Electronic Invoice Adapters

Rute Sofia Rodrigues Duarte
Create It, Lisboa. Portugal, rsrd@mega.ist.utl.pt

Abstract. The development of Information Systems has been led by the necessity of interconnection between the functionalities and information they manipulate, the biggest challenge being the integration of internal systems, which solve company specific problems, with external information systems of their partners, leading to a successful e-business.

Integration among companies is an important aspect in electronic invoice since it is based in the exchange of data between the different company applications. This project consists in the integration of the Information Systems, available at the e-market, through the construction of an adapter capable of sending and receiving electronic invoices in a transparent way to the ERP[1]'s of the involved companies. The studied mechanism will be the use of metadata, to describe distinct and specific application repositories to ease the integration between the different formats used by the invoice system users as well as new users. This metadata layer will be used as a data converter to a known pattern to allow data exchange with an invoice Broker.

Keywords: Legacy systems, Electronic Invoice, Business Process and Application Integration, Adapters, Metadata.

1 Introduction

The Enterprise Application Integration (EAI) has always played an essential role in the Information Systems development. Recently this role has increased due to, the need of integration of the different information systems among companies, and the emergence of the Service Oriented Architecture (SOA). The biggest challenge nowadays consists in the integration of the internal systems, which solve specific problems, with external information systems of their partners, leading to a successful e-business.

The Business to Business Integration (B2BI) is important in this study given that electronic invoice is based in the exchange of data between different company applications. Differences dwell in the method used for the data exchange, the data type, liability, etc.

The typical scenario to electronic invoice goes through acquisition to the ERP´s of a specific module to the safe e-documents exchange. This project consists in the integration of the Information Systems, available at the e-market, through the construction of an adapter capable of sending and receiving electronic invoices in a transparent way to the ERP's of the involved companies.

As a prototype an adapter was developed capable of exchanging messages between the ERP in the national market and an invoice Broker, without having to perform alterations in the ERP.

The studied mechanism is the use of metadata, (rules description, data structure and applicable restrictions – data related with API[2]'s and ERP's) to describe distinct and specific application repositories, so that the construction of the metadata layer can ease the integration between the different formats used by the invoice system users as well as new users integration. For each business application, the metadata define the business entities with which the business application interacts and its available methods. The metadata is characterized using XML and stored in a metadata repository.

The metadata layer will be used to convert the data to an outlined XML schema to the data exchange with the invoice Broker. The WFC technology for the Web Services development will be used for the communication with the invoice Broker. The prototype will be developed using *Microsoft .Net 3.0* technologies and the ERP used in the integration is the *Primavera Express* from *Primavera Business Solutions*.

[1] ERP – *Enterprise Resource Planning*

[2] API – is a source code interface that a computer application, operating system or library provides to support requests for services to be made of it by a computer program.

2 State of the art

The productivity growth in the company's finance and administrative areas depends on the capacity of removing time-consuming and low value added tasks and fomenting the use of tools that allow the process automation and digitalization.

In these days, the majority of time and financial resources used by companies is related with the invoice processing, this method involves a group of manual tasks, from the insertion of data in the system to the invoices and documents imprint and mailing. This problem increases with the company dimension, since big companies deal daily with the emission and reception of hundreds of documents, which reflects in high expenses related with paper, processing, imprint, envelopes and mail taxes.

The use of electronic document transaction solutions is, therefore, every day more important since it allows the invoice process digitalization and automation, promoting the company's productivity, efficiency and competitivity.

By adopting the electronic invoice, companies will, not only promote the process simplification and co-workers productivity, but also invest in innovation and business competitivity. As referred in [14], the electronic invoice is a commercial document that resembles the conventional invoice but in an electronic format, enclosing the same fiscal/legal value that the paper version, as long as it contains the necessary fields to any invoice, and satisfies the legal conditions. The main purpose is to grant the authenticity of the document origin and its contents integrity.

2.1 Information Systems Integration

The integration of Information Systems is an obstacle to many businesses, as supply chain partners consist of independent systems that in some cases cannot communicate one another. Recently this situation has been aggravated as due to the necessity of integration of the information systems both inside the business and between different businesses. [15]

Integration of the different Information Systems inside the company is crucial to take advantage of e-business. The method implemented by many Portuguese companies went through the integration of the existing systems through ERP's. It was assumed that the problem might be solved by replacing all the Information Systems by one ERP.

The company's internal ERP's are frequently implemented according to the data pattern, business processes and logic. The failure of the system to achieve its goals can be due to the fact that ERP do not cover all the information technology requirements, and do not meet the business processes. [13]

As companies started installing disparate information systems the integration among them became more difficult. [15] The top priorities of companies these days is the exchange of data and sharing of business processes across multiple trading partners. This is called B2B Integration. [4]

Although there are many kinds of integration- interface, data method – all are based on exchanging data between two Information Systems, The differences reside on how this data exchange occurs, what kind of data is exchanged, which guarantees are offered, and so on. [11] This led to the development of a Enterprise Application Integration (EAI).

There are two major approaches for legacy systems integration: application integration and data integration. [12] [5].

1. Application Integration: In this approach applications contain the business logic of the enterprise, and the solution lies in preserving that business logic by extending the application's interfaces to interoperate with others or sometimes newer applications. The previously used solutions like the *User Interface Modernization*, *Peer-to-peer integration*, *message routers*, *Common Object Request Broker Architecture* (CORBA) and *Common Gateway Interface* are overage and obsolete. Recently, the more adopted solution is the *Service Oriented Architecture* (SOA), which takes advantage of some integration technologies like the *Web Services*. As referenced in [6], *Web Services* offers a set of standards based on the Internet, very simple and popular, for information systems and applications integration.

2. Data Integration: In this approach the real currency of the enterprise is its data. The implied business logic in the data and metadata can be easily manipulated directly by applications in the new architecture of the enterprise. Some of data integration solutions are XML Integration and Data Replication, however, the XML Integration has the disadvantage of only resolve the syntactic problem. Actually, the biggest problem in Data Integration resides on the semantic aspect, however,

is being developed technologies like *Semantic Web Services* [7] and ontology definition languages, such as OWL [8], for achieve one solution.

2.2 Documents Format

As mentioned above, the data structure is an important matter in the information system interconnection.
One of the requisites of e- business – particularly in the electronic invoice – is that the data is structured in order to ease data exchange.
At first the e-market used technologies like Electronic Data Interchange (EDI) or Electronic Data Interchange For Administration, Commerce, and Transport (EDIFACT). However, the costs to implement and maintain the use of these technologies became very high since, for example, excusive lines for the system were needed. This aspect was a setback for the integration of Small and Medium Enterprises (SME's) as they couldn't support such expenses.
The use of technologies like EDI [1][14], XML [1][3][10], ebXML [1], among others, in the exchange of structured messages between companies in a standardized way, is an important issue to the acceptance of best practices in electronic invoice. The level of access to information that Internet allows nowadays is also a very important aspect.
As referenced in [6], XML is assumed by many as the elected language for formatting messages. By the use of XML, a company is able to create new messages formats for orders, invoices or any other document type. This flexibility allows companies to adjust the messages formats to fit in their business particularities. In the other hand, this solution forces the use of tools and/or converters for introducing these messages in the company information system and/or retrieving them.

2.3 Adapters

Adapters or connectors are pieces of software that are used in application integration and act as intermediaries to access applications that weren't developed for that purpose, such as legacy systems.
Nowadays, and concerning companies integration, the used adapters are becoming "smarter", which makes the event capture, when executing intermediary systems, easier. These intelligent adapters are assigned to many of the dynamic tasks necessary to the application integration, like code updates, when another application is changed. Instead of being done by the programmer, these changes are all made by the adapter itself.
As referred in [2] there are two kinds of adapters referring *Message Brokers*: *Thin Adapters* and *Thick Adapters*. These adapters can act in a dynamic or static way. *Thin Adapters* are mostly API's that map the system's interface to a common interface supported by the *Message Broker*. *Thick Adapters* expose common software and functionalities to be used by the *Message Broker* infrastructure and the source/destination application. They also make it easier to manage the information flow and the process invocation, due to its abstraction layer. The static adapters must be implemented manually according to the requirements and specificity the systems used by them. They do not have mechanisms to comprehend the relational schema of the databases, which results in a manual configuration to receive the information of the data original schema. On the other hand, the dynamic Adapters, also known as intelligent adapters, are used to deal with the dynamic tasks involved in application integration. They have the ability to learn about the systems they interact with, through the reading of the information acquired in the initial connection to the application or database.

3 Basic Information Systems

The ERP used in this study was the *Primavera Express*. This system was built taking in consideration some important aspects to small companies, with a very low number of employees. Built to the commerce and services markets, and based in the more evolved versions of the product, this software allows managing sales e invoicing, as well as Customer Relationship Management (CRM) and Stock Management. However, it's a limited product in terms of functionality, because it is mainly meant to give the possibility to manage some essential business operations, to small and newly created companies. [17]
At the technological level, *Primavera Express* has a big advantage: it was built over the same technological base of the other, more advanced products, directed to other market segments and types of companies. This allows that, in the future, if a company wants to migrate to a more evolved system of the *Primavera* line, it can do this with virtually no effort. It uses, as a Database Management System (DBMS) the *Microsoft SQL Server* (the *Primavera Express* uses the *Microsoft Desktop Engine / Microsoft SQL Server Express*).

The *Primavera Express* ERP is commercial management software that allows the management of all the business operations of a company, including invoicing, through the use of recent technology, and without any cost. By releasing the new *Electronic Transactions* module, this ERP gives the possibility to change the invoicing method of the company to electronic invoicing. With this initiative, *Primavera Express* aims to equip companies with the most advanced technologies, promoting the innovation of the company itself, at the processing level. All these offers allow the company to use this ERP, independently of the market segment in which they are included. This was one of the main reasons for the usage of the *Primavera Express* software in this study. [17]

3.1 Tables

There were many possibilities of integration with other systems, however the choice was to interact directly with the *Primavera Express* database. The involved tables on the electronic mediation scenario, used by the ERP *Primavera Express*, are represented in **Figure 1** corresponding, respectively, to the tables used for sales and to the tables used for purchases.

The information obtained from the Selling tables is converted and sent to the client, and the information to insert in the Buying tables is obtained from the document sent by the supplier.

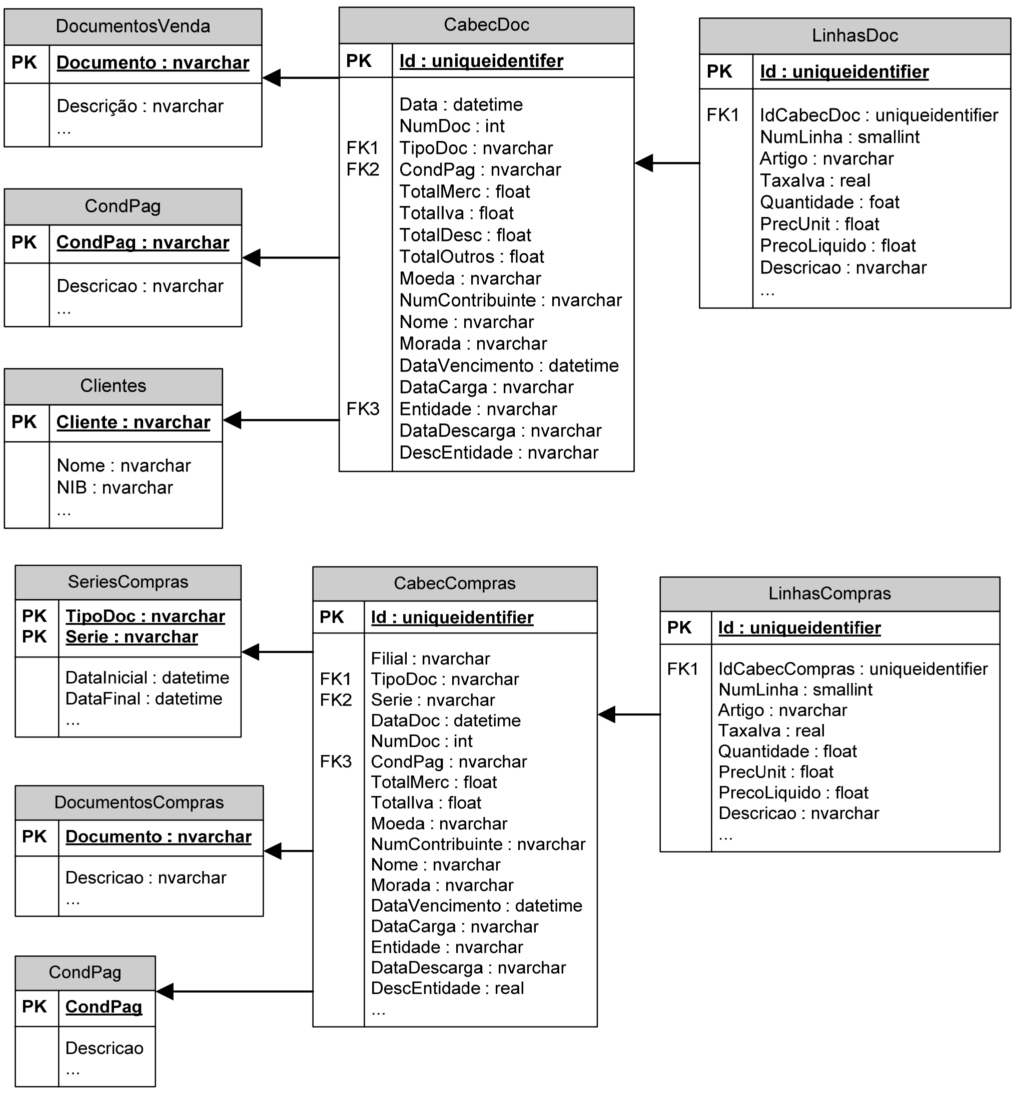

Figure 1 – Buying and Selling Relational Diagram

4 Metadata Repository

This adapter is used mainly to ease the access to legacy systems, converting its information to a canonic format, and by this allowing the communication among various systems. This conversion is defined using XML configuration files, which allows to easily integrating another ERP without the need of additional programming.

The purpose of this adapter is to be a solution to the problems that arise when a company wants to exchange messages between different systems, that can't usually do it in a standardized way. To solve this problem, it's necessary to define a single format to the messages, and the systems only need to convert from the relational format to this unique format. The adapter not only solves this problem but also, solves the biggest issue when dealing with message conversion: the need to program the modules that will convert to and from the canonic format. The adapter will execute these operations automatically, easing the work of the developer. This is achieved using a metadata repository that contains information related with the connections to the system's databases, as well as the operations, parameters a output results expected in the coding and decoding process of data to be written in the database.

This metadata repository contains information regarding the ERP systems, including the data format that it uses, in terms of entities and methods, and is written by a programmer that has know-how about the ERP system itself.

The integrator element between the adapter and the ERP System is a *SQL Server* database and, consequently, the language used to code it is SQL.

Taking in consideration that the technological architecture of the company, as well as the integration between the different systems, is very important issue to the system's "big picture", the metadata repository assumes a very important role. In the end, the metadata could contain information about all the applications and information systems that exists in the company, becoming a real corporate repository for information.

In conclusion, the adapter allows a standardized execution method, capable of reading the information stored in the metadata repository, and obtaining the information that is to be inserted in the ERP system. This information is converted to a canonic and standardized format, defined by the systems involved, and used to build messages that are to be exchanged among those systems.

4.1 Functionalities

The metadata information has two main purposes: describe the system's API, and make that API easy to use by external systems. Using a metadata model, the systems will be able to integrate the information from multiple sources, using the configurations stored in the repository, and converting them to the desired format.

The configuration components that exist in the metadata model are shown in Figure 2.

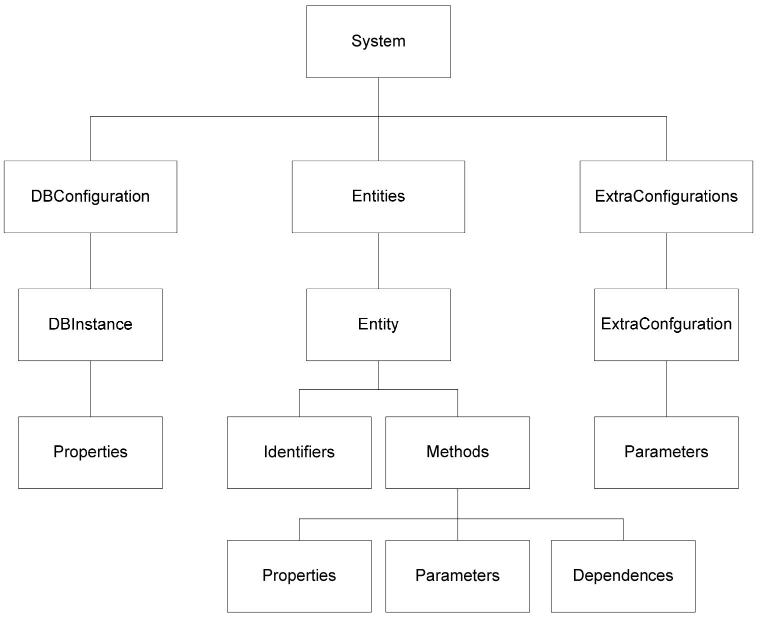

Figure 2 – Metadata Repository Schema

The *DBConfiguration* component corresponds to the parameters that are needed when communicating with the database, namely the database server, the database name and the login and password information.

The *Entity* component is related to the set of configurations necessary to obtain or insert information from the system. Each *Entity* contains at least one *Method* that allows the insertion of a set of possible system operations. In each *Method*, it is necessary to describe the entry parameters and the output results. Depending on the purpose of the *Method*, it is possible to specify only the entry parameters (if the objective is to insert information in the system) or the output results (when the method doesn't require any entry parameters to be executed). It is also possible to specify necessary data type conversions, using the *Conversions* component to describe the data type returned by the system and the data type to which we want to convert. This is extremely useful when integrating with other systems that use different data types. Another important component is the *Dependences* that describe the dependency relationships between tables. This way, it is possible to deal with situations when the main table, that contains the information, has some foreign key relationships to other tables. By specifying these cases using the *Dependences* component, the foreign key value will be replaced by the corresponding value on the foreign key's table.

The *ExtraConfiguration* component is related to all the configuration information that is necessary to some important operations that run in background. In this study, it's used to know specifically how to obtain the necessary attributes to find documents recently inserted in the system.

5 Integration Technologic Solutions

A possible usage scenario to this adapter is shown in Figure 3. In this example, there is a *Message Broker* that connects the different ERP systems. The *Message Broker* is used to support business process modeling and is responsible by the communication and secure routing of documents exchanged by two or more ERP systems. On the other hand, the *Invoice Adapter* is a component that exists in each system, where all the aspects related to the integration and conversion of data are dealt with, in order to make it possible to exchange documents between the different legacy systems. This way, instead of a point-to-point connection between a system and all the others, each ERP only needs to be connected to the *Message Broker*.

As shown in Figure 2, each ERP system has one extra-application, the *Invoice Adapter* that is parameterized and configured in accordance to the system to which is associated. Another relevant aspect is the usage of the WCF Framework between the *Message Broker* and the ERP System. The choice to use WCF was due to the fact that it is a unified programming framework used to rapidly build safe, reliable and service-oriented applications, that support communication standards like Ws-Security, Ws-Trust, Ws-Addressing, and others.

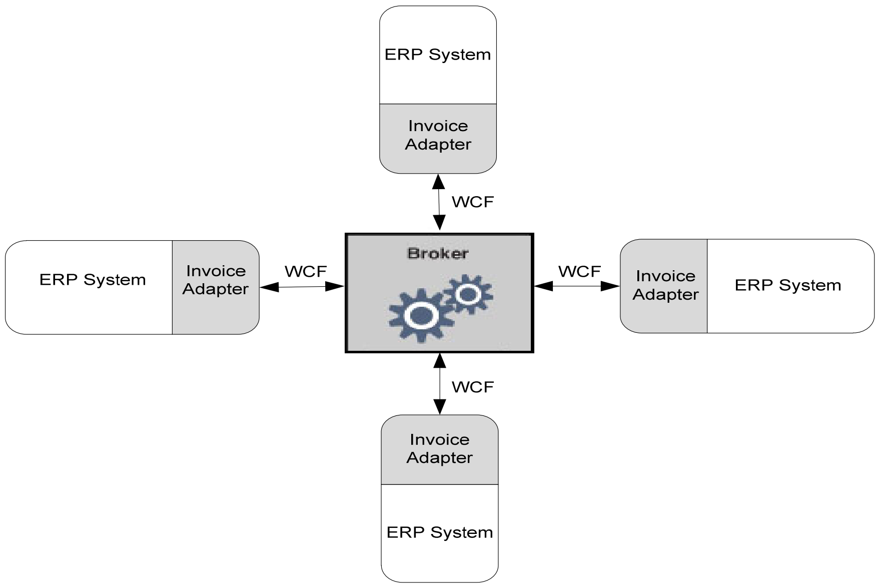

Figure 3 - Invoice Adapter Usage Scenery

The Invoice Adapter has three fundamental layers: the Communication Layer, the Message Creation / Reading Layer and the Database Communication and Data Conversion Layer. In Figure 4, it is possible to see how these layers relate to one another.

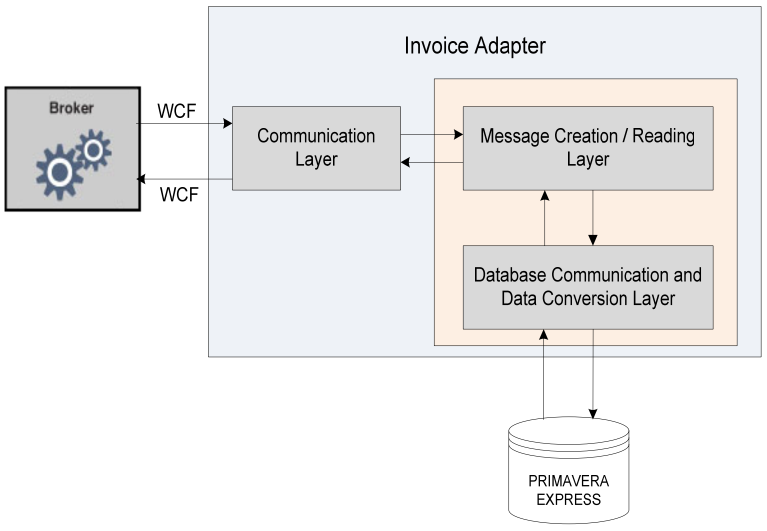
Invoice Adapter
Broker
WCF
WCF
Communication Layer
Message Creation / Reading Layer
Database Communication and Data Conversion Layer
PRIMAVERA EXPRESS

Figure 4 –Invoice Adapter Overview

The **Communication Layer** is responsible for the communication with the *Broker* and also for the management of all the messages that were received, and that are to be sent, to the *Message Broker*. As mentioned before, the WCF framework is used in the communication and message management, through the use of the MSMQ technology to store, in message queues, the messages received and the messages to send.

The **Message Creation / Reading Layer** is the component responsible to deal with the messages that go through the adapter, as well as to manage the events related to the reception and sending of messages. This layer is also the one that will read the information in the metadata repository and send information to the Database Communication & Data Conversion Layer.

Finally, the **Database Communication and Data Conversion Layer** is the component that will deal with the data conversion, to the canonic format and to the relational format or the ERP's database. This layer is also responsible for reading the information, from the database, that is to be sent, and for inserting new data in the database, received from another ERP System.

5.1 Communication Layer

This module is like a separated layer from the rest of the modules. As shown in Figure 5, this layer is composed by four *Windows Services* that are always executing, in order to manage the communications. Using a Design Pattern's naming convention, each one of these services implements an *Observer*.

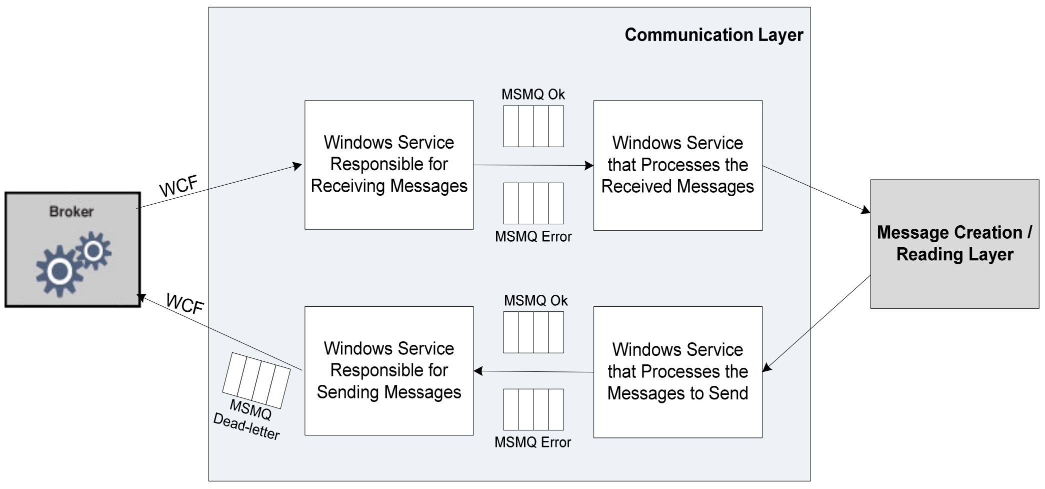
Communication Layer
Broker
WCF
WCF
Windows Service Responsible for Receiving Messages
MSMQ Ok
MSMQ Error
Windows Service that Processes the Received Messages
Message Creation / Reading Layer
Windows Service Responsible for Sending Messages
MSMQ Ok
MSMQ Error
Windows Service that Processes the Messages to Send
MSMQ Dead-letter

Figure 5 – Communication Layer Arquitechture

The Windows Service responsible for receiving messages deals with the messages sent by the *Message Broker*, through the WCF. This service implements a WCF Server, and is always waiting for new messages to be inserted in the ERP System. On the other hand, the Windows Service responsible for sending messages will send messages to the *Message Broker*, also using WCF. This service will invoke the services exposed by the WCF Server on the *Message Broker* to send new messages, as soon as they are ready to be sent. Both services use the message queues (MSMQ) to store the messages received or to obtain the messages to be sent through WCF.

The Windows Service that processes the received messages obtains the messages stored in the message queue by the Message Reception Service, and generates events that will lead to actions in the Message Creation / Reading Layer. The message will then be read and stored in a generic object in memory that will be used later by the Database Communication and Data Conversion Layer, to convert the data from the canonic format to the relational format, and consequently write it to the database.

The Windows Service that processes the messages to be sent is responsible for the detection of new documents in the ERP System’s database, that are ready to be sent to the *Message Broker*. First, this service accesses the ERP’s database to obtain information about the most recent documents. This access is made by invoking a method in the Message Creation / Reading Layer that deals with all the necessary operations to obtain the most recent documents. After receiving the list of the documents to be sent, they are stored in the message queue, that is read later by the service responsible by sending the messages. This service will then send the messages to the *Message Broker*.

5.2 Message Creation/Reading Layer

This layer creates messages to send to the Communication Layer and reads the messages received by the same layer. As shown in Figure 6, this layer is also responsible for managing the operations to be executed in the Database Communication and Data Conversion Layer, in case of database readings to obtain a message build registry and in cases of writing information read from messages to the database.

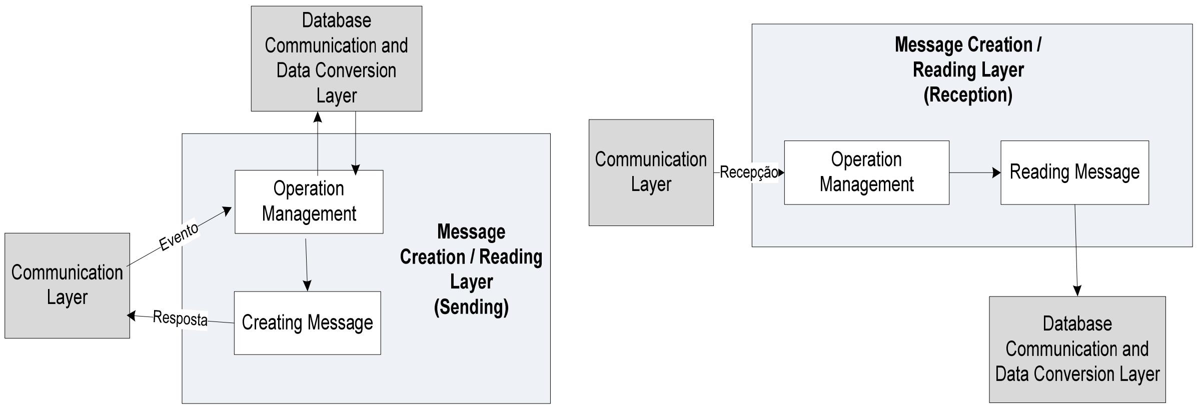
Database Communication and Data Conversion Layer
Operation Management
Message Creation / Reading Layer (Sending)
Communication Layer
Evento
Resposta
Creating Message
Message Creation / Reading Layer (Reception)
Communication Layer
Recepção
Operation Management
Reading Message
Database Communication and Data Conversion Layer

Figure 6 – Message Creation / Reading Layer Architecture

The content of the messages is stored in the object obtained through the desserialization process of the XML document provided by the WCF framework. This way, and in the context of this study, the *Message* object contains all the information about an electronic invoice. The way how the information is stored in a generic object will be different depending on the situation.

This layer is responsible for many of the operations when executing the *Invoice Adapter*, like verifying and obtaining new documents from the ERP system, reading and writing messages as explained before, and also by invoking the methods, of the Database Communication and Data Conversion Layer, responsible by inserting and reading information from the ERP system.

5.3 Database Communication and Data Conversion Layer

This layer is responsible for the communication with the persistence component of an ERP System, a legacy database. Generic and complex, this layer will essentially be used to integrate data, supporting many operations like:

- Extract information from a database, transform or convert that information in a canonic format, load the information in a generic object to be used later by the Message Creation / Reading Layer, as shown in Figure 7.
- Obtain the information from a generic object, transform or convert that information in a relational format, load the information in a database as shown in Figure 8.

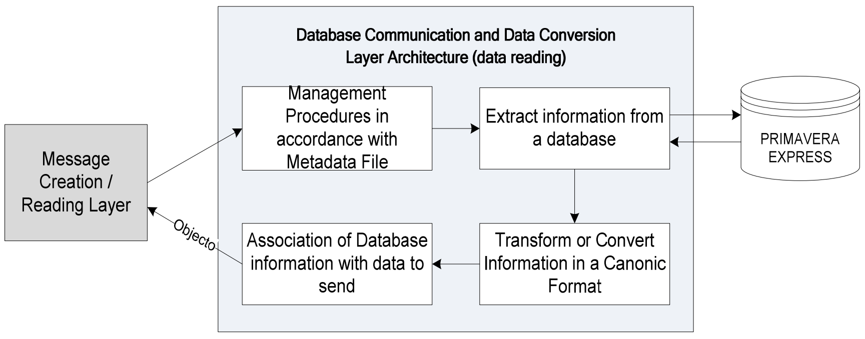
Database Communication and Data Conversion Layer Architecture (data reading)
Message Creation / Reading Layer
Management Procedures in accordance with Metadata File
Extract information from a database
PRIMAVERA EXPRESS
Objecto
Association of Database information with data to send
Transform or Convert Information in a Canonic Format

Figure 7 – Database Communication and Data Conversion Layer Architecture (data reading).

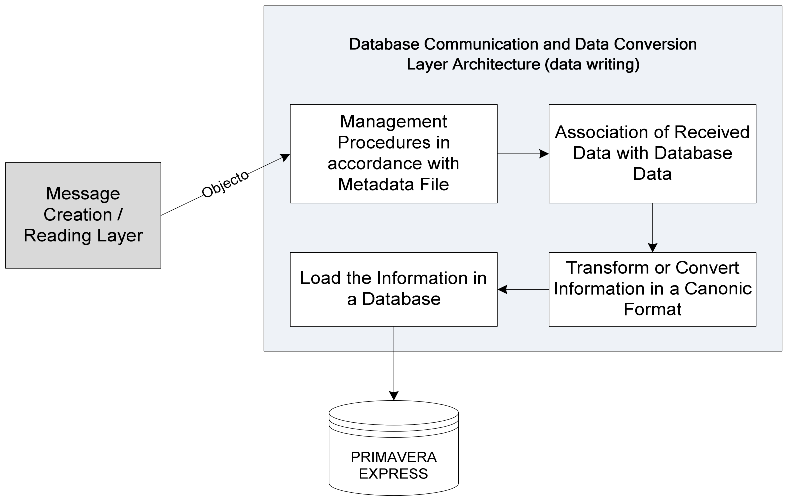
Database Communication and Data Conversion Layer Architecture (data writing)
Message Creation / Reading Layer
Objecto
Management Procedures in accordance with Metadata File
Association of Received Data with Database Data
Load the Information in a Database
Transform or Convert Information in a Canonic Format
PRIMAVERA EXPRESS

Figure 8 - Database Communication and Data Conversion Layer Architecture (data writing).

The database access is made in an efficient way that can be adapted to any type of database, in any ERP system, specified in the metadata repository, as referred in Chapter 4.

5.3.1 Data Base Generic Reading and Writing

The access to the ERP System's database is made using generic objects that contain information from the metadata repository. The parameters that are present in the configuration file, related with the connecting, obtaining and writing information from the database, are used by this layer to make the access generic, and without the need of additional programming in case a new database is used. The process of accessing the database is not visible to the user, and is executed automatically according to the event that made it happen (new messages from the *Message Broker* or new documents in the database).

5.3.2 Data Conversion

The conversion or transformation of information between the two formats, the relational format and the canonic format (defined by all the ERP systems involved) is one of the most important aspects of this study, because it's the proposed solution to incompatibility problems between different systems.

The adopted solution is based on the usage of elements and attributes that exist on the metadata repository, like the *Conversion* element. It is performed a comparison between the data format in database and the data format in the received message from *Message Broker*. If they are different, is made a conversion of the relational format for the standard format, respectively.

The *Conversion* element, on the other hand, will be used to extra conversions and also before the conversion to the canonic format. This means that situations where it is necessary to transform a field in a certain type, before converting it to the final format, can happen.

These types of conversions are frequently present in EAI and ESB products (like the *Message Broker* with which the adapter interacts). Usually, there are *mappers* that allow the visual specification of the conversions to be made. In the adapter case, the desired functionality is simple and *lightweight*, and it is not necessary to use one of the products mentioned.

6 Conclusions

This study described an architecture and a methodology for the development of an integration adapter that should be used in electronic mediation scenarios.

The objectives proposed initially were fulfilled, and a version of the adapter was produced that allows the integration of different ERP systems in electronic mediation scenarios. The adapter receives electronic documents in a canonic format from the *Message Broker*, reads them, processes them (converting the data) and finally inserts them in the ERP's database. All the processing related to the message is done according to the relational model specified in the configuration file (metadata repository). On the other hand, the adapter also reads information from the database, processes and converts the information, and creates an electronic document that is to be sent to the *Message Broker*.

After a careful analysis of the state-of-the-art, and of the results obtained with this study, we can conclude that this adapter has the following advantages:

- Motivates the integration of different ERP Systems.
- Abstraction, to the users, of how the adapter works internally.
- Automatic execution of Windows Services.
- Usage of message queues to temporary store messages to be sent or messages that were received, solving the problem of message management.
- Secure communication, using the WS-Security protocol of WCF.
- Usage of digital signatures and certificates to ensure the authentication and confidentiality.

The following disadvantages were also identified:

- Relatively low performance.
- Some restrictions in the internal integration level with the ERP System, considering only integration with databases.

A proposal for future work to be developed is the implementation of code blocks that send and receive confirmation and error messages, which reflect the result of the message processing in the client and *Message Broker*, and also the result of the communication.

It would be interesting if one made a more advanced approach, when considering the Database Communication Layer, which is responsible for reading and writing information on the database, and also for the conversion between de data formats. Also related to the communication with the database, some methods should be redesigned to allow connections to other databases, not only *Microsoft SQL Server*. This layer is the most important of the adapter, because it makes it possible to exchange messages between systems with different data formats.

At the electronic invoice level, there are some improvements that can be made related to legal aspects, like the existence of an external certification entity, responsible by emitting and distributing digital certificates, making the certificate exchange more secure.

7 References

[1] P. Marques, Troca de Informação de Negócio para Negócio – Do EDI ao XML/EDI e EBXML, Universidade Fernando Pessoa, 2003.

[2] D. Linthicum, B2B Application Integration – E-Business – Enable Your Enterprise, Addison-Wesley Information Technology Series, 2001.

[3] F. Cummins, Enterprise Integration – An architecture for enterprise application and systems integration, John Wiley & Sons, 2000.

[4] G. Samtani, M. Healey, S. Samtani, B2B Integration: A Practical Guide to Collaborative E-Commerce, Imperial College Press, 1992.

[5] I. Gorton e A. Liu, “Architectures and Technologies for Enterprise Application Integration”. Em Proceedings of 26th International Conference on Software Engineering, 2004.

[6] M. M. Silva, Integração de Sistemas de Informação, FCA – Editora de Informática, 2003.

[7] Martin Hepp: Ontologies: State of the Art, Business Potential, and Grand Challenges, Hepp/De Leenheer/de Moor/Sure. (Eds.): Ontology Management: Semantic Web, Semantic Web Services, and Business Applications, ISBN 978-0-387-69899-1, pp. 3-22, Springer, 2007.

[8] S. Bechhofer, F. Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneider, L. Stein, “OWL Web Ontology Language”, W3C Proposed Recommendation, 2003.

[9] K. Qureshi, “Enterprises Application Integration”. Em Proceedings of International Conference on Emerging Technologies, pp. 340-345, 2005.

[10] H. M. Sneed, “Using XML to integrate existing software systems into the Web”, Em Proceedings of Computer Software and Applications Conference, pp. 167-172, 2002.

[11] A. Vasconcelos, M.M. da Silva, A. Fernandes, J. Tribolet, An Information System Architectural Framework for Enterprise Application Integration. Em Proceedings of 37th Hawaii Conference on System Sciences, 2004.

[12] M. Chowdhury e M. Zafar, Integration of Legacy Systems in Software Architecture. Em Proceedings of Workshop at SIGSOFT, pp. 110-104, 2004.

[13] J. Lee, K. Siau., and S. Hong, Enterprise integration with ERP and EAI. Commun. of the ACM, Vol. 46, No. 2, pp. 54-60, 2003.

[14] UMIC, Guia da Factura Electrónica, 2006.

[15] M Themistocleus, Z. Irani, P. Love, Enterprise Application Integration: An Emerging Technology for Integrating ERP and Supply Chains. Em Proceedings of European Conference on Information Systems, 2002.

[16] S. Dorda, K. Wallnau, R. Seacord, J. Robert, A Survey of Legacy System Modernization Approaches, CMU/SEI-2000-TN-003, 2000

[17] Primavera Express
http://www.primaverabss.com/pt/PortalRender.aspx?PageID={906727d5-b773-491d-b5e6-5d089089d110}