Scalable Computing: Practice and Experience
Volume 12, Number 1, pp. 51–61. http://www.scpe.org
ISSN 1895-1767
© 2011 SCPE

AUTOMATIC DATA MIGRATION E-SYSTEM FOR PUBLIC ADMINISTRATION E-SERVICES

CIPRIAN DOBRE[*] AND ANDREEA MARIN[†]

Abstract. Interoperability represents an important issue in developing successful e-Services applications. In such applications data is often produced by services specifics to one organization and is consumed by services belonging to another one. Still, automatic data migration and transfer utilities are currently insufficiently addressed. We present a solution for automatic data migration that is able to handle data transfers between different database instances. The solution works with dynamic data schemas, and requires minimal user input and interaction. The proposed e-System was successfully used to support public administration automatic collection of data about companies. Various government processes require information from businesses. Examples include statistics about number of employees, revenue data, etc. However, collecting such data generally requires it to be voluntarily transmitted by business organizations. We present results for evaluating in a real-world e-System the proposed solution, as a backbone designed to automatically collect reporting data.

Key words: data migration, automatic data conversion, interoperability, e-System

AMS subject classifications. 15A15, 15A09, 15A23

1. Introduction.

Many e-Systems are made by composing the functions provided by services running in different organizations. Such services often use similar data, stored in different ways. Service composition often uses the data provided by one service as input for other service. In this case interoperability, the way data is semantically linked between these two systems, becomes a critical issue.

Organizations usually collect large volumes of data in their internal databases. In many cases, the stored data refers to the company structure, employees, customers and projects. With the development of new database systems, companies might want to change the database in use and transfer large amounts of data. Moreover changes in company structure and number of employees might generate modifications in the database schema. As a consequence, not only that the underlying database system is changed, but the data transfer would occur between two different schemas.

Data Migration is not an isolated process. A typical scenario involved the implementation of an application which requires data already existing in current storage devices (e.g., already used by other applications). Such a scenario is a typical premise for data migration. Recent studies show that the industry spends as much as $5 billion on data migration, considering software and consulting services [1].

This paper extends the results presented in [12]. We describe the design and implementation details for the system designed to automate the data migration between various data sources. As such, we describe how our solution is able to handle data transfers between different database instances. It is designed to work with dynamic data schemas, and requires minimal user input and interaction.

In addition, we present results for evaluating the proposed solution in a real-world e-System, as a backbone designed to automatically collect reporting data. An important problem with public administration is collecting data about companies. Various government processes require information from businesses. Examples include statistics about number of employees, revenue data, etc. However, collecting such data is not an easy task. It generally requires the data to be voluntarily transmitted by business organizations. In many cases on the business side the data, instead of being electronically transmitted, is copied manually from the database in the internal back-office system and then sent to the public administration entities. In this context the proposed e-System was successfully used to support public administration automatic collection of data about companies.

The paper is organized as follows. In Section 2 we present related work. Section 3 gives a broad view of data migration and presents its importance in any business environment. Section 4 presents the architecture of the data migration e-system. Details regarding the implementation of the system are provided in Section 5 and Section 6 describes a form of usage of the system. Section 7 gives conclusions and presents future work.

[*]Computer Science Department, Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania(ciprian.dobre@cs.pub.ro).

[†]Computer Science Department, Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania(andreea.marin@cti.pub.ro).

2. Related work. Data migration has been previously studied in research literature. A number of solutions ([2] [3] [5]) were previously proposed, each having various limitations. A number of research papers ([1]) analyzed the pros and cons of such solutions, presenting clear explanations about their differences and problems.

In [1] the authors make a distinction between data migration and traditional data moving solutions. Furthermore, the authors make an analysis about how large volumes of data can migrate between database systems and what are the steps in developing an efficient migration strategy. At present there are various methods for performing data import or export. Various software applications as well as database systems offer the possibility for exporting or importing data. Furthermore they allow exporting data in various known formats: csv, xls. Consequently they allow data import from these types of files also.

Previously, several research projects and software products tackled ideas related to our proposed solution. We next present details about several such solutions, which represent the state-of-the-art of this area. According to [2], SQL Server Integrated Services (SSIS) provide the developer the necessary means to develop workflows for data transfer. The workflows take the form of packages that contain data and control flow information, used both for creating mappings between source and destination, but also for connecting them. Moreover, the means for data transfers are provided also by SQL Server Management Studio Express, in the form of tasks that can be executed on top of the database. This utility transfers data in known file types (csv, xls) or in Oracle and SQL Server Databases having the same schema. On the other hand, the transfer depends on the Oracle Database System's version (10g, 9i).

Talend Open Studio is a very complex data integration and business modeling software product. According to [3] using specific jobs in Talend Open Studio can help transfer data between any two sources. It offers wide support for the most important database systems and for the best known file types. What it may be considered as an inconvenient is that the schemas of the source and destination entities have to be known at design time for the developer to be able to create mappings between them. However, Talend Open Studio provides a very user friendly graphical interfaces and offers broad connectivity for both target and source systems.

Unlike previous work, the solution presented in this paper tries to automate the steps in converting the data and automatic mapping of fields, even if they pertain to different data schemas. The solution can be used to automate many of the steps required when services require the data produced by other services, or the migration of data between different databases.

3. Data Migration. Data migration is a challenging process that aims to transfer existing data into a new environment. This transfer is made by companies when starting to use new software products or upgrading to a new type of database. These changes are frequent in the business world as both databases and software products are evolving quickly.

As described in [7], a series of factors should be taken into account when planning data migration within a business environment. On the first hand a high volume of data is involved. The migration process must be supervised and also capable of treating interruptions. The problems that might arise during the migration process might affect the integrity and consistency of data. Furthermore, the majority of migration processes often occur between heterogeneous environments. An aspect that should be taken into account is the set of transformations required by certain data types. This aspect is treated by the software solution presented in the paper.

Usually, data migration is a static lengthy process that requires careful planning. According to [8], the following steps should be followed when planning the data migration process:

1. Analyze and define the source database structure: tables, data types.
2. Analyze an define the destination database structure: tables, data types.
3. Perform field mapping: mapping between the source and destination data, data type transformation.
4. Define the migration process.

As it is described in the paper, the proposed software system provides a dynamic environment for executing the above tasks. The end user is presented with a very friendly and easy interface, and in the same time with a product that can perform data migration using a dynamic approach.The importance of data migration is explained in [9]. One of the most important factors that determine this process is its cost. A significant amount of the IT budget, as well as labour, is spent on data migration. Data migration is carried on frequently in companies thus increasing the amounts of money invested in such processes. Projects regarding data migration can become very complex creating a very large market for labor, consulting, software and hardware.

4. The architecture of the automatic data migration system. The architecture of the Automatic Data Migration System is presented in Figure 4.1. The system maps data between two data sources. It can then dynamically transfer the data whenever required.

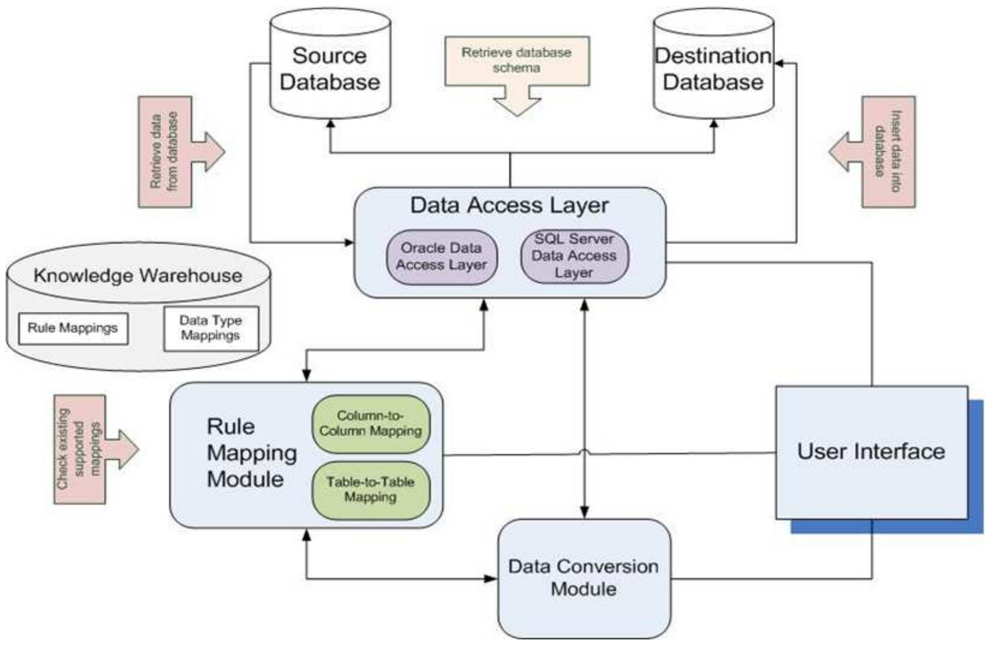


FIG. 4.1. *The proposed architecture.*

The system first loads the data schema (fields, relationships, etc) from the first data source. The user is then presented with an intuitive representation of the data schema. The user has full control over the data migration process using this data interface. On the other hand, the system includes mechanisms to automatic recognize data types and map data fields belonging to different data sources. It can be instructed to move data on various triggered actions, even if conversions among data types are required.

The architecture consists of specialized modules that interact with each other maintaining a continuous flow of data. The main components are as follows. The three main modules of the software application are the rule mapping module, the data conversion module and the data access layer. These modules are controlled indirectly by the graphical user interface.

The data access layer retrieves the database schemas in order to be displayed by the user interface. In addition it is used for database specific operations such as data retrieval and insertion.

The rule mapping module can be used to register mapping rules in the system, to manage and send for execution against the involved databases. The rule mapping module checks the validity of the rules using a repository. The repository consists of several implicit and explicit mappings that are supported by the application.

The data conversion module is used for the data conversion that must occur during data migration. In addition, a knowledge warehouse is used for storing conversion mappings between known data sources.

5. Implementation details. The implementation of the e-System supports data migration/transfer between data sources without having prior details about the structure of the data. The implementation of the automatic data migration system considers several of the most known data sources.

In the majority of batch data transfer applications the schemas must be defined at design time. For example, the Typed Data Sets from .NET is a data mapping that can be used in design [6]. However, such an approach cannot be used for implementing the migration system. In our system no schema is defined at design time and this increases the usability of our system for non-technical users. This is an important improvement because many data migration projects have to be developed and run by database specialist.

The system allows the user to perform data migration activities on his own, without him having advanced technical knowledge over database systems. Furthermore the application supports migration between similar and different databases. For example, one of the first features supported was data migration between two SQL Server databases. In addition, the second important feature supported was migration between an Oracle database instance and a SQL Server one.

Concerning security issues, the only information the user has to know is the credentials of these databases. As a consequence, both databases must allow remote connections and the user must have administrator rights on the system on which he is running the data migration system.

Software requirements for the proposed system include Microsoft .NET Framework 3.5, Microsoft IIS 7.0 and Microsoft SQL Server 2008 Express. Microsoft .NET framework is used for the implementation of all the three modules. The database is the main component of the Knowledge Warehouse, used to store mapping rules information.

In the following paragraph we present details about the implementation of the main components.

5.1. User interface. The user interface supports several functions:

1. Authentication.
2. Creation and management of mapping rules.
3. Display of data schemas, conflicts or resolutions.
4. Display of migration results.

The user inserts the credentials for the source and destination data sources. The user interface supports a wide range of authentication mechanisms, designed to support connections to various databases and flat file sources. After the credentials are verified and a connection is created, the data schemas are retrieved from the both data sources and they are displayed. All information is displayed in a web portal, designed using .NET platform TreeViews. An example of this portal is presented in Figure 5.1.

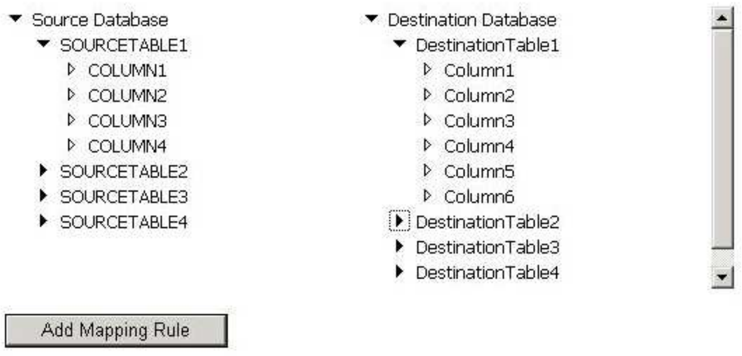
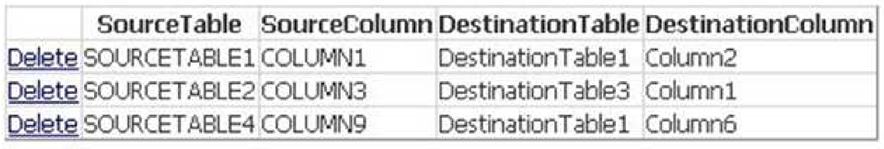


Fig. 5.1. *The data sources structure.*

Using the interface, the user can browse through the tree structure and select the columns and tables he wants to use for constructing the mapping rules. The mapping rules are dynamically created by selecting a source end point and a destination end point. The end points are selected from a list of available data sources.

After the mapping rules are created they are added into a list and can be further reviewed and managed. The list, presented in Figure 5.2, is available for the user to delete the rules he does not want to continue with. After the rules are saved the user can proceed to the rules checking and execution. If all the mapping rules have structures similar to the supported mapping rules, the migration can occur. If not the user is displayed with a list of conflicts and the possibilities to resolve them. This feature is explained in the following sections.



	SourceTable	SourceColumn	DestinationTable	DestinationColumn
Delete	SOURCETABLE1	COLUMN1	DestinationTable1	Column2
Delete	SOURCETABLE2	COLUMN3	DestinationTable3	Column1
Delete	SOURCETABLE4	COLUMN9	DestinationTable1	Column6

Fig. 5.2. *Examples of mapping rules.*

5.2. Rule mapping module. The rule mapping component is used to create mapping rules according to the specifications of the user, to verify the correctness of the rules and, if all the conditions are verified, to further send the rules for execution against the database. The system supports two types of mappings: column-to-column mappings and table-to-table mappings.

The column-to-column mapping is described as follows. By means provided by the interface the user can select the source and destination location of the data. The user can choose any column from the first database (the source column) and any column from the destination database (the destination column). Also, this module uses the Knowledge Warehouse component to automate the process of column mapping. This component includes various predefined rules of mappings columns. For example, if two columns have the same name (for example the name of an employee or salary) then the user is already presented with the mapping rule. Also, if the user previously selected a mapping rule and the same two columns are again used in another selection, again the two columns are already mapped. However, the user has full control over these predefined mappings.

After the two columns are selected the user can register his mapping in the system. In the end, when all the mappings are registered the user instructs the system to send the rules to be processed against the databases.

The table-to-table mapping is similar to the column approach, but instead of columns, the end point entities are tables. This feature was added to support the case study reporting system that is presented in the next Section. This reporting system considers that there is a slight possibility that users have similar data structures in both source and destination databases. When this kind of mapping is performed the system checks that the involved tables have the same structure exactly. This is equivalent with comparing two hash tables having the same key values, in this case, the keys being the column names. If the column names and data types are not the same the migration cannot be performed and the user is notified accordingly, being required to act in concordance with the best suited solution for his needs.

The rule mapping module also allows managing the created rules. Before starting the data conversion, the user can check the created rules and decide whether he wants to use all of them or not. Both, the user interface and the rule mapping module provide the ability to delete the rules considered unsuitable.

The module also supports the management of user generated errors, such as creating an incorrect mapping. An incorrect mapping can be defined as a mapping whose end point entities do not have the same type. For example a user can create a table-to-column mapping, which would generate serious damage to the data migration process if not discovered prior to the start of rule execution.

5.3. Data conversion module. This module assists the rule mapping and migration functions of the e-System with data conversions. This has to be performed according to the user needs and in such a way that data is not altered during the conversion. An incorrect data conversion can negatively affect the applications relying on that data as well.

The module supports two types of conversions:

1. Implicit conversions.
2. Explicit conversions.

Implicit conversions are made between well-known database data types [4]. Examples include conversions between SQL Server, Oracle or other databases, such as Binary to Raw, Image to Long raw and others (see Table 5.1). Implicit data conversions are automatically created by using predefined rules provided by the Knowledge Warehouse component.

The explicit conversions are used for conversions between non-compatible data types. For example, for some situations the user might want to convert numerical data types into strings or vice-versa. Such conversions are part of the requirements that the user is presented with an large collection of data conversions. Table 5.2 presents several such explicit data conversions.

To prevent possible data losses, when the user requires such a conversion, he is presented with a warning message stating that continuing this action might alter the data. As a consequence, the user must agree on continuing with the data conversion. If the user does not find the data conversion suitable for his requirements he has two choices:

1. To perform just the data conversions that are considered safe and then alter the destination database; or
2. To stop the whole migration process, alter the destination database and restart the migration process.

This component is also capable of automating the conversion processes. For implicit conversions the system is able to recognize formats and correctly manage data migration between columns of compatible data types.

Table 5.1
Examples of implicit data conversions.

Data type	SQL Server	Oracle	MySQL
boolean	Bit	Byte	N/A
integer	Int	Number	Int Integer
float	Float Real	Number	Float
currency	Money	N/A	N/A
String(fixed)	Char	Char	Char
String(variable)	Varchar Nvarchar	Varchar Varchar2	Varchar
Binary object	Binary Varbinary Image	Long Raw	Blob Text

Table 5.2
Examples of explicit data conversions.

Source data type	Destination data type
String	Int
Boolean	Int
Boolean	Bit
Int	String
Boolean	String
Char	Boolean

Also, the Knowledge Warehouse stores information for both implicit and explicit data conversions. Whenever a data conversion rule that does not exist in the repository is created by a user, and the rule is validated by the data conversion module, it is saved for further usage in a temporary storage space. The system administrator checks the temporary tables periodically and if the rules are in concordance with the system's requirements and purpose, they can be added to the permanent mapping tables.

6. A case study. The application presented in this paper was tested and used in an e-Services system for public administration reporting services. The e-Services system is able to provide optimized automatic data transfers, document workflows and business reporting of business organizations. Its main purpose is to optimize businesses improving management of routine tasks such as periodical reporting data or automatically managing interactions between the organization and the public administration.

In many countries companies are required by government to periodically report their fiscal, social and statistics information to various institutions. Many data required by these reports are redundant and in many cases the company has to report the same information to different institutions. Such reports and the methodology for filling them are well documented by the local legislation. Many believe the public administration will soon be affected by various changes and challenges due to the private sector's expectations [10]. In Romania, in particular, many institutions adopted their own "in-house" systems for managing data using different technologies. The existing heterogeneous means of communication can have a negative impact on the reporting workflow between enterprises and public authorities. Due to this issue it is important to model the reporting process as close to administrative reality as possible.

The architecture of the e-System (presented in Fig. 6.1) involves a high degree of interaction between the system and the public institutions. It consists of several software components, called Processors, responsible with the automatic generation of reports based on the data available within a business organization. A Central Authority is responsible with communicating with all processors, updating their rules and report templates and managing their processes. In addition, public institutions interact with the Central Authority and provide information regarding the reports using a natural language.

The role of the Central Authority is to represent a connection between the companies that use a Processor and the Public Institution. From a general view the Central Authority takes input from the Public Institutions and provides as output information for the Processors. Any public institution involved in the workflow can send information regarding a report to the Central Authority. This type of information can contain the structure of a report, as a pdf form and the methodology describing how the form must be filled in, which information it

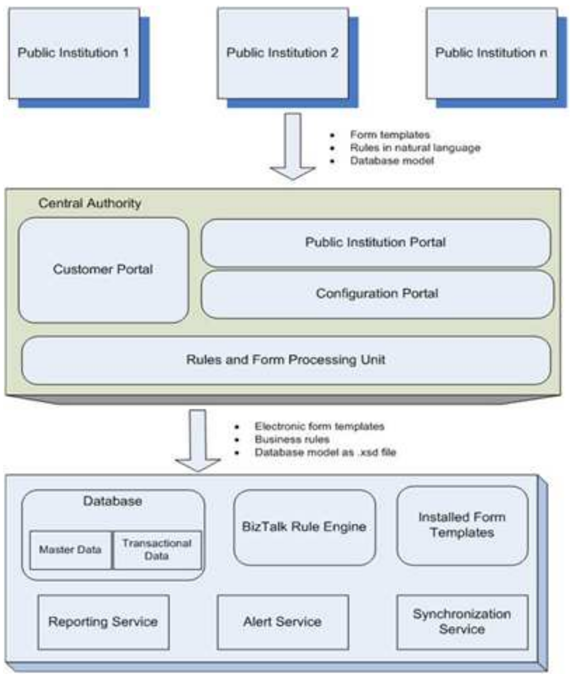


Fig. 6.1. *The architecture of the reporting e-System.*

should contain, the deadline until the completed form should be returned to the public body and the periodicity of the form (yearly, quarterly, monthly, weekly). The Central Authority creates three components that are sent to the Processors registered with it in a single communication package. The first component is an InfoPath form that tries to resemble closely the original form sent by the Public Authority. InfoPath is a Microsoft technology used to easily create and fill in forms. However it is based on open source standards such as XML and XSLT. Characteristics of this type of form include navigation, web browser compatibility and automatic completion of data based on a specified data source. The second component is an XSD schema that describes the types of data used by the report. The last component is the formal set of rules created to represent the methodology in an electronic format. These rules are written and interpreted using the BizTalk engine. BizTalk is a technology that allows integration and communication between various heterogeneous applications. In the current project it was used to integrate a series of web services and to create workflows inside the Processor according to the rules mentioned above.

The Processor, that is called the Pro Processor or simply Pro in the project, represents the core of the reporting system. Its components work together in order to automate tasks and complete periodical workflows. Each Processor has its own database. This database contains two types of data: master data and transactional data. Master data represents data that does not change frequently during a company's lifetime, like company name, address, different elements of identification given by the public bodies, employees' information. Transactional data represents data that has to be inserted periodically (weekly, monthly) into the Processor's database or data that can be computed using a combination of master data and other transactional data. One of the roles of the Processor is to infer transactional data when possible. For describing a typical workflow in the Processor we will consider a moment in its lifetime when data resides in the database, the transfer of data during the initialization of the reporting system will be explained in a following paragraph in the current section. The Processor will receive from the Central Authority a communication package containing an InfoPath form, a XSD template and a set of BizTalk business rules as mentioned above. The form will be installed in the system with its additional information, such as periodicity, data needed and the roles of the persons responsible for filling in the form. The XSD template is taken as input by one of the services of the Processor. This service creates a new database according to the contents of the template. Furthermore, the Synchronization Service is responsible for copying necessary data in the new database from the existing one. The third component of the communication package, the set of BizTalk business rules, is installed in the BizTalk Rule Engine. The Alert

Service is responsible for starting a reporting workflow. This service is aware of the deadlines of each report, information that resides in the Processor's database. Whenever a certain deadline is met, this service notices the Reporting Service which tries to fill in the report with the data needed. After filling in the report it is send to the person inside the company that is responsible with it. This person can verify, correct or fill in missing information. Lastly, the form is signed. If any information was added or changed in the form, the Processor is aware of these changes and propagates them into the database. After the form is signed, the Processor sends it to the Public Administration body responsible with collecting that type of information. The described workflow is presented in Fig. 6.2.

Several modules are involved in the implementation of the processor. The first one is the configuration module and is responsible for configuring the master data inside the PRO. The second module is the data acquisition one, which allows the PRO to interconnect to other local systems inside the companies. Next, the data processing module generates the reports accordingly to the business rules implemented. The eSafe module stores all the required data securely. The last module is report dispatching, which accomplishes the transportation of reports to the corresponding public institutions [11].

The main modules shown in Fig. 6.3 are described further. The Portal is responsible for data entry for both master and transactional data and consists of a number of sections, each section with a corresponding web form and a corresponding XML mapping considering the PRO internal XML representation of master and transactional data. Each section has one or more persons responsible for filling the data. There are two types of sections: master sections and transactional sections, for the corresponding data types. There may be a third section for "user and role management", where a designated administrator can manage the users authorized to fill in and sign the form. The *Data Warehouse* is responsible for storing both master and transactional data. Here, a module stores and selects the different versions of the business rules.

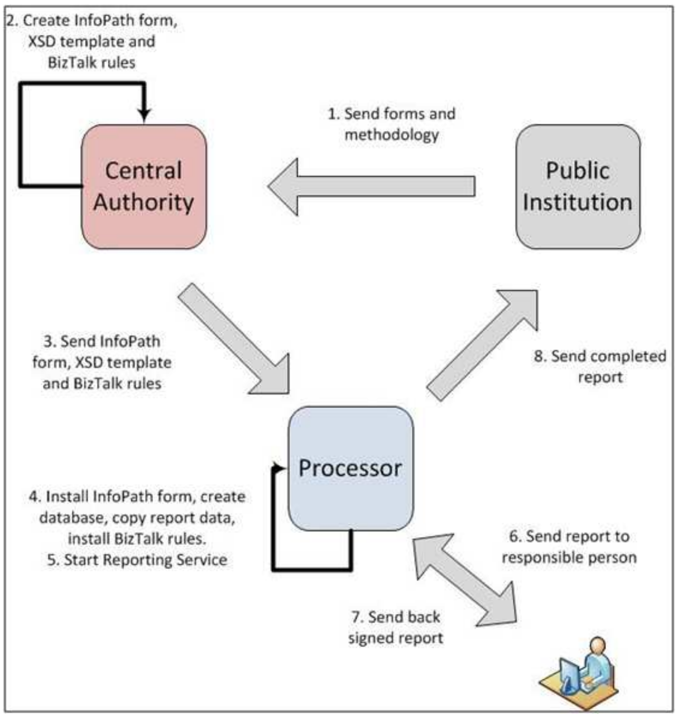

2. Create InfoPath form, XSD template and BizTalk rules
1. Send forms and methodology
Central Authority
Public Institution
3. Send InfoPath form, XSD template and BizTalk rules
8. Send completed report
Processor
4. Install InfoPath form, create database, copy report data, install BizTalk rules.
5. Start Reporting Service
6. Send report to responsible person
7. Send back signed report

FIG. 6.2. *Workflow for reporting services.*

The *Central Management Unit* is responsible for starting and scheduling jobs. For starting a job the services involved in this unit test if all the required data is available and if so the job is started. If the data is not available yet, PRO sends alerts via email to the users responsible for providing the missing data. This unit's roles include logging system's states and managing functional reports. Furthermore, it communicates with the Central Authority according to the protocol defined for updates and deployment. It also sends completed reports to the Public Institutions. The jobs are executed by the *Business Rules Processing Unit.* Security issues and certificates are managed by the *Security Module*. The *Report archiving module* is responsible for archiving the XML reports following predefined rules.

Two types of adapters are shown in Fig. 6.3. The *enterprise adapter* implements the enterprise specific interfaces to PRO. It is responsible for mapping the PRO internal XML representation of reports to enterprise specific forms and vice versa. It also maps the enterprise specific representations of master and transactional data to PRO representations. The *public institution adapter* is a module that implements the public institution specific interfaces to PRO. As in the case of the previous adapter, it is responsible for mapping the PRO internal XML representation of reports to the public institution specific forms and vice versa.

The role of the *Data Interpretation Service* is collecting data for the Processor. For achieving this, the Service handles two types of data: master data and transactional data, which were described before in the current section. Master data is submitted in the Customer Site of the Central Authority and is transferred along with the processor. Transactional data, on the other hand, is obtained by aggregating master data and other transactional data using the Business Process Rules that are received from the Central Authority. The *Data Interpretation Service* uses the migration system described in the paper. The migration system is configured through its interface and helps the company set up connections between local back-office systems and the Processor. The system further connects to the company's database and to the database inside the Processor. The user must provide credentials and localization data for its own database for the system to be able to connect. After the connections are completed, the user can see the extended structure of the both databases. Using the web interface, the user can define mappings of columns and tables, as previously described in the paper.

The communication between the Public Institutions and the Processor is done through means of email. An email usually contains a completed form.

The reporting system was developed using Microsoft .NET platform, with C# as the main language. Other technologies used are InfoPath forms and XML, which is involved in the communication process between the entities mentioned. Furthermore, for implementing such a system in a real life environment, the entities involved should have the following system configurations.

For the client side, the Processor consists of two servers. Both servers should have as operating system Windows Server 2008 SP1 Standard Edition. The first server is responsible with the security of the Processor and has the following software products installed: Active Directory Domain Services, Active Directory Certificate Service and Internet Information Services (IIS) 7.0. The second sever of the Processor is responsible with the reporting workflow and the software installed consists of the following products: Internet Information Services (IIS) 7.0, Microsoft SQL Server 2005, Microsoft BizTalk Server 2009 and Microsoft Office SharePoint 2007.

The Central Authority can be implemented using a single server running Windows Server 2008 SP1 Standard Edition operating system. The following software products have to be installed on the server in order that it works properly and can fulfill its designated tasks: Internet Information Services (IIS) 7.0, Microsoft SQL Server 2005, Microsoft BizTalk Server 2009, Microsoft Office SharePoint 2007, Active Directory Domain Services and Active Directory Certificate Service.

The Public Administration should have an operating system that permits the installation of Microsoft Outlook 2007 and Microsoft InfoPath 2007. The type of operating system used in the experiments was Windows Vista SP2.

The role of the data migration software is to provide the initial data for the reporting system. The reporting system in discussion has its own database system. At the first use of this system, the only information available is the database schema. For the initialization a series of internal data is needed. The data is considered to reside partially or totally in the userś own database. For the transfer of the data between the two databases, the presented software is used. The issues arising from this approach are: which data is needed for the transfer, how the data is transferred and what is the structure of the source and destination database schemas.

Based on this architecture, we developed a pilot implementation based on the business realities in Romania. The basis of such an implementation consisted of a series of specific Romanian public administration documents used in domains such as social insurance, environment and fiscal reporting. The analysis of the reporting processes between businesses and public authorities revealed the existence of 4 classes of reports, for reporting fiscal, social, environmental and statistical data.

Such reports can generate a lot of documents and bureaucracy that can have a negative impact on performance of the private business enterprises. For the pilot implementation we concentrated, in particular, on the 010 Fiscal Registration Declaration, officially known as declaration of amendments for judicial persons, associations, and other entities without judicial personality. The sensitive data that has to be inserted into this report consists of the identification data of the taxpayer and the categories of declaration tax obligations according to the law, hereinafter called fiscal vector. The fiscal vector can be defined as the permanent obligation of the taxpayer and consists of data regarding the following financial aspects: VAT, excises, petrol tax and natural gases from the internal production tax, gambling tax, profit tax, fiscal royalties, micro corporation income tax, wages income tax, special taxes such as: social health insurance tax, unemployment tax, professional illness and accidents tax, social insurance tax. The migration was done between two different database types. The source database was Oracle 11g and the destination database was Microsoft SQL Server 2008. In this context the migration system optimized the overall performance of the reporting system by ensuring a transparent data retrieval process. It allows the user to easily manage the data mapping processes for filling the required reports.

Without the migration service, the user should be presented with mechanisms to export the data required by the reporting system manually. The data cannot even be directly imported, manually conversions being needed for many of the fields required. Using the migration service there is no need to have a data operator to deal with the data, many of its tasks being performed automatically. Therefore, the system ensures a lower operating time and fewer errors when moving the data.

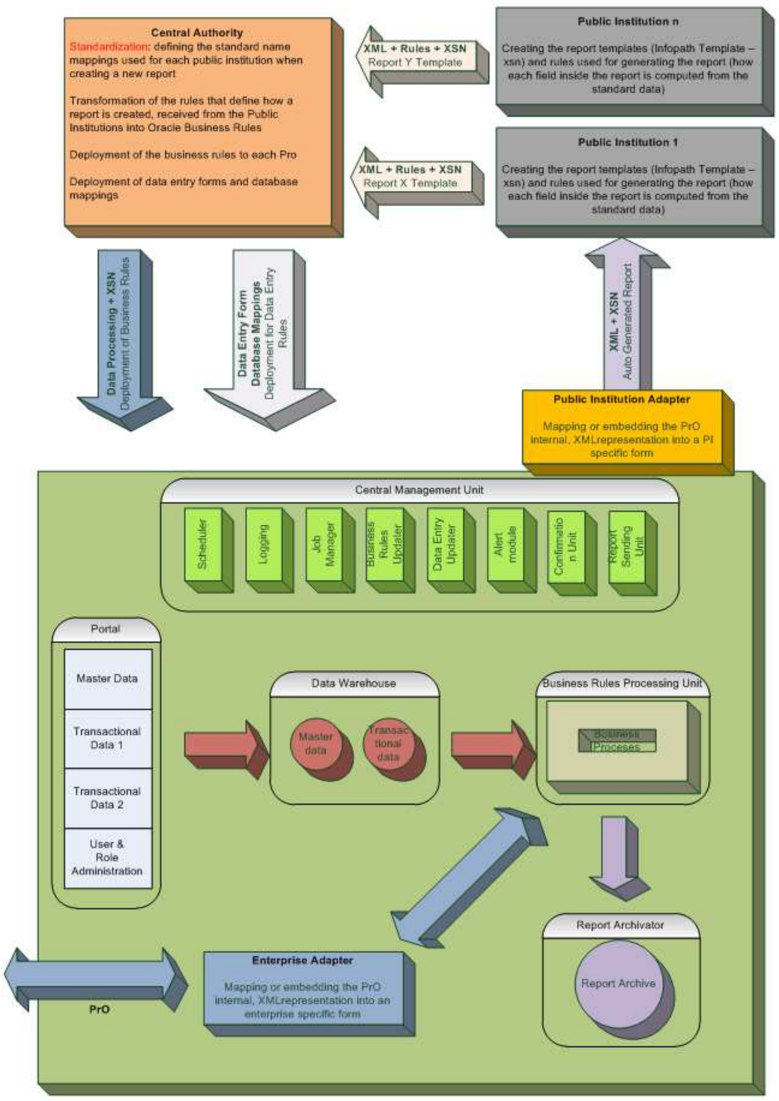

Central Authority
Standardization: defining the standard name mappings used for each public institution when creating a new report
Transformation of the rules that define how a report is created, received from the Public Institutions into Oracle Business Rules
Deployment of the business rules to each Pro
Deployment of data entry forms and database mappings
XML + Rules + XSN
Report Y Template
Public Institution n
Creating the report templates (Infopath Template – xsn) and rules used for generating the report (how each field inside the report is computed from the standard data)
XML + Rules + XSN
Report X Template
Public Institution 1
Creating the report templates (Infopath Template – xsn) and rules used for generating the report (how each field inside the report is computed from the standard data)
Data Processing + XSN
Deployment of Business Rules
Data Entry Form
Database Mappings
Deployment for Data Entry Rules
XML + XSN
Auto Generated Report
Public Institution Adapter
Mapping or embedding the PrO internal, XMLrepresentation into a PI specific form
Central Management Unit
Scheduler
Logging
Job Manager
Business Rules Updater
Data Entry Updater
Alert module
Confirmation Unit
Report Sending Unit
Portal
Master Data
Transactional Data 1
Transactional Data 2
User & Role Administration
Data Warehouse
Master data
Transactional data
Business Rules Processing Unit
Business Proceses
Report Archivator
Report Archive
Enterprise Adapter
Mapping or embedding the PrO internal, XMLrepresentation into an enterprise specific form
PrO

Fig. 6.3. *The Architecture of the reporting PRO type e-System.*

7. Conclusions and future work. In this paper we presented a system designed to automate the data migration between various data sources. The solution is able to handle data transfers between different database instances. It is designed to work with dynamic data schemas, and requires minimal user input and interaction.

Organizations usually collect large volumes of data in their internal databases. In many cases, the stored data refers to the company structure, employees, customers and projects. With the development of new database systems, companies might want to change the database in use and transfer large amounts of data. Moreover changes in company structure and number of employees might generate modifications in the database schema. As a consequence, not only that the underlying database system is changed, but the data transfer would occur between two different schemas. In this context, the system presented in this paper can optimize the effort needed to correctly manage the migration process. As a case study, for example, we evaluated the solution in a real-world system implementation designed to automatically collect report data. The migration system leaded to the optimization of the overall performance of the reporting system by ensuring a transparent data retrieval process. It allows the user to easily manage the data mapping processes for filling the required reports.

In the future we plan to further extend the migration system with advanced mechanisms for data type recognition and conversions. We plan to design a solution where the Warehouse repository can be dynamically updated from a central repository of knowledge. Furthermore, in order to provide platform independence, the software will be ported to Java. The process might be lengthy as there is a significant difference between the facilities Java and C# provide as programming languages. A possible outcome of having two software products written in separate programming languages is that the capabilities offered by these languages with respect to database connections can result in a significant efficiency gain. That is to say both Java and C# have improved database connectors for certain types of databases. Due to this a user can choose the software product to use according to the types of the source and destination databases. Platform independence is a desired goal of the project. Another feature that should be implemented in the near future is support for migration interruption. There are cases when software is subject to failures and for a migration system this type of event is critical. Interrupting the migration system may result in data loss or data inconsistency. Therefore a module that takes care of any unwanted interruptions and helps restore the migration process from the point it was interrupted is necessary.

Acknowledgments. The research presented in this paper is supported by national project: "TRANSYS - Models and Techniques for Traffic Optimizing in Urban Environments", Contract No. 4/28.07.2010, Project CNCSIS-PN-II-RU-PD ID: 238. The work has been co-funded by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Romanian Ministry of Labour, Family and Social Protection through the Financial Agreement POSDRU/89/1.5/S/62557.

REFERENCES

[1] P. Howard, *Data Migration*, A White Paper by Bloor Research, Informatica, October 2007, Last accessed March 12, 2011, from http://vip.informatica.com/?elqPURLPage=552.

[2] M. Otey, and D. Otey, *Microsoft SQL Server 2005 developerś guide*, Microsoft, *McGraw-Hill Osborne Media*, 2005.

[3] Talend, *Talend Open Studio, User guide*, Last accessed June 29, 2010, from http://www.talend.com/index.php.

[4] J. Paul, *SQL Data Types*, Last accessed July 1, 2010, from http://onlamp.com/pub/a/onlamp/2001/09/ 13/aboutSQL.html.

[5] J. Morris, *Practical Data Migration*, British Computer Society Ltd, 2006.

[6] Microsoft, *Implementing Data Transfer Object in .NET with a Typed DataSet*, Last accessed July 4, 2010, from http://msdn.microsoft.com/en-us/library /ff650461.aspx.

[7] Talend, *Data Migration*, Last accessed March 13, 2011, from http://www.talend.com/solutions-data-integration/data-migration.php.

[8] M. Tungare, P. S. Pyla, M. Sampat, and M. A. Perez-Quinones, *Syncables: A Framework to Support Seamless Data Migration Across Multiple Platforms*, IEEE International Conference on Portable Information Devices (PORTABLE07), Orlando, FL, 2007, pp. 1–5.

[9] P. Allaire, J. Augat, J. Jose, and D. Merrill, *Reducing Costs and Risks for Data Migrations*, Hitachi, Last accessed February, 2010, from http://www.hds.com/assets/pdf/white-paper-reducing-costs-and-risks-for-data-migrations.pdf.

[10] M. Chatzidimitriou, and A. Koumpis, *Marketing One-stop e-Government Solutions: the European OneStopGov Project*, IAENG International Journal of Computer Science, 35:1 (2008).

[11] F. Pop, C. Dobre, D. Popescu, V. Ciobanu, and V. Cristea, *Digital Certificate Management for Document Workflows in E-Government Services Infrastructure*, in Proc. of The 9th IFIP WG 8.5 International Conference on Electronic Government (EGOV 2010) Lausanne, Switzerland, 2010, pp. 363–374.

[12] A. Marin, C. Dobre, D. Popescu, and V. Cristea, *E-System for Automatic Data Migration*, in Proc. of the 12th Intl. Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Timisoara, Romania, 2010, pp. 479–485.

Edited by: Dana Petcu and Alex Galis
Received: March 1, 2011
Accepted: March 31, 2011