurate decision tree.

Acknowledgments.

The authors would like to thank Uma Ganapathy and Francesco Caruso of Telcordia Technologies, who took part in much of the actual analysis work, Munir Cochinwala of Telcordia for his constant support towards the data quality initiative, and all the administrative staff that made this international collaboration possible in the first place.

References.

[1] R.Agrawal, H..Mannila, R.Srikant, H. Toivonen, A.Inkeri Verkamo, *Fast Discovery of Association Rules.* In Advances in Knowledge Discovery and Data Mining, U.Fayyad, G. Shapiro, P. Smyth, R. Uthurusamy, eds. AAAI press, 1996.

[2] D. Bitton and D. J. Witt. *Duplicate record elimination in large data files*, ACM Transactions on Database Systems, 8(2):255-265, 1983.

[3] Peter Cheeseman and John Stutz. *Bayesian Classification (AutoClass): Theory and Results*, In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, Advances in Knowledge Discovery and Data Mining, pages 153-180. AAAI Press/MIT Press, 1996.

[4] M. Cochinwala, V. Kurien, G. Lalk, and D. Shasha. *Efficient Data Reconciliation*, Bellcore Research, February 1998.

[5] I.P. Fellegi, and A.B.Sunter,"A Theory for Record Linkage", Journal of the American Statistical Association, 64, 1183-1210, 1969

[6] H.Galhardas, D. Florescu, D. Shasha, E.Simon, *An Extensible Framework for Data Cleaning*, Procs. EDBT, 1999.

[7] M. A. Hernadez and S. J. Stolfo. *The merge-purge problem for large databases*, In Proc. of the 1995 ACM SIGMOD Conference, pages 127--138, 1995.

[8] M. A. Hernadez and S. J. Stolfo.*Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem*, Journal of Data Mining and Knowledge Discovery, 1(2), 1998.

[9] M. Kubat, I. Bratko, R. Michalski., Machine Learning and Data Mining, Methods and Applications, John Wiley & Sons Pub, 1998.

[10] M.A, Jaro, "UNIMATCH: A Record Linkage System, User's Manual", ashington, DC, U.S. Bureau of the Census, 1976

[11] A. E. Monge and C. P. Elkan. *An efficient domain-independent algorithm for detecting approximately duplicate database records*, In Workshop on Research Issues on Data Mining and Knowledge Discovery, 1997, to appear.

[12] A. Motro and I. Rakov. *Not all answers are equally good: Estimating the Quality of Database Answers*, In T. Andreasen et. al., editor, Flexible Query-Answering Systems, pages 1-21. Kluwer Academic Publishers, 1997.

[13] H. B. Newcombe, J. M. Kennedy, S. J. Axford, and A. P. James. “Automatic linkage of vital records” Science, 130;954-959, October 1959.

[14] D. Quass, A Framework for Research in Data Cleaning, Draft, 1999, Brigham Young University.

[15] R.Quinlan, C4.5 - Programs for Machine Learning, Morgan Kauffman publ., 1993.

[16] V. Raman, J. M. Hellerstein, *Potter's Wheel: An Interactive Framework for Data Cleaning and Transformation.* University of California, Berkeley. Submitted, SIGMOD 2000.

[17] G. K. Tayi and D. P. Ballou. *Examining data quality*, Communications of the ACM, 41(2):54-57, 1998.

[18] Umar, A., Karabatis, G., Ness, L., Horowitz, B., and Elmagarmid, A. *Enterprise data quality: A pragmatic approach*, Information Systems Frontiers, 1,3, 279-301.

[19] Verykios, V. S., Elmagarmid, A. K., and Houstis, E. N., *Automating the approximate record matching process*, Information Science, to appear.

[20] Y. Wand and R.Y. Wang. *Anchoring data quality dimensions in ontological foundation*s, Communications of the ACM, 39(11):86-95, 1996.

[21] R.Y. Wang and H.B. Kon. *Towards Total Data Quality Management (TDQM)*. In R.Y. Wang, editor, Information Technology in Action: Trends and Perspectives", Prentice Hall, Englewood Cliffs, NJ, 1993.

[22] R.Y. Wang, V.C. Storey, and C.P. Firth. *A framework for analysis of data quality research*, IEEE Transactions on Knowledge and Data Engineering, 7(4):623--640, 1995.