MASTER DATA MANAGEMENT: A PROOF OF CONCEPT

(Practice-oriented)

Helena Galhardas
Technical University of Lisbon and INESC-ID
helena.galhardas@ist.utl.pt

Luis Torres
Technical University of Lisbon and Link Consulting
luisstorres@ist.utl.pt

João Damásio
Link Consulting
Joao.damasio@link.pt

Abstract: Nowadays, organizations deal with multiple data sources. Therefore, they must cope with data quality problems such as inconsistent and inaccurate data, as well as with data integration and synchronization. In this paper, we present an implementation of an MDM architecture using the open source tool Mural. Furthermore, we report preliminary experimental results that demonstrate the advantages of an MDM system in a real world context.

Key Words: Data Quality, Data integration, Data Cleaning, Schema Mapping, Duplicate Detection

1. INTRODUCTION

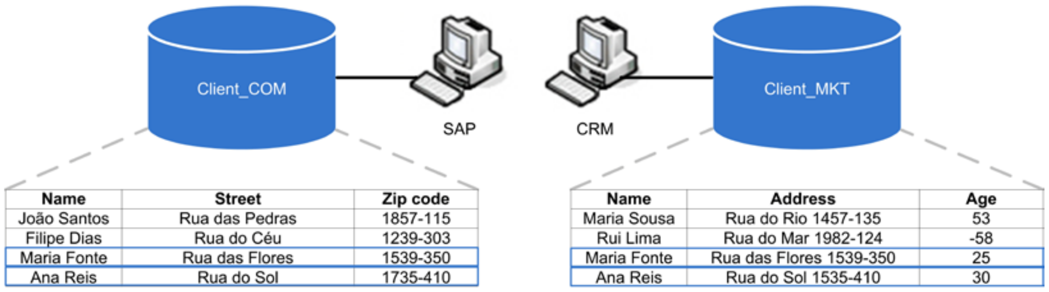
Nowadays, organizations deal with heterogeneous information that needs to be managed and maintained. Given the departmental structure of many organizations, each department manages its own data sources. Organizations have their data spread across multiple systems. This situation often produces data inconsistencies, lack of information, duplicated and non-normalized data. The existence of these data quality problems can lead to inefficient decision making. Consequently, the organization may spend resources unnecessarily or reduce its profits.

Motivating example

Suppose an organization whose commercial department needs to analyze information about its customer's behavior in what concerns invoicing in order to assess the timeliness of payments. The system used for collecting this information is an ERP (Enterprise Resource Planning) whose function is to integrate information from multiple business processes, such as sales, marketing, production and human resources [1].

Consider as well the marketing department that needs more specific information about customers. The system used is a CRM (Customer Relationship Management); it aims at managing information about customers and improving the relationship between them and the organization. In fact, it provides information to coordinate all business processes that deal with customers with regard to sales and marketing campaigns. The two data sources are represented in Figure 1 as tables under the names of

Client_COM and Client_MKT, respectively. Both the data schemas and some data instances are represented.



Name	Street	Zip code
João Santos	Rua das Pedras	1857-115
Filipe Dias	Rua do Céu	1239-303
Maria Fonte	Rua das Flores	1539-350
Ana Reis	Rua do Sol	1735-410

Name	Address	Age
Maria Sousa	Rua do Rio 1457-135	53
Rui Lima	Rua do Mar 1982-124	-58
Maria Fonte	Rua das Flores 1539-350	25
Ana Reis	Rua do Sol 1535-410	30

Clients from Commercial department (Client_COM) and Clients from Marketing department (Client_MKT) (Figure 1)

In Figure 1, we observe that the information is not consistent in both sources. In fact, the street and zip code of client “Ana Reis”, in the Client_COM table (on the left) does not match the corresponding address value stored in the Client_MKT table (on the right). Moreover, there are customers whose information is only kept in one of the systems. For example, the information about customer “Rui Lima” is stored only in the Client_MKT table.

These data quality problems may have a big impact in the organization. In particular, we identify the following three important consequences of the existence of data quality problems.

First, a client may not be considered in a situation where it should be. Given the existence of a marketing campaign tailored to each customer needs, the client "Filipe Dias" may be unaware of the campaign since there is no record referring to him in the Client_MKT table. Therefore, he would not join a campaign that could possible increase his satisfaction with respect to the services of the organization.

Second, performing an action based on inconsistent and outdated information can cause harm to the organization. An example that illustrates this is the case in which the client “Maria Fonte” changes her address to “Rua do Perfume 3231-12” and communicates this change to the organization. The new address is registered within the commercial department. Unaware of this update, the marketing department keeps sending advertisements to the client's old address, and thus wasting resources.

Third, a situation where a client is contacted several times regarding the same subject due to information replicated in both systems may arise. This can lead to the customer dissatisfaction because he is contacted multiple times about the same subject. This might occur with the client “Maria Fonte” during a campaign whose goal is to contact all the existing customers recorded in the Client_MKT and Client_COM tables.

Possible solutions

Several existing technologies provide partial solutions for the data quality problems identified above. We highlight the following two: (i) data integration and cleaning in a data warehousing architecture; or (iii) virtual data integration.

The first type of technology consists in a *data integration and cleaning or ETL tool* [2][4] that transforms data coming from heterogeneous sources and produces target data containing cleaned data. If the target source is a single repository supporting data analysis, it is called a data warehouse [1][3]. There is a main problem associated to this approach. The data source systems are independent from any data repository produced. Therefore, the data quality problems existing in the sources will continue to exist. To eliminate the data quality problems, the organization business processes must use only the data stored in the integrated repositories.

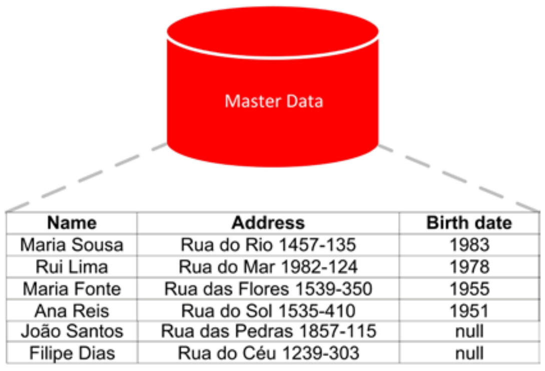
If the target data repository is a data warehouse, there is a process to update it according to the modifications occurring in data sources, which is called data refreshment. Data refreshment is performed periodically (e.g., daily or weekly), except in cases where there are processes that need constantly updated information. This approach has the advantage that the integrated repository is always updated when changes occur in the source systems.

Virtual data integration is a technique in which a virtual unified data view is defined over a set of data sources. Data is kept only in the source systems [5]. The virtual integration architecture encloses a *mediator* (that exports a global data schema) and one *wrapper* for each data source (that exports a source schema). The *mediator* keeps the mapping between its schema and the schema of data sources. A query posed to the mediator is decomposed into as many queries as the number of data sources that are able to answer total or partially to the query. Each wrapper is responsible for translating the initial queries to partial queries understood by each data source, then translating the result and forwarding it back to the mediator. Finally, the mediator integrates the responses of each *wrapper* and displays the result of the interrogation initially placed [5][6]. Although it is a cheap, fast and non-intrusive method for data integration, virtual data integration does not solve all the problems mentioned in the motivating example. In fact, similarly to the previous data integration and cleaning approach, the organization business processes must be modified so that data stored in sources is accessed via the mediator. In addition, this solution does not solve the data quality problems.

Master Data Management

The existence of the problems mentioned in the motivating example and the lack of a complete solution provided by the existing technology require an alternative solution that is able to integrate the various data systems of the organization that contain relevant information. This component must support the definition of an integrated data schema. Moreover, it must provide a data quality control module to solve existing data quality problems. It should also have information broadcasting mechanisms for keeping the data repository synchronized with sources.

Such component is called a *Master Data Management* (MDM) system [8][9] and the data resulting from the integration of existing information in different data sources has the name of *Master Data* (see Figure 2).



Name	Address	Birth date
Maria Sousa	Rua do Rio 1457-135	1983
Rui Lima	Rua do Mar 1982-124	1978
Maria Fonte	Rua das Flores 1539-350	1955
Ana Reis	Rua do Sol 1535-410	1951
João Santos	Rua das Pedras 1857-115	null
Filipe Dias	Rua do Céu 1239-303	null

Master Data - Integrated clients view (Figure 2)

In the motivating example, this component allows the organization to take advantage of a repository containing integrated information about the Client entity from the two data sources. That is, there is now a single materialized view that contains reliable data about the Client informational entity [7][8].

In this paper, we define MDM as a software system whose main function is to enable an organization to create a single point of access to all data concerning an informational entity. This system is responsible for ensuring the quality of master data. Once the master data repository is populated with high quality data, the system must disseminate data to all data sources. Thus, each data source becomes as reliable as the master data repository.

The integration of the MDM system with the data sources is carried out by the following processes: (i) data loading; (ii) data updating; and (iii) data broadcasting.

The *data loading* process corresponds to the initial phase in which the MDM system contains no data. This process is responsible for extracting information about the relevant fields of the master data schema from the source systems. At a later stage, when the MDM system contains data, whenever an update of the sources occurs, the master data repository must be updated accordingly. Reloading all records is too expensive, so a process that is responsible for incrementally identifying the updates that occurred in the data sources and apply them to the master data is required. This process is referred as *data updating*.

The data loaded into the MDM system is subject to a process of data integration and cleaning that increases their quality. Master data contains updated data for each instance of the informational entity. Therefore, an update applied to the Master Data must involve the dissemination of this change to the various source systems in order to keep them synchronized with the MDM system. This process is called *data broadcast*, and can be run in real time or batch mode (e.g., daily at 01:00 am).

Contributions

This work was led by the need of understanding the details involved in an MDM implementation. As far as the authors are aware of, the single work regarding MDM which is described in the scientific literature [12] describes an architecture for data integration in a P2P environment.

This paper has the following two main contributions. First, we describe a prototype of an MDM solution using an open source MDM tool called Mural (https://mural.dev.java.net/). This solution is focused on the Customer informational entity, although it was developed so that it can be implemented with any other informational entity. Second, we report the preliminary results obtained by the validation of the solution implemented in terms of its capability to solve data quality problems concerning Clients in a specific organization. To this end, we evaluated the MDM solution based on a business process. In this validation, we analyzed the data quality problems in two scenarios: (i) the execution of a business process without the MDM solution; and (ii) the execution of the business process with the MDM solution.

Paper organization

This paper is organized as follows. In Section 2, we describe the existing MDM architectures. Then, in Section 3, we present the validation scenario. Section 4 describes the prototype of the MDM system developed. Section 5 reports the results obtained when applying the prototype developed in the validation scenario. Finally, Section 6 presents the main conclusions from this paper and the work identified as future work.

2. BACKGROUND

In this section, we present the existing architectures used to implement an MDM Solution.

Typically, an MDM architecture (see Figure 3) consists of two main types of systems: (i) source systems; and (ii) an MDM System. The source systems are the software systems previously existing in the organization that manage data concerning a given informational entity in a dispersed way. The MDM system is a system designed to integrate the source data and create a unique materialized view of it. The MDM system itself is composed of the following four components: (i) a master data repository; (ii) a metadata repository; (iii) a data quality module; and (iv) a data integration module.

The *master data repository* persistently maintains master data of the organization. The *metadata repository* maintains information about the data kept in data sources and in the master data repository, namely the schema mappings between data sources and the master data repository.

The *data quality module* is responsible for pre-processing the input data of the MDM system. This pre-processing consists of a set of techniques to solve problems in the data so that it becomes reliable. These techniques include data cleaning and data standardization [5]. The *data integration module* is a component that combines data from different data sources in order to provide a unified view of them [11]. The data combination is achieved using schema mappings. This module is also responsible for detecting records from different systems that represent the same instance from the real world.

An MDM solution can be implemented using different types of architectures. Three main architectures have been identified in [8][9][12][13]. In this section we describe only the two most important.

Central Master Data / Transaction HUB

The *Central Master Data Architecture* (see Figure 3) is the most common. The system maintains a persistent and integrated view of all the attributes that represent a given informational entity, which is called master data. The source systems may represent additional attributes of the informational entity, which are only relevant locally.

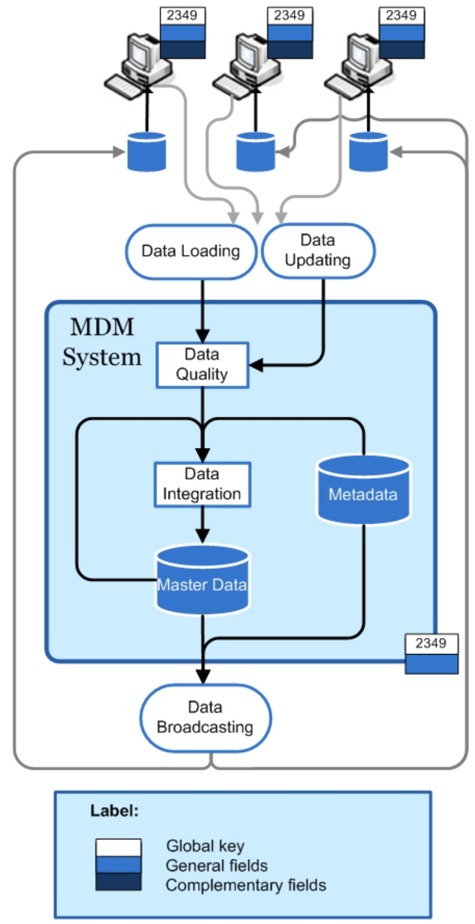
As Figure 3 illustrates, the architecture requires a primary key that must identify all the records in the organization systems, which refer to the same instance of the informational entity (e.g., for the Customer informational entity, the record that identifies the client “João Santos” in the various systems has the identifier number 2349). To this end, all records must be created and modified using the MDM system, that is responsible for creating and managing these keys. The system is also responsible for managing asynchronously the broadcast of the master data to all data sources [8][9][13].

The creation/insertion of new records must conform to norms and standards specified in advance by the organization (e.g., a customer name is always composed by first and last name and the first letter of each name must be capitalized). The use of norms and standards, together with the fact that the architecture is centralized, has the advantage of providing a reliable and synchronized master data repository.

However, this architecture has the problem of requiring that all applications running on the source systems are changed so that the insertion of data takes place directly in the MDM system[8][9][13].

A variant of Central Master Data architecture is followed by some commercial tools, such as *SAP Master Data Management (SAP MDM)* (http://www.sap.com/platform/netweaver/components/mdm/) and *Universal Customer Master (Siebel UCM)* (http://www.oracle.com/master-data-management/cdh.html).

These commercial tools adapt this basic architecture, using techniques to detect data source updates, and turning the creation of new records no longer an MDM system requirement [12].

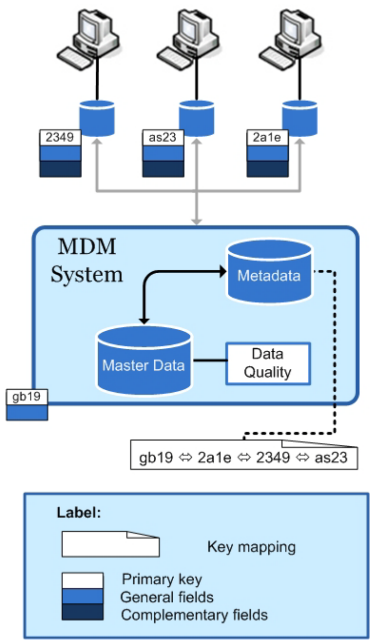


Central Master Data System Architecture (Figure 3)

Repository/Registry System

The *Repository System* (see Figure 4) is an architecture where the MDM system stores the minimum number of attributes required to uniquely identify a real world entity in the Master Data repository. This system stores metadata that allows to find additional data in sources related to these entities in the real world. Each data source has its primary key for each record, and the MDM system is responsible for managing these keys and mapping them into global keys for records relating to the same real world entity [8][9][12][13].

Data about an informational entity are created and maintained independently in each system and not distributed to other systems. These data are accessed through the MDM system when trying to obtain information about any instance of the informational entity (e.g., a particular customer). Initially, a query is posed against the repository of master data to obtain the record identifier. Then, the key mappings stored in the metadata repository are accessed to find references to records of the same instance. Finally, these records are retrieved, integrated and presented to the requesting system [8][9]. In Figure 4, it is possible to verify that the source records identified by “2349”, “as23” and “2a1e” are referenced as representing the master record with key “gb19”.



Repository System Architecture (Figure 4)

Since the data remains in the source systems, this architecture allows the implementation to be faster compared to other architectures. Another advantage is that data integration is made only when a data source requests information about an informational entity instance. When changes occur in the data sources, they are readily available in the local repositories and accessible through the MDM system [8].

This architecture assumes a dependency between the data source and the MDM system. In terms of performance, when a request is made to the MDM system, sources wait the answer from the MDM system, leading to a delay that is propagated to the source systems. A change in the structure of data sources also requires that the MDM system is changed so that the MDM solution remains operational [8].

Data quality is applied to the attributes maintained in the MDM system, being these the minimum number of attributes that uniquely identify an entity in the real world. Since all attributes are kept in the supplementary data sources, it is important that the corresponding data has quality. This architecture is not recommended for solutions that intend to solve data quality problems [8][12]. Moreover, this architecture has the disadvantage of data being created and modified in a decentralized environment and based on different processes [9].

3. SCENARIO

To prove the MDM concept, we implemented an MDM prototype in the **Link Consulting** (http://www.link.pt/) company. The problem presented was to keep consistent and up to date data from two databases of the organization that were inconsistent.

In this section, we present the scenario. We present the data sources used, the master data schema defined for the solution and the mappings between the source and the master schemas. The first data source is the database of the *Communication and Integrated Marketing Department (CIM)*, while the other one is the data repository from the organization CRM system. These two data sources contain information about the organization customers, such as name, address and company. Not all customers are represented in both data sources, some of them being only represented in one of the data sources. There are also customers whose data is not consistent in both data repositories (e.g., address values).

Data Source - Marketing database

This data source consists of the information contained in a Excel sheet that was converted to a database table. It is used by the CIM department to keep information about customers that will be the target of marketing campaigns. Marketing campaigns (e.g., sending *Cadernos Link* - annual production) are regularly sent to customers whose information is in this data source.

This table has seven attributes, namely: (i) **ID**; (ii) **Salutation**; (iii) **Name**; (iv) **Company**; (v) **Address**; (vi) **ZipCode** and (vii) **Country**. The first attribute is the unique identifier of the client system. The second is associated with the greeting for the client (e.g., “Mr.,”). The attribute **Name** is the name of the company representative with whom Link Consulting has contacted. The fourth attribute is the company name. The remaining attributes constitute the client address. The table contains 899 records.

Data Source - CRM

The second data source used is the organization CRM system. This data source stores information about all clients of Link Consulting. The CRM repository consists of several tables. For this solution, we used a view called **CustomerAddress**, composed of seventeen attributes, that gathers the main information about clients and their contacts.

The attributes of the CustomerAddress view are the following: (i) **ID** - Unique identifier of the customer in data source; (ii) **AccountIdName** - Name of the company; (iii) **Salutation** - Greeting to the client; (iv) **FullName** - Full name of the client; (v) **JobTitle** - Position occupied by the client; (vi) **Address1_Name**, **Address1_Line1**, **Address1_Line2** and **Address1_Line3** - Can be used to store the client address; (vii) **Address1_PostalCode**; (viii) **Address1_City**; (ix) **Address1_Country**; (x) **Telephone1** and **Telephone2** - telephone contacts; (xi) **MobilePhone**; (xii) **EmailAddress1** and **EmailAddress2** - e-mail contacts; and (xiii) **Fax**. This view produces 3459 records.

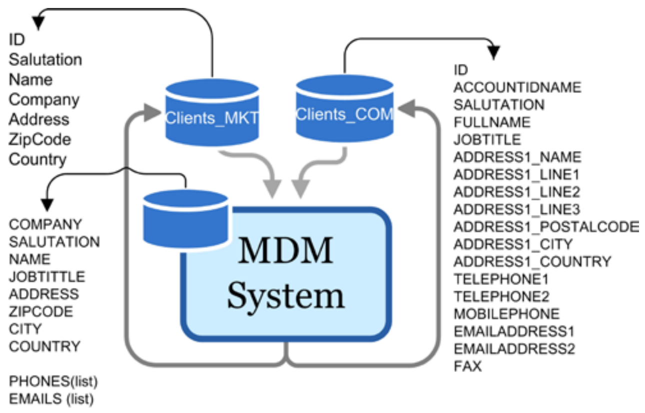
Master Data

During the implementation of the solution, it was necessary to study the data sources and decide the most appropriate structure for the repository of master data. The schema definition for the master data was carried out with employees of the organization responsible for the applications that were using the data sources previously identified. It was decided that this repository should have three tables, one main and two secondary. The main table has the name **LINKMDM**, and keeps general information about its customers. The schema of this table is composed of the following attributes: (i) **Company**; (ii) **Salutation**; (iii) **Name**; (iv) **JobTitle**; (v) **Address**; (vi) **ZipCode**; (vii) **City**; and (viii) **Country**.

In terms of secondary tables, we considered the existence of one for email contacts and another for phone contacts. These tables are named: **Email** and **PhoneNumber**. The first one keeps only the attribute **Email**. The second one is composed by two attributes: (i) **Number** - that keeps phone numbers, and (ii) **Type** - indicating whether the number corresponds to a fixed number (**Tel.**), mobile (**TM**) or fax (**Fax**).

Mappings

To load data into the master data repository, we needed to define mappings between source and master data attributes. These mappings are also necessary to the broadcast process. Figure 5 illustrates all repositories and their attributes. Some mappings are simple, like **Name**, **Salutation**, **Company**, amongst others. Others are more complex and need some data transformation. For example, the **ZipCode** from the Marketing database keeps data from zip code and from city. Thus, it has to be split so that it maps the attributes **ZipCode** and **City** in the master data repository. Inversely, in the broadcast process, these attributes have to be concatenated to match the **ZipCode** from the marketing database. Another example is the case of the address attributes existing in the CRM repository. They need to be concatenated in order to match the master data repository.



Repositories and attributes (Figure 5)

4. PROTOTYPE

The architecture used in this solution (see Figure 6) is a variant of the *Central Master Data (CMD)* architecture, imposed by the tool chosen to implement the MDM solution - *Mural*, an *open source* tool provided by Sun Microsystems . There are three main differences between the two architectures. First, with this variant of the CMD architecture, the organization does not have to change all the applications so that the data is inserted directly into the MDM system. Each application keeps using its data source in the same way as before. The MDM system then applies change data capture techniques to detect data source updates, accesses these data sources and loads the new data into the Master Data repository. Second, the data quality module does not belong to the MDM system. Therefore, the data quality processes are performed before the data is inserted into the MDM system. They are invoked during data loading and updating. Third, the metadata repository is directly accessed by the data loading, broadcasting and updating, in order to map the structure of the data stored in the sources into the data schema of the master data repository. Mural requires that the data entered into the system obeys to the master data schema.

Other tools commercially available have been surveyed in [12]. Mural is extensible, thus we were not limited to the features available and can expand them, if necessary. Mural is implemented in Java and requires the use of *Integrated Development Environment (IDE) NetBeans* (http://netbeans.org/) and Glassfish Java server (https://glassfish.dev.java.net/). Netbeans is used to create and develop the MDM project. Glassfish is a Java server where the MDM solution is executed.

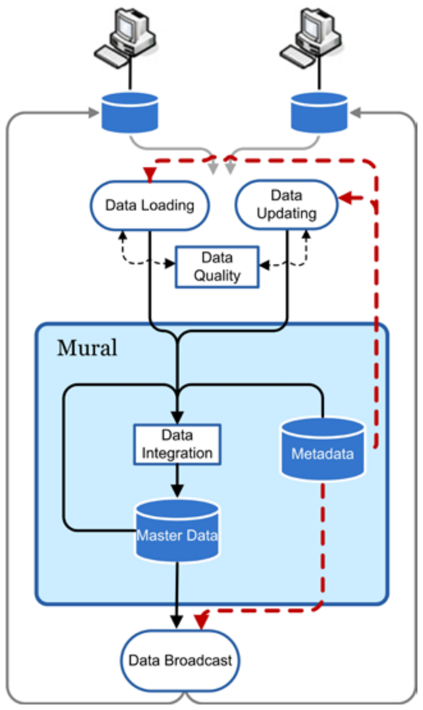
The components of the architecture adopted are as follows:

Master data repository: The master data repository consists of several tables that maintain master data and information used by the tool. These data can be accessed through SQL queries. The tables are divided into four groups: (i) system tables, (ii) MDM main table, (iii) MDM secondary tables, and (iv) tables showing information about duplicated data.

The first group represents the system tables that are used to ensure the proper functioning of the tool [10]. The second and third groups, MDM main and secondary tables, are associated with master data. The MDM main table keeps the information used to uniquely identify an entity in the real world. For example, this table stores the attributes: **client name**, **product code** or **employee number**.

The secondary tables store additional information about the client, for example, **phone** or **product version**. These tables are also used to model a "one to many" relationship between tables, where each record of the MDM main table may be associated with several records of the MDM secondary tables. For example, a customer may have multiple phone numbers or a product may have multiple versions.

The last group, that consists of tables showing information about duplicated data, is composed of tables that aim at identifying the various records that represent the same real world entity. This group consists of the following two tables: **SBYN_POTENTIALDUPLICATES** and **SBYN_ASSUMEDMATCH**. These tables are used internally by the tool when the data is loaded. The first holds a reference to records that the tool considers that possibly correspond to the same real world entity, i.e., pairs of records with a certain degree of similarity. The second maintains information about pairs of records that are identified by the tool as representing the same real world entity, due to the high similarity between them, and have therefore been integrated into a single record.



MDM architecture using Mural (Figure 6)

Metadata repository: The metadata repository is a set of XML documents that store all the information about the schema of master data. These documents also define the main configurations of the tool to ensure the proper functioning. Mural graphical interfaces are automatically generated based on the information contained in these documents.

The most important XML documents are: (i) **Object.xml**; (ii) **Midm.xml**; and (iii) **Mefa.xml**. **Object.xml** stores the meta information about master data. It keeps metadata for all tables that store master data. All MDM table schemas are defined in this document, as well the dependency relationships between them. **Midm.xml** keeps, for each attribute of the master data, integrity constraints and privacy settings. **Mefa.xml** is used to configure the data integration process, to define which attributes to be used in the record comparison and how this comparison is made. This document also maintains attribute normalization rules.

The metadata repository also keeps the mappings between schemas from master data repository and data sources.

Data quality module: Mural contains a data quality module with *data quality audit*, *data cleaning* and *data normalization* features (Information based on documentation available on the website of the tool, https://open-dm-dq.dev.java.net/docs/analysis/dsgn_mi-analysis-overview_c.html}.

We were unable to get this module up and running due to version incompatibilities. However, Mural enables the integration with third-party data quality tools. Thus, we developed a software module in Java responsible for cleaning and normalizing data. This module is focused on the key attributes used in the master data repository, particularly in the attributes that are used for detecting duplicate records (i.e., name, address, zipcode, city, and phones). In terms of data standardization, we used a service provided by Acxiom (http://www.acxiom.pt) for normalizing Portuguese addresses.

Data integration module: The data integration module is responsible for performing the comparison of records that are inserted into Mural. This comparison intends to detect records, from different data sources, that represent the same real world entity. For records that are found to correspond to the same real world entity records, this module is responsible for consolidating them into a single record.

The duplicate data detection process supported by this module uses a probabilistic technique [5] to decide whether a pair of records corresponds to the same real entity. It is divided into three stages: (i) definition of weights; (ii) setting limits; and (iii) establishing integration rules. The first step consists in associating two weights to each attribute that constitutes the master data schema: a weight for the case when attribute values compared are exactly alike, and other for the opposite case. The definition of limits intends to give a higher weight to attributes that better identify or discriminate a real world entity. For the more generic attributes, an insignificant weight is associated. The values assigned to the case of attributes being completely different may even be negative in order to reinforce the idea that if the values of these attributes are different, the likelihood of records being different is even higher. The records are compared attribute by attribute, using comparison functions (e.g., string matching functions). The user may choose one of the pre-defined comparison functions provided by the tool or create new functions from scratch. The comparison yields a value resulting from the comparison of each pair of attribute values. This value varies between two previously defined weights. The value associated with the comparison of a part of records is the sum of the results obtained by comparing the values of all attributes.

In the second step, setting limits, we must sum the weights assigned to each attribute in the case of the values compared are exactly alike. Thus, we obtain a value representative of the result of the comparison of two records exactly alike.
Once the maximum value is obtained, it should be set two limits, one lower and one higher. These limits are set during the solution implementation, and their modification is possible in the future. The *lower limit* is

used to indicate that all pairs of records for which the value of the comparison is less than this limit are considered as two distinct entities in the real world. The *upper limit* indicates that all pairs of records for which the comparison results in a value exceeding this limit correspond to records representing the same real world entity. For the remaining records, i.e., pairs of records where the value obtained after comparing is between the two limits correspond to possible duplicates. These records must be analyzed by the user, in order to see if they really refer to the same real world entity. If this situation occurs, the records must be manually marked as duplicates and then the tool may integrate them into one unique record.

The third step of the integration process corresponds to the definition of rules to merge duplicate records. These rules should define for each attribute of the master data, which source system contains the most reliable information. If the value of a particular attribute of a record to integrate has the value *NULL*, it is considered the value of the other register whose value is not null. Using this kind of rules ensures that the generated record that represents a real world entity contains the most complete information about this entity.

5. VALIDATION OF THE SOLUTION

We validated the prototype introduced in Section 2, in terms of the accuracy of data obtained. In this section, we explain the methodology used and the results obtained. This evaluation is intended to show that the MDM prototype solves the data quality problems identified in Section 1 and thus proofing the advantages of using an MDM system.

We evaluated the MDM solution using a business process used by Link Consulting and compared the results obtained after executing the business process with the MDM Solution and without it. The validation of the solution was carried out by simulating the execution of the business process called "Management of marketing campaigns". For target customers we consider the use of two scenarios: (1) sending the campaign without using the *MDM* solution, and (2) sending the campaign using the *MDM* solution.

We compared the two scenarios according to the following metrics:

- Number of clients contacted two or more times for the same subject
- Number of clients not included in campaigns using the CRM data source
- Number of clients not included in campaigns using the marketing data source
- Number of data inconsistencies

We start by analyzing the results obtained for Scenario (1). The CRM repository consists of **3459** records and the marketing database is composed by 899 records. Analyzing the data sources, we found that there are **381** customers whose information is represented in the two data sources. Thus, there are **3078** clients that are represented only in the repository of CRM and **518** customers whose information is only available in the marketing database. Analyzing the records of customers stored in both data sources, we concluded that **133** records are not consistent, usually due to incomplete addresses.

For Scenario (2), where the MDM solution is used, after executing the data broadcasting process, the number of records of data sources and master data repository was **3927**. In this scenario, there are no clients whose information is only represented in the CRM repository or only on marketing database. There were no inconsistencies in the information of customers who are represented in both data sources. This is because the sources of data were synchronized with the master data and contain the same information.

In Scenario (1), i.e. without using an MDM solution, three data quality problems were detected. First, if the organization focus its marketing campaign on the CRM repository, it would ignore **518** customers. If the focus of the marketing campaign is the marketing database, the number of customers that would not be covered would be **3078**. Second, there were **133** customers whose information was inconsistent in both data sources. Third, there were **411** customers who would be contacted two times if the organization decided to focus its marketing campaign on both data sources. From these **411** customers, **30** are doubly represented in the repository of CRM.

In Scenario (2), i.e., by using the MDM prototype and manually confirming the results automatically produced by the data integration module, these problems were solved by ensuring that: (i) there are no customers to be contacted twice about the same subject; (ii) there are no customers whose information is not in the repositories of CRM, marketing and master data; and (iii) the information contained in the three repositories of data is synchronized and consistent. The results associated with these two scenarios are presented in Table 1.

Measure	**Results**	
	Scenario (1)	**Scenario (2)**
# clients contacted 2x for the same campaign	411	0
# clients not included using CRM	518	0
# clients not included using marketing	3078	0
# clients with different information on both systems	133	0

Results (Table 1)

6.CONCLUSIONS

We presented data quality problems existing in organizations that have multiple data sources responsible for storing data concerning to the same informational entity. The problems identified are: lack of information in data sources, existence of duplicate information and/or outdated and inconsistent data about the same real world entity. Based on these problems, we presented the MDM as a possible solution and described its characteristics, typical architectures and processes. Finally, we have shown how we validated the implemented solution in a real-world context.

Future work could be approached in two ways: (i) solution optimization, and (ii) solution validation.

To optimize the solution, the comparison functions used by the duplicate detection algorithm could be improved in order to produce more accurate results. Moreover, it would be necessary to adjust the limits used by this algorithm. In what concerns the criteria used for merging duplicates, it would be interesting to create heuristics to analyze the data instances for deciding which value must be inserted in the master record. Finally, the data quality module developed could be improved. For example, some type of service provided by operators of national telecommunications to validate phone numbers could be used.

The solution validation was developed around the Customer informational entity. The prototype developed could be validated using other informational entities. Other validation scenarios, where more than two data sources and more complex business processes are available, should also be tested.

REFERENCES

[1] Laudon, J. and Laudon, K., Management Information Systems: Managing the Digital Firm. 10th Edition ed. Prentice Hall, 2006.

[2] Rahm, E. and Do, H. H., Data cleaning: Problems and current approaches. IEEE Data Eng. Bull., 23(4) 2000, pp 3-13.

[3] Chaudhuri, S. and Dayal, U., An overview of data warehousing and OLAP technology. ACM Sigmod Record, 26(1) 1997, pp. 65-74.

[4] Kimball, R. and Caserta, J., The data warehouse ETL toolkit. Wiley New York, 2004.

[5] Batini, C. and Scannapieco, M., Data Quality: Concepts, Methodologies and Techniques (Data-Centric Systems and Applications). Springer-Verlag New York Inc, 2006.

[6] Panchapagesan, B., Joshua, H., Wiederhold, G., Erickson, S. and Dean L., The INEEL data integration mediation system. In ICSC Symposium on Advances in
Intelligent Data Analysis (AIDA). Rochester, NY, 1999.

[7] Wise, L., The intrinsic value of master data management, March 2010, http://www.information-management.com/news/10001093-1.html, 2008.

[8] Dreibelbis, A., Hechler, E., Milman, I., Oberhofer, M., van Run, P. and Wolfson, D., Enterprise Master Data Management: An SOA Approach to Managing Core. Information. IBM Press, 2008.

[9] Loshin, D., Master Data Management. Morgan Kaufmann, San Diego, 2008.

[10] Inc. SunMicrosystems. Understanding sun master index processing (repository). Technical Report 820-2667-15, SunMicrosystems, Inc., 4150 Network Circle Santa Clara, CA 95054 U.S.A., 2008.

[11] Lenzerini, M., Data integration: A theoretical perspective. In Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS). ACM New York, NY, USA, 2002 pp 233-246.

[12] Kokemüller, J. and Weisbecker, A., Master data management: Product and research. In Proceedings of the Fourteenth International Conference on Information Quality (ICIQ), Potsdam, Germany, November 2009, pp 8-18.

[13] Loser, C., Legner, C. and Gizanis, D., Master data management for collaborative service processes. In International Conference on Service Systems and Service Management, Beijing, July 2004, pp19-21.