eXtensible Business Reporting Language

ESPAÑA

White Paper
Technology Working Group 2005

Translated by Allyson Ugarte

Member of Spanish Association for Accounting and Business Administration (AECA)

Asociación Española de Contabilidad y Administración de Empresas

TABLE OF CONTENTS

Foreword

Technological advances in recent decades have given us a new digital platform for financial information. When we say "digital", we do not mean "compatible" because each software application generates its own output document (balance sheet, income statement, etc.), which unfortunately cannot be read directly by a different application. Why not? Because each application stores its data in its own proprietary format.

Traditionally, this problem has been resolved by exporting the file in an ASCII format (a text file with each data field separated from the next by a delimiting character, such as a comma) so that it is possible to retrieve the original ASCII file into another application. However, this solution is very time-consuming because it is often necessary to modify the ASCII file when the exporting application does not use the same delimiting character as the importing application. As a result, this may cause the data to be skewed and placed in the wrong table. Another solution is to setup a specific application to convert the data between two applications, but the best solution is to use a standard to exchange data.

The need for a digital standard to exchange accounting information among software applications is even greater if you want to get data points from many financial statements published in various formats (pdf, xls, html, doc, etc.). Today, this standard is known as XBRL, which is widely accepted by the international accounting community. It was developed by XBRL.org, an international consortium of companies and organisations, and is sponsored by the AICPA (*American Institute of Certified Public Accountants*). Member organisations include large accounting and consulting firms, and institutions such as the IASB (*International Accounting Standards Board*), the IMA (*Institute of Management Accountants*), the CICA (*Canadian Institute of Chartered Accountants*), and the ICAEW (*Institute of Chartered Accountants* in *England and Wales*).

In 2001, XBRL was introduced in Spain by the Spanish Association for Accounting and Business Administration (AECA) (*Asociación Española de Contabilidad y Administración de Empresas*) through their Committee for New Technologies in Accounting. AECA began to study the standard and its possible use in Spain, and organised a working group composed of the *Central de Balances* Department at the Bank of Spain, Informa, the Spanish Institute of Certified Auditors (*Instituto de Auditores Censores Jurados de Cuentas*), Navision (now, Microsoft), PriceWaterhouseCoopers, and the University of Huelva. In February 2002, Spain was recognized as a provisional jurisdiction of XBRL International, and attained permanent jurisdiction status in April 2004. At that time, the XBRL Spain was formally constituted, under the presidency of the Bank of Spain and AECA, to be the main facilitator for the outreach program and the development of the XBRL standard in Spain.

Thanks to the establishment of a permanent jurisdiction in Spain, XBRL initiatives have been consolidated, and the first national taxonomies have been developed: (1) DGI Taxonomy (*Datos generales de identificación)* for elements relating to an entity's identification in financial reports used by several official agencies, such as the *CNMV* for the stock market, the *Central de Balances* at the Bank of Spain, the *Registros Mercantiles* for the official business registration sites, and other agencies that collect or distribute financial information;

(2) IPP Taxonomy (*Información pública periódica*) for elements relating to public and periodic information submitted by publicly traded companies to the *CNMV* for stock market information; (3) PGC90 Taxonomy (*Plan General de Contabilidad de 1990*) for elements relating to the submissions of annual reports in accordance with required national accounting standards; (4) other taxonomies being coordinated by the Bank of Spain, such as reports required from commercial firms for statistical and tax purposes; financial information required from financial institutions and a system to exchange such financial data; COREP (Common Reporting) for the Bank Supervisors of the European Union; and SEPLAC (*Servicio Ejecutivo de la Comision para la Prevencion del Blanqueo de Capitales e Infracciones Monetarias*) for the prevention of money laundering and other illegal monetary transactions.

This White Paper has been compiled by the Technology Working Group of XBRL Spain and covers the main aspects of the XBRL standard in a clear and concise manner. The main concepts are outlined so that the average reader interested in XBRL can understand the scope and implications of XBRL, such as Specification 2.1, creation and exchange of XBRL information, taxonomies, error control and validation of instance documents, transmission, reception and storage of XBRL reports, security, versioning, and products and services.

The purpose of this White Paper is to be a reference document, the first of its kind in Spanish, and required reading not only for professionals who are responsible for the adoption and implementation of XBRL in their companies or organisations, but also for those readers who want to understand the fundamentals of this international standard for exchanging and comparing ever-increasing financial data from around the globe.

I would like to close this foreword by giving recognition and gratitude to the work of the members of the Technology Working Group of XBRL Spain, who have made it possible to make this document a reality. To each and every one of them, thank you very much.

Huelva, Spain, June 2005

Professor Enrique Bonsón Ponte
Chair of Finance, Economics and Accounting Department, University of Huelva, Spain
Vice-President of XBRL Spain

1 Introduction

Adoption of the XBRL standard for the presentation of business and financial information always requires some effort on behalf of the organisation interested in integrating a new technology into their existing information systems. The work involved may vary depending on the level of integration of XBRL, from preparing a simple report using a taxonomy created by another entity, to a total restructuring of the business process and systems, including the development, approval, and publication of taxonomies.

Those organisations already accustomed to working with XML are not going to see XBRL as especially new, since it is simply a language built on XML. However, other organisations will surely appreciate a reference document like this one as a first point of contact with the XBRL standard. This White Paper is the first study to be published by the Technology Working Group of XBRL Spain, and it is their intention to address the basic questions that future users of this open-source standard will need to understand in order to exchange financial data and reports.

Chapter 3 of this White Paper[1] introduces XBRL as a common, global language to label data used in business reporting worldwide. The basic concepts of XML and XBRL are covered in this chapter, and differences between them are pointed out.

Chapter 4 offers a closer look at XBRL and the elements that comprise a taxonomy and those that comprise an XBRL instance document. Also covered are the various possibilities to send, receive, store, and retrieve XBRL reports, and a functional and technical architecture is outlined to serve as a reference for the adoption of XBRL. It is absolutely essential is to have control over the creation and maintenance of a taxonomy, so here you will find suggested best practices for versioning. You will also find information on the performance of XBRL applications, and recommendations are made to optimize performance. Although security is somewhat outside the scope of the XBRL specification, security is considered to be very important and this chapter outlines those aspects relating to the generation, storage, and transmission of XBRL documents.

Chapter 5 offers an overview of vendor solutions and tools currently available in the marketplace, both for the creation of taxonomies and the generation and management of XBRL reports, each with their individual features and deliverables.

Chapter 6 covers the area of training, but only to give some basic ideas on the minimum knowledge that technicial IT staff should have when carrying out a project with XBRL. This chapter also references a document entitled “Training and Best Practices”, prepared by the Working Group for Training and Development of XBRL Spain, which goes into depth about training requirements and how to develop this type of project

Finally, Chapter 7 contains some examples of XBRL implementations that may serve as a reference for those who plan on adopting this standard.

[1] The Web version of this White Paper can be found at www.xbrl.org.es

2 Objectives

The purpose of this White Paper is to provide a first look at the XBRL standard for anyone interested in this new technology, with the objective of mitigating the risk of the unknown, and offering some basic information and useful recommendations for any type of organisation that plans to use XBRL and integrate it into their systems. Technicians will find this information helpful if they plan to start an XBRL project of any kind, as the technical references cover the type of subject matter that they will need to know during implementation.

During the preparation of this document, there was considerable concern to make the document as complete as possible, covering the basic concepts used in the world of XBRL and making it easy to read, even for those not familiar with the material. The authors want to avoid being too technical in the subjects discussed, and that is why many references are made to other sources and websites where the reader may learn more about the topic and go into as much depth as desired.

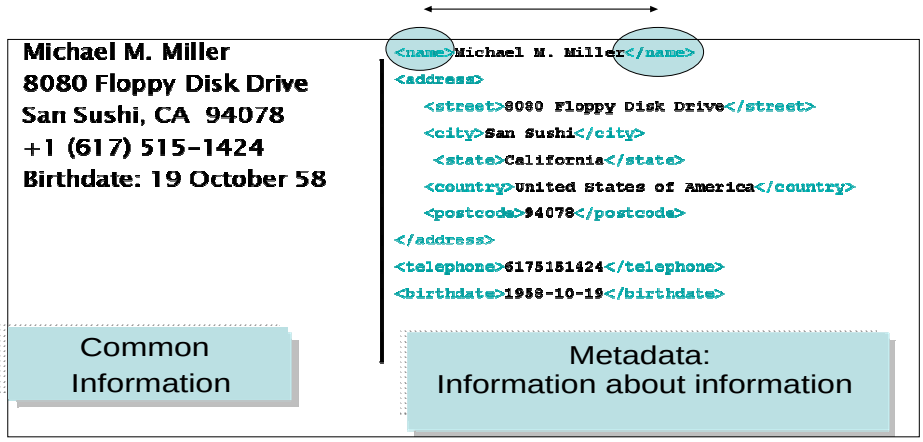
3 Basic concepts of XML and XBRL

3.1 XML is a global language

XML, the acronym for eXtensible Markup Language, is a common, global markup language and standard, as defined by the World Wide Web Consortium (W3C), for the formatting of labelled or “tagged” information.

The formatting of XML labels adds meaning to information that is commonly exchanged between users in the form of metadata, which is readily understood by software applications.

Example 1: Common information is placed between labels for more meaning.



As we can observe in the example above, a software application that is capable of interpreting the metadata in the label <name> can "understand" the name of the person "Michael M. Miller" and can distinguish it from the address <address>, and this feature allows it to automate the actual processing of data.

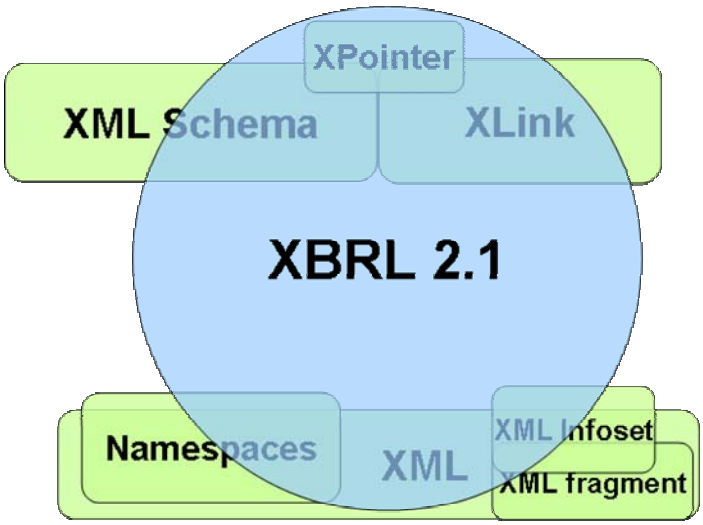
Since its creation in 1998, XML has served as a base platform for the development of other computer languages:

- For the exchange and extraction of information: SOAP, WSDL, XQuery, XPath, SAX, DOM
- For specific business "vocabularies" or "dictionaries": MathML, MusicML, OTA, HL7, XBRL
- For the formatting or presentation of information: XHTML, XForms, WML, SVG
- For using and transforming XML itself: XSLT, XSL-FO, XML-Schema, RelaxNG, XLink, XPointer

3.2 The need for a standard business reporting language

- When the stage was set to describe a business reporting language, the following requirements were considered necessary for its syntax:
- It should be based on a common and open-source format: eXtensible Markup Language, XML
- The definitions of the metadata to be exchanged should be standard definitions, i.e., a term like "Cash Deposits in Central Bank" should mean the same in any software application that uses this term. This is the central pillar that supports all taxonomies, which are in reality common data dictionaries expressed in the XBRL language.
- Another requirement is that these taxonomies should be easily extensible, which means that different industry sectors, companies, and analysts may publish and tailor their own concept definitions (however, always independent from the software applications).
- Finally, and because these special dictionaries are not part of the actual software application, the manner in which data is collected may vary, thus giving us a language which expresses data quality via business rules which can be used by different applications.

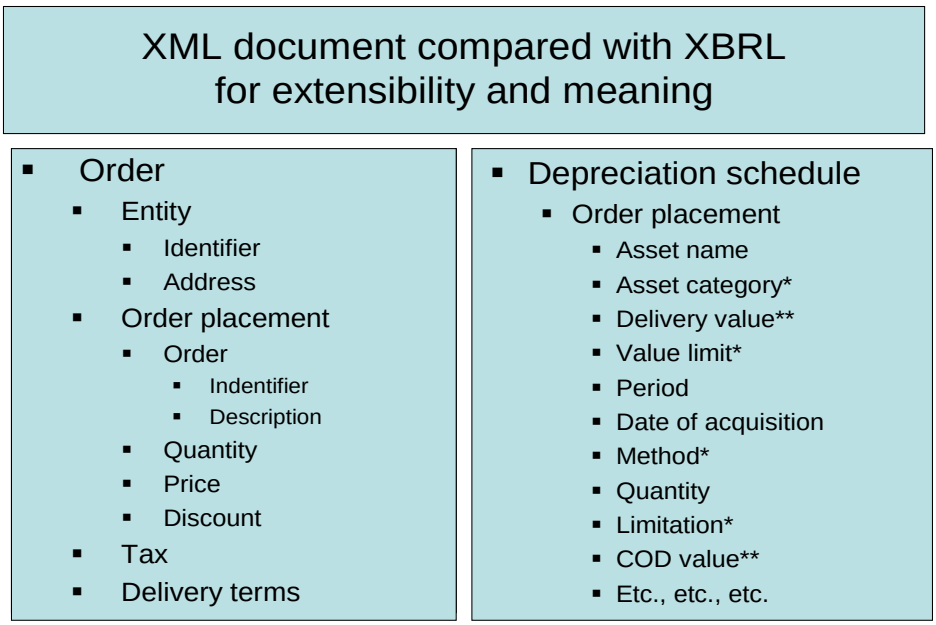
In this manner, and after many revisions from 1998 until today, XBRL, eXtensible Business Reporting Language, has been built as a language whose syntax has been designed for the exchange of business reports, based on XML and other complementary W3C standards such as the specification of Namespaces, the definition of XMLSchema for data templates, and the definition of XLink for linking resources.



3.3 XBRL is a labelling or "tagging" language expressing rules

In order to achieve extensibility and guarantee the uniqueness of a concept definition, the labelling of XML information is not enough. It is necessary to add rules to this information.

The reason for adding rules to the labelling language is based on the following:



* The meaning of these concepts may vary depending on the different accounting principles applied.
** The name of identical concepts may vary from one jurisdiction to another, not to mention the difference in language translation from one geographical location to another.

Therefore, rules are defined to achieve the same semantic value as the information expressed in the taxonomies.

In the XBRL language, in addition to the definition of concepts to be reported, the metadata of the concepts and their cross-concept relationships are expressed according to the rules of XML syntax, in the form of linkbases, which are described in greater detail in the next chapter.

- To get a better idea of the rules that define these relationships, here are a few examples:
- To define simple calculations, such as between two concepts, the concept TotalEmployees is expressed by the sum of the values of FixedEmployees and TemporaryEmployees.
- To define a business relationship, the concept TotalIndirectCost is similar to TotalIndirectExpense.
- To define presentation relationships among concepts, hierarchical dependencies are used to help understand the information being reported.
- To define the relationship of concepts and the relevant authoritative documentation, such as legal references, etc.
- To define the relationship of concept labels, as in a translation of the text into various languages.

4 XBRL concepts and recommendations

4.1 XBRL version 2.1 specification

4.1.1 Description of XBRL 2.1 specification

The document used for this edition is the following: XBRL-RECOMMENDATION-2003-12-31+Corrected-Errata-2005-03-24.doc

This document is based on XBRL version 2.1, which was issued by the Specification Working Group of the XBRL Community, and can be found on the Internet webpage http://www.xbrl.org.

The specification is organized by the following chapters:

Chapter 1: Introduction: The purpose of the specification is discussed, as well as the relationship to other work of the W3C and standard-setting bodies, terminology used, levels of conformance for XBRL processors, and namespaces used throughout the document.

Chapter 2: Changes from Specification 2.0a: This chapter explains the differences between published versions of XBRL 2.0a and XBRL 2.1

Specification 2.0a was approximately 40 pages long, and some aspects were quite ambiguous or too flexible, which led to the situation that certain taxonomy editors created instance documents that were validated per the specification, but were incompatible with one another. Specification 2.0a was used during the year 2003, until the new version 2.1 was approved at the end of 2003, at which time all the problems encountered during the early stages of XBRL adoption were resolved.

Chapter 3: XBRL Framework: This chapter gives an overview of XBRL instance documents and taxonomies. The function of the DTS (Discoverable Taxonomy Set) is discussed, which is a new feature in XBRL 2.1, and mention is explicitly made of the need to keep the security requirements separate from the XBRL application format of business reporting.

XBRL does not require, limit or deter the use of any security procedures, such as hash totals or encryption, and deliberately indicates that such procedures are outside the scope of XBRL. Security considerations are independent from the transmission of XBRL content.

The rest of the chapter describes the validation of XBRL instance documents and taxonomies, as well as the use of XLink in XBRL. XLink is a technology based on XML to define relationships between elements that are located in different files, similar to the way HTML handles links to other documents.

The first version of XBRL 1.0, as is true with the majority of XML-based languages, had a hierarchical structure with nested elements.

XBRL INSTANCE DOCUMENT VERSION 1.0 (ILLUSTRATION) **EXAMPLE 1**

```
<balanceSheet>
        <assets>
                <fixedAssets>1000</fixedAssets>
                ...
                <totalAssets>1000000</totalAssets>
        </assets>
        <liabilities>
                <subscribedCapital>2000</subscribedCapital>
                ...
                <totalLiabilities>1000000</totalLiabilities>
        </liabilities>
</balanceSheet>
```

It was quickly realized that this system had its limitations, as it was only possible to present a single structure for the balance sheet. It was not possible to define business rules to work with element values. Nor was it ever a business requirement to coincide the presentation of a balance sheet with the structure of XML. In the end, by going down this path, there were more problems involved than adopting a completely different structure.

XBRL 2.1 presents all the data values in a flat structure.

XBRL INSTANCE DOCUMENT - VERSION 2.1 (ILLUSTRATION) **EXAMPLE 2**

```
<xbrl>
  <schemaRef xlink:href="taxonomy.xsd"/>

  <fixedAssets>1000</fixedAssets>
  ...
  <totalAssets>1000000</totalAssets>
  <subscribedCapital>2000</subscribedCapital>
  ...
  <totalLiabilities>1000000</totalLiabilities>
</xbrl>
```

This instance document includes information from the taxonomy which describes the elements used in the document (fixedAssets, totalAssets, subscribedCapital, and totalLiabilities), as well as the possible hierarchies among the elements that the authors of the taxonomy wanted to achieve.

Examples of multiple relationships among elements are the following: Balance sheet in order of liquidity, balance sheet in the approved accounting presentation, and other calculated relationships between elements, labels in different languages, and relationships with legal texts which help to interpret the meaning and use of elements and inter-dependent relationships among elements.

The technology that gives us the opportunity to connect elements in such structures is called XLink. XLink appears for the first time in XBRL 2.0, and has been used extensively from that point on, so much so that the XBRL specification has an entire chapter devoted to the use of XLink in XBRL. It is highly improbable that XBRL will give up the use of XLink, since it offers the perfect procedure to connect information in the same manner that web pages link to each other.

4.1.2 XBRL instance documents

The next chapter of the specification offers an in-depth explanation of the format of instance documents in XBRL 2.1, and what it is possible to do with them.

With XBRL it is possible to create a pattern for any type of business report. XBRL is not designed to present exclusively financial information. Specification 2.1 does not presuppose that it will be used for any specific type of report.

It is true that XBRL originated at the AICPA (American Institute of Certified Public Accountants, http://www.aicpa.org) and that major advances have been made in financial reporting. However, even back in 1998 and 1999, XBRL was called XFRML (Extensible Financial Reporting Markup Language), and the current name Extensible Business Reporting Language was adopted later when it was shown that XBRL could be used not only for financial information, but also for any type of business report that one wanted to set up and exchange with others.

The specification starts off with explaining how an element in XBRL is defined in the XBRL instance schema xbrl-instance-2003-12-31.xsd, and shows a sample instance document, which we will go over now:

INSTANCE DOCUMENT - XBRL VERSION 2.1 **EXAMPLE 3**

```
<xbrl xmlns="http://www.xbrl.org/2003/instance"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:link="http://www.xbrl.org/2003/linkbase"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ci="http://www.xbrl.org/sp/general/2005/taxonomia-pgc-2005"
    xsi:schemaLocation="
http://www.xbrl.org/sp/general/2005/pgc-2005
http://www.xbrl.org.es/general/2005/pgc-2005.xsd">
  <link:schemaRef xlink:type="simple"
        xlink:href="http://www.xbrl.org.es/general/2005/pgc-2005.xsd"/>
  <ci:assets precision="3" unitRef="u1" contextRef="c1">727</ci:assets>
  <!-- ... other elements from the taxonomy with values ... -->
  <ci:liabilities precision="3" unitRef="u1" contextRef="c1">727</ci:liabilities>
  <context id="c1"><!-- ... --></context>
  <unit id="u1"><!-- ... --></unit>
</xbrl>
```

The element *xbrl* usually contains namespace (*xmlns*) definitions that are used in the rest of the document. This avoids having to redefine the namespace each time it is used. As an example, the text *xmlns:link="http://www.xbrl.org/2003/linkbase"* appears as an attribute of the element *xbrl* associated with the prefix *"link"* in the namespace *"http://www.xbrl.org/2003/linkbase"*, so that each time that we use *"link:"* in the rest of the document, we are referring to an element defined in the namespace above without needing to rewrite the entire text address.

The DTS (Discoverable Taxonomy Set) of this document is composed of the element *<link:schemaRef xlink:type="simple" xlink:href="http://www.xbrl.org.es/general/2005/pgc-2005.xsd"/>*

We can see in the DTS that there is only one referenced taxonomy *pgc-2005.xsd*. However, we may also use elements from many taxonomies in a single instance document, as long as each one of them is properly identified via the corresponding element *<link:schemaRef>*.

A taxonomy can also incorporate elements from other taxonomies. In this case, the taxonomy that appears in the DTS of the instance document is the one that is found at the end of the hierarchy. In our example we have two elements in the taxonomy (*ci:assets and ci:liabilities*) with the corresponding value 727 in each case. These elements contain the following information:

"precision=3" means that the three digits, counting from the left, starting at the first non-zero digit of the number, are meaningful and can be used for computations; in this case, the value 727. To obtain more information on the use of "precision" and "decimals", please consult the XBRL specification starting at Chapter 4.6.3, where there are detailed explanations with examples of how to interpret the value in each case.

We next see *unitRef="u1"*, which refers to the unit of measure for the value 727. In XBRL, all numeric values must have a unit of measure. This is shown by combining the attribute *unitRef* and successive elements *<unit id="u1">…</unit>* in the instance document. The syntax of the definition of the unit element allows you to define simple units such as *iso4217:EUR* to identify euros, and complex units such as m/s^2 to define acceleration if necessary. Please refer to Chapter 4.8 of the XBRL specification for a detailed description of simple and complex units of measures.

The next attribute of the elements in our example is *contextRef="c1"*. This attribute shows the relationship of the element with the dimensional context of the information being reported. In XBRL, the contexts contain at least two dimensions: time and entity. However, the context is extensible so that we can add information relating to the organisational unit of the entity (department, person, etc.) and the scenario of the instance document (estimate, actual, etc.) This information is created by using elements *<context id="c1">…</context>*, the syntax of which is described in Chapter 4.7 of the XBRL specification.

4.1.2.1 Simple and complex elements: Items and tuples

In the previous example, we see two elements in the taxonomy: ci:assets and ci:liabilities. These two elements, within a determined context, represent all the information necessary to be understood. The value of these elements (the number 727) has meaning on its own, and that is why we call them simple elements.

However, there are many times when the reported information must remain grouped for it to be understood. For example, if we need to report on a company's management, with the name, position, fixed salary, and variable salary for each person, we use a structure in XBRL called a tuple to express such complex data as they are dependent on each other for proper understanding.

The tuple is a data structure which groups together simple elements that are dependent on each other to be understood. In the previous example, the tuple could be called Director, and it contains an element called Name, an element called Position, an element called Fixed Salary, and another one called Variable Salary.

In an XBRL instance document, we would see the following:

TUPLE IN AN XBRL INSTANCE DOCUMENT **EXAMPLE 4**

```
<ci:director>
  <ci:name contextRef="c1">Juan Ramón Martínez</ci:name>
  <ci:position contextRef="c1">Finance Director</ci:position>
  <ci:fixedSalary unitRef="Euro" contextRef="c1" precision="INF">45000</ci:fixedSalary>
  <ci:variableSalary unitRef="Euro" contextRef="c1"
precision="INF">15000</ci:variableSalary>
</ci:director>
<ci:director>
  <ci:name contextRef="c1">José Fernandez</ci:name>
  <ci:position contextRef="c1">Managing Director</ci:position>
  <ci:fixedSalary unitRef="Euro" contextRef="c1" precision="INF">55000</ci:fixedSalary>
  <ci:variableSalary unitRef="Euro" contextRef="c1"
precision="INF">20000</ci:varaibleSalary>
</ci:director>
```

4.1.2.2 Explanatory notes in XBRL

One of the most significant features of XBRL instance documents is the possibility of adding explanatory notes to the elements that appear in the reports. In real life, these notes appear in practically all corporate financial reporting, and often contain valuable information that should not be ignored. XBRL features the possibility of incorporating these notes thanks to using the power of XLink.

FOOTNOTE FOR THE ELEMENT ACCOUNTSRECEIVABLE **EXAMPLE 5**

```
<ci:accountsReceivable id="item1" unitRef="Euro" contextRef="c1"
precision="INF">455680</ci:accountsReceivable>

<link:footnoteLink
  xlink:type="extended" xlink:title="Notas"
  xlink:role="http://www.xbrl.org/2003/role/link">
  <link:footnote
    xlink:type="resource"
    xlink:label="footnote1"
    xlink:role="http://www.xbrl.org/2003/role/footnote"
    xml:lang="en">Fees, events, and courses receivable (fiscal year).- Variation of amount
from transfer of balances in 2004 to past years, from allowance for unpaid fees in the timef
established by some members of the XXXX, in addition to events pending payment.
</link:footnote>
  <link:loc xlink:type="locator" xlink:label="fact1" xlink:href="#item1"/>
  <link:footnoteArc
     xlink:type="arc"
     xlink:from="fact1" xlink:to="footnote1"
     xlink:title="see explanatory note"
     xlink:arcrole="http://www.xbrl.org/2003/arcrole/fact-footnote"/>
</link:footnoteLink>
```

In this example we see how the element accountsReceivable has the identifier *id="item1"*, which is used in the locator *<link:loc …/>* to refer to the element that belongs to the footnote. The text of the footnote is presented as a resource, and the relationship between the element and the text of the footnote is defined in the arc footnoteArc.

It is possible to publish notes in many languages, and this allows the reader to view the information in the language that he or she understands. See below for an example in Spanish. In the previous example, it is sufficient to create only one element for this purpose.

```
<link:footnote
    xlink:type="resource"
    xlink:label="footnote1"
    xlink:role="http://www.xbrl.org/2003/role/footnote"
    xml:lang="sp">Cuotas, eventos y cursos por cobrar …
</link:footnote>
```

It is placed inside the element <link:footnoteLink>.

Please refer to Chapter 4.11 of the XBRL specification for further information on footnotes.

4.1.3 XBRL taxonomies

According to the theory of communication, in order to exchange a message between a transmitter and a receiver, the participants must know a common code. Such is the role of XBRL Taxonomies.

An XBRL taxonomy contains the following:

- **Schema:** A group of structured elements that may be used in instance documents. We will call this the dictionary of defined terms.
- **Label linkbase:** Labels or text associated with the elements in the dictionary may be created in different languages and used for different purposes when composing different instance documents.
- **Reference linkbase:** References to legal texts or accounting standards on which the concept is based. These references play an important role when determining the use of concepts during the creation of instance documents.
- **Presentation linkbase:** Rules to specify the formatting of a report .
- **Calculation linkbase:** Rules for calculations (additions and subtractions) between elements in the taxonomy that are used to validate instance documents.
- **Definition linkbase:** Additional rules to document relationships between elements in the taxonomy and that are used to validate the instances.

Chapter 5 of the XBRL specification is totally devoted to explaining XBRL taxonomies.

Every XBRL taxonomy must include an XML schema. The rules and limitations of XML schemas are also applicable to XBRL taxonomies.

An XBRL taxonomy may include another XBRL taxonomy. This feature of XBRL is essential to implementing the extensibility of taxonomies. A business entity that needs to provide more information in its reports may create an extension to the original taxonomy, with the appropriate elements and inter-relationships, for any special reporting requirements

4.1.3.1 Definition of concepts in XBRL taxonomies

Concepts are defined in taxonomies by creating elements that have a data type defined in XBRL schemas, and which belong to either the item or tuple group of elements.

DEFINITION OF AN ELEMENT IN AN XBRL TAXONOMY **EXAMPLE 6:**

```
<element
id="tax_CashInBanks"
name="CashInBanks"
xbrli:periodType="instant"
  type="xbrli:monetaryItemType"
  substitutionGroup="xbrli:item" nillable="true"/>
```

In this example, we have the definition in the taxonomy schema for a simple concept called *CashInBanks*.

Now we see how this element can be used in an XBRL instance document:

USE OF THE PREVIOUS ELEMENT IN AN XBRL INSTANCE DOCUMENT **EXAMPLE 7**

```
<tax:CashInBanks unitRef="u1" contextRef="c1"
precision="7">1523554</tax:CashInBanks>
```

Every element that is defined in the XBRL taxonomies must have the attribute *xbrli:periodType**="instant"*** or *xbrli:periodType=**"duration"*** to be able to identify its relationship with the appropriate period of time.

Concepts defined with the attribute type "instant" are associated with a specific moment in time. For example, concepts on a balance sheet are of the type "instant".

Concepts defined with the attribute type "duration" are associated with a range of dates (*startDate* and *endDate*). Data on the income statement always refer to concepts with the attribute type "duration".

When building taxonomies, it is to be assumed that the concepts are of the type "duration" unless there is a reason for them to be of the type "instant".

In XBRL there are other optional attributes that may be added to the definition of a concept. An example is the attribute balance="credit" or "debit", that may be used to identify accounting concepts and if they appear on the right or left-hand side of the financial report.

Chapter 4.9 of the XBRL specification gives us documentation and examples of how to create a tuple in a taxonomy and how to represent data in an XBRL instance document.

It is possible to define abstract concepts in a taxonomy. These concepts are not used in instance documents, but they do add clarity to the structure of the taxonomy. For example, the concept "Balance Sheet" is not an element that contains data; however, we can use this element in the presentation linkbase as the parent of Assets and Liabilities, and thus maintain a well-organized taxonomy.

4.1.3.2 Label linkbase

XBRL taxonomies keep metadata information for defined concepts in the schema, and this helps to document the concepts and is useful for software applications.

A good example of this is the label linkbase. All linkbases are located in separate files in the taxonomy schema, and the relationship between the information in these linkbase files and the concepts in the taxonomies are created via XLink.

The label linkbase provides the text that appears to the left of the data. People can easily understand that the data corresponds to the concept that appears on the same line. For example, if a report shows:

Cash in banks ..... 1.000 €

A person quickly understands the concept because of the text on the left-hand side of the page, which corresponds to the value across in the right-hand column.

The text “Cash in banks” is stored in the XBRL label linkbase, and this system allows us to have various label linkbases with a different text for each desired language, thus fulfilling any requirement to format and present the information in a different foreign language.

Labels have an associated role. It is possible to define the roles of a taxonomy, and then define the labels that a concept will have in different roles. For example, roles may be associated with the concept “operating profit (loss)”. If the balance is positive, the label will read “Operating Profit”, but if the balance is negative, the label will read “Operating Loss”.

Chapter 5.2.2 of the XBRL specification provides more information on the use of label linkbases.

4.1.3.3 Reference linkbase

One of the most important linkbases in the definition of a public taxonomy is the reference linkbase.

This linkbase provides information about the authoritative documentation where we can find additional information on the concept. This linkbase is very useful when we need to locate the correct concept for creating XBRL instance reports, as when using the IFRS taxonomy.

The easiest way to select a concept from the taxonomy is the following:

1 Starting with the concept that appears in the accounting document, we locate the IFRS reference by referring to the documentation provided by the entity’s auditor or the corporate chart of accounts. For example, this reference may be IAS-20, which means that the authoritative literature for this concept may be found in the guide for IFRS under International Accounting Standards, paragraph 20.

2 2. The next step is to search in the reference linkbase for concepts using IAS, 20 as their reference documentation.

This simple procedure is advisable for creating a mapping between the accounting concepts used in an entity’s books and the use of international accounting standards.

Chapter 5.2.3 of the XBRL specification defines the syntax to use in taxonomies in order to document the elements that define the relationship with legal texts. Taxonomies are very flexible when creating new concepts and this method is very useful.

4.1.3.4 Presentation linkbase

This linkbase serves a double purpose: on the one hand, it is used by software tools for the creation and visualization of taxonomies in a more “friendly” format than a simple list of concepts. On the other hand, it serves as a basis for software applications to format the reports automatically and to have a point of departure for creating stylesheets.

The presentation linkbase has a hierarchical structure. It is built with parent and child relationships using XLink.

Chapter 5.2.4 of the XBRL specification provides more information on the use of this linkbase.

There is an optional attribute called preferredLabel which is use by the XLink arc to join two elements in a presentation linkbase. This attribute may contain the role value that the element is using at a specific time. It is very normal to find schedules in the annual report similar to the following:

Movements in reserves:
Reserves at beginning of period ……10.000 €
Increase or decrease of reserves …… ……2.500 €
Reserves at end of period ……12.500 €

In XBRL, this is done in the following manner:

There are three concepts in the taxonomy: one called “Reserves”, another called “Increase or decrease of reserves”, and an abstract concept called “Movements in reserves”. The concept “Reserves” has three labels:

“Reserves”, for the role “normal label” (used on the balance sheet, for example).
“Reserves at beginning of period”, for the role “beginning of period”.
“Reserves at end of period”, for the role “end of period”.

The concept “Reserves” is of the *periodType “instant”*.

The concept “Increase or decrease of reserves” is of *periodType “duration*”.

The concept “Movements in reserves” is an abstract type, so it is neither “instant” nor “duration”.

In the presentation linkbase, in some part of the disclosure notes, there will be the concept “Reserves” as the first child of the abstract element “Movements in reserves”, with the preferredLabel “beginning of period”.

In the same presentation linkbase, there will be the element “Increase or decrease of reserves” as the second child of the abstract element “Movements in reserves.”

Finally, the element “Reserves” will appear a third time as the third child of the abstract element “Movements in reserves”, but this time it will have the preferredLabel “end of period”.

4.1.3.5 Calculation linkbase

This linkbase allows us to create calculation relationships between elements, similar to those in the presentation linkbase. However, in this case, the parent element will be the result of the mathematical operations of the child elements.

These operations can only be the addition or subtraction of child elements.

Chapter 5.2.5 of the XBRL specification defines how these calculation networks are set up.

XBRL documents may be validated for these calculation networks. The validation process may result in calculation errors, which will result in an invalid XBRL document . Software programs will take such errors into account when automatically accepting or rejecting instance documents.

4.1.3.6 Definition linkbase

The last linkbase documented in XBRL 2.1 specification is the definition linkbase. It contains and documents a variety of miscellaneous relationships between concepts in taxonomies, and adds some very important rules for validating XBRL documents.

The following four arc relationships between elements are defined in the specification:

1 "general-special": Defines the relationship between the general and the specific. The example given in the specification is that postalCode is a generalization of zipCode.

2 "essence-alias": This arc role is used to relate concepts from different taxonomies, or two concepts from the same taxonomy, to indicate that both are essentially the same concept.

3 "similar-tuples": This arc role is used in a similar way as the “essence-alias”, but for tuples. The differences with “essence-alias” are documented in the specification.

4 "requires-element": This arc role is used to require that an element that occurs in one instance must occur in the other XBRL instance.

Linkbases are also extensible. There is nothing written in the specification to prohibit the development of proprietary linkbases to relate internal data structures with taxonomy elements. These linkbases will have to remain private as long as there is no specification approved by the XBRL consortium for all XBRL processors to understand.

4.1.4 References

Specification XBRL 2.1:
XBRL-RECOMMENDATION-2003-12-31+Corrected-Errata-2005-03-24.doc

4.2 Storage

4.2.1 Storage of XML / XBRL

The appearance of new technologies usually brings along solutions to existing problems and opens new possibilities for development, albeit with new requirements.

The introduction of XML technology has resolved certain problems, especially in the area of communicating across different platforms, but it has also brought along with it certain new requirements, such as the storage of documents and access to its content, which is structured in a different way when compared with information stored previously.

XBRL has emerged as a powerful and flexible version of XML, conceived specifically to satisfy the demands of business and financial information. Although XBRL, from a functional point of view, appears to be a specialisation of XML, it does not require a "structural" change from a technological point of view. Most tools or management software programs that were designed for XML can be used with XBRL.

This document is meant to identify the requirements of XBRL content as well as technology solutions available for its storage, management and operations. We will review different storage methods, discuss the main products that are currently available in the marketplace for these methods, and describe some tools for the administration, management and use of XBRL content.

4.2.1.1 Storage in files

XBRL instance documents are structured and normally received in plain text files. Based on this observation, the first option to consider is the storage of XBRL content in its usual "container", i.e., the file itself.

Advantages:

- The document is not processed, so its information content has not been altered or distorted.
- 'Ease of use' to manage the documents at the file level, as long as the file structure can be classified in accordance with the criteria of the organisation.

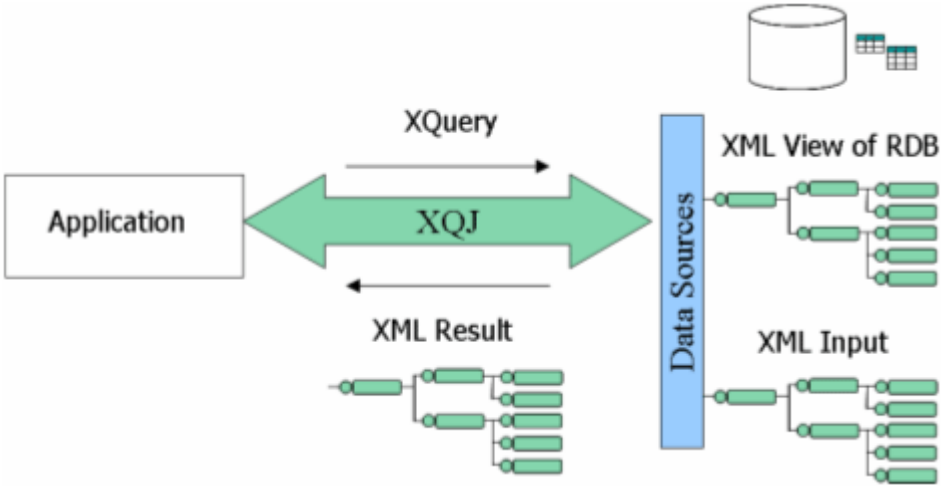
Disadvantages:

- There are the typical problems associated with file management: lack of concurrence, testing for data integrity, security, etc.

Products/Tools:

Based on the understanding that XBRL content is stored for future reference and queries, as well as for publishing and use by other systems, it is important to use tools that allow the user to access, locate, and extract information. In the case of storage in a file structure, the following tools are available:

- Search engines for XML/XBRL files whose query syntax is based on XQuery. Most of these search engines use an API for access from outside applications (normally Java or C++).
- **XQEngine** (http://xqengine.sourceforge.net)
- **XQJ: Java Community Process**: http://jcp.org/en/jsr/detail?id=225.



- **Microsoft.Xml.Xquery.dll**: from the Microsoft library using XQuery.
- **DataDirect Xquery** (http://www.stylusstudio.com)

4.2.1.2 Storage in relational databases

Those companies that are looking at options for storing XBRL documents may consider using the storage systems that they have already acquired. The most common system is the relational database. So, the question is "How to store XBRL content in a relational database?" The answer is based on the following options:

- Transforming the XBRL content into a relational model.
- Storing the entire contents in columns of a specific type.

Advantages:

- Relational databases are very robust products and may be considered "mature" based on their development and place in the market.
- Most of the database applications currently in use that access, query, manage, analyze, and publish content are based on relational database systems. Therefore, it would be desirable to reuse these applications in connection with XBRL information.

Disadvantages:

- The conversion and transformation of XBRL data to a relational model requires some type of manipulation of the information, although it is not always possible to guarantee the integrity of the output, nor ensure that it will be exactly the same as original document received in XBRL.
- If the XBRL document has not been generated from a relational schema, the subsequent adaption to a relational model is not easy, especially for the conversion of certain elements, such as:

  - Nested elements.
  - Repetitive elements (multi-value attributes).

4.2.1.2.1 Transformation of XBRL content to relational models

The process of transformation includes the “fragmentation” of the document content, i.e., the data contained in the XBRL file is extracted and stored in entities of the database.

First of all, it is important to clarify that when you use this storage technique, you are not saving the information in the same XBRL format, since the XBRL document is completely foreign to the database, and the document is no longer useful once the information has been extracted.

When a document is recovered, the database is queried and an XBRL document is created based on the results obtained. El hecho de recuperar un documento significa consultar a la BDD y construir un documento XBRL con los resultados obtenidos.

An important consequence of using this system is that there is information relating to the XBRL document that has not been incorporated into the database and so is lost, such as the order in which the elements appear in the document.

The relational model will only be efficient to the extent that the data is highly structured and uses a known schema. This model has the advantage of being able to query the database in a traditional manner, although such queries usually require a lot of joins due to the structure of XML. However, this option is not the best one if there are nested elements or elements that are repeated, since you need to use a tree view or store the relationship between elements in a high-to-low order.

The simplest way to use this type of storage is to define a mapping between the data in the XML data in the file and the tables of the database. In this way, it is possible to load the data all at once. This mapping can also be used in the reverse, as when we have data in the tables and we want to generate an XML document.

Products/Tools:

- SQL Server 2005 (http://www.microsoft.com)
- DB2 XML Extender (http://www.ibm.com)
- Oracle XML DB (www.oracle.com)

4.2.1.2.2 Storage of full documents in ‘XML’ type columns

A solution for the storage of XML/XBRL is the use of XML-type columns that the different database software vendors have included in their products.

It should also be pointed out that when the content is based on “plain text”, it is always possible to save the XML in a VARCHAR field, although it is only useful if you choose to store and retrieve the full text without internal queries. Nevertheless, in this case the following should be taken into consideration:

- It is possible to combine XML-type columns in order to keep an exact copy of the document (although there are redundancias), for example, for legal papers.

- It is possible to convert into XML type at the time of executing Xquery, for example, although perfomance would be quite penalised.

Normally, the XML type will be used instead of the VARCHAR type. This practice is advisable when:

- The structure of the data is not known, or the structure of the data may change significantly in the future.
- The data represents the hierarchy of contention (as opposed to references between entities) and many are recursive.
- The order is inherent in the data.
- Data is required to be validated by the database engine using a schema, DTD, or taxonomy.

In short, storage in XML-type columns is only useful when you have XML documents with a variety of structures, or XML documents which conform to complex or different schemas and which are very difficult to map to relational structures.

The XML type of data allows for data to be indexed, and can be queried or modified by means of XQuery and XML DML (extension for modifying data).

Products/Tools:

- SQL Server 2005 - (http://www.microsoft.com)
- DB2 XML Extender - (http://www.ibm.com)
- Oracle XML DB - (www.oracle.com)

4.2.1.3 Storage in native XML databases

Introduction and background:

As we have seen in previous chapters, relational databases offer the possibility of storing XML data, but they are not designed specifically for storing hierarchical structures like XML documents:

- Relational databases have a regular structure when compared to the hetereogenous nature of XML documents.
- XML documents usually contain many levels of nesting, whereas relational data are “flat”.
- XML documents have an intrinsic order, whereas relational data are “unordered”.

- Relational data are generally “thick” (a table where each column has a data category and each row represents the actual data), whereas XML data are “disperse” and may represent a lack of information due to the absence of an element.

An alternative to the relational database management system (RDBMS) is the object-oriented database management system (ODBMS). Object-oriented DBMS support a model of pure objects, as it is not based on extensions of other more classical models, such as the relational database. This type of system offers some attractive features for the storage of XML information:

- It sees all information in the document as predefined class objects, interconnected by links embedded in the structure of the XML document.
- It uses the DOM approach and can deal with well-formed documents.

- The ODBMS option is used by various native dababases, but the mechanisms for indexing, optimization, processing of queries, etc. are part of the ODBMS and, generally speaking, are not specific to the XML model.

Here are some examples of ODBMS currently on the market:

- Poet (http://www.poet.com)
- Jasmine (http://www.cai.com)
- ObjectStore (http://www.odi.com)
- GemStone (http://www.gemstone.com)

Native XML databases:

The organization XML: DB Initiative for XML Databases (http://www.xmldb.org) describes the Native XML Database as a: "*logical mode for XML documents that stores and retrieves documents in accordance with said model*".

In contrast with relational databases that operate with 'atomic or fragmented data', the XML native databases lack the concept of fields, and focus on the storage of the XML document instead of individual pieces of data.

This type of database stores information in XML format thanks to "XML type" repositories such as DOM or Infoset. These repositories also store the general index and the index associated with each document.

We would like to point out that the available native XML databases do not have a specific physical storage infrastruture; they are built on different database structures, such as relational, hierarchical, object oriented or proprietary storage formats.

Advantages:
- This type of storage system is designed specifically to store XML content, with the following features:
- It is not necessary to convert the information to either a relational structure or any other structure which is not exactly XML.
- It uses consulting management tools expressly designed for this type of information, which benefits its operation, efficiency, and performance.
- It uses tools specifically designed for this type of intormation, such as validation for schemas, DTDs or taxonomies.
- It uses indexing specially designed for XML content, which is why it is very effective in locating, handling, and retrieving XML content.

Products/Tools:

Tamino XML Server (version 4.2) - (http://www.softwareag.com)
Semansys Tamino XBRL Suite - (www.semansys.com–www.softwareag.com)
Xindice (version 1.1.b4) - (http://xml.apache.org)
eXist (version 1.0.b2) - (http://exist.sourceforge.net/)

BBDD NATIVAS XML **LISTADO DE PRODUCTOS / FABRICANTES:**

Nombre	Fabricante	Tipo Licencia	Sistema Almac.	APIs
Centor Interaction Server	Centor Software Corp. http://www.centor.com	Comercial	Propietario	C++, Java
DXML	http://www.dbxml.com/	GNU LGPL	Propietario	Java, Javascript
GoXML DB	The Xenos GoXML™ Integration Solution http://www.xmlglobal.com/solutions/prod_goxml_overview.htm	Comercial	Propietario	Java
Infonyte DB	Infonyte http://www.infonyte.com/en/products.html	Comercial	Propietario	Java
Ipedo XML Database	Ipedo http://www.ipedo.com/html/ipedo_xml_database.html	Comercial	Propietario	Java
Tamino XML Server	Software AG http://www2.softwareag.com/Corporate/products/tamino/default.asp	Comercial	Propietario	Java
TeamXML/ TeamSite	Interwoven http://www.interwoven.com/products/features/xml/teamxml.html	Comercial	Propietario	Java
Virtuoso	OpenLink http://www.openlinksw.com/virtuoso/	Comercial	Propietario	
Xindice	Apache Software Foundation http://xml.apache.org/xindice/	Apache Software License	Propietario	Java
XDBM	http://sourceforge.net/projects/xdbm/	GNU LGPL	Propietario	
DBDOM	http://sourceforge.net/projects/dbdom	GNU LGPL	Relacional	Java
EXist	http://exist.sourceforge.net/	GNU LGPL	Relacional	Java
XDB	MindSuite http://xdb.wiredminds.com/	Comercial	Relacional	C++, Java
Birdstep RDM XML	Birdstep Technology http://www.birdstep.com/database_technology/rdm_server.php3	Comercial	Orientado Objetos	C, C++, Java
Ozone	http://www.ozone-db.org/frames/home/what.html	Open Source, Ozone Library License	Orientado Objetos	Java

4.2.2 XML query languages

4.2.2.1 XQuery - XPath (http://www.w3.org/XML/Query)

XQuery is a language that is still in the process of development, although quite advanced, and is scheduled to be released in 2005. The features in the latest version of the working draft, dated October 29, 2004, can be found at: http://www.w3.org/TR/xquery .

XQuery is a functional language. Instead of executing commands like a procedural language, each Query is an expresión to be evaluated. Expressions can be combined to form new expressions

XQuery is based on the following languages:

- XPath y XQL: Syntax to navigate through hierarchical documents.
- SQL: (relational language). FLWR expressions and operators like 'join'.
- XML-QL: To assign variables to create new structures.

XQuery makes extensive use of XPath (a language used to select portions of XML); in reality, some people see XQuery as a superset of XPath. The new version of XPath (2.0) is being developed by the same working group and in coordination with XQuery version 1.0.

A list of software vendors that offer XQuery solutions is found on the webpage of the W3C Consortium. Here is a partial list of such vendors:

- BEA's Liquid Data
- Berkeley Lab's Nux, an open source XQuery extension to XOM.
- Blackpearl's Blackpearl 4 platform, with an embedded XQuery engine
- Bluestream Database Software Corp.'s XStreamDB
- Cerisent's XQE
- Cognetic Systems's XQuantum
- GNU's Qexo (Kawa-Query)
- Compiles XQuery on-the-fly to Java bytecodes. Based on and part of the Kawa framework. An online sandbox is available too. Open-source.
- Ipedo's XML Database v3.0
- Microsoft's SQL Server 2005 Express, with XQuery support
- Neocore's XML management system (XMS): http://www.neocore.com/products/products.htm
- Oracle's Xquery Technology - Preview
- Renmin University of China's OrientX
- Sleepycat's Berkeley DB XML 2.0, an embedded native XML database with support for XQuery 1.0 (July 2004 draft), implemented in C++, with interfaces for Java, Python, Perl and PHP. Open source.
- Software AG's - Tamino XML Server
- Sonic Software's - Stylus Studio 5.0 (XQuery, XML Schema and XSLT IDE)
- Sourceforge's eXist. Open-source.
- Sourceforge's XQEngine. Open-source.
- Sourceforge's XQuench. Open-source.
- Worcester Polytechnic Institute's RainbowCore. Java

4.3 Sending and receiving XBRL documents

Given that XBRL is modelled on XML, exchanging files may done in several different ways. Here is a description of some of the alternatives:

Web services facilitate the sharing of information generated from software applications, and these applications can, in turn, invoke functions from other programs independently across virtually all operating system platforms and devices. There are many Web services using php (hypertext pre-processor which runs on the server).

The protocols which support Web services normally communicate via port 80, based on HTTP, GET and PUT methodology. This means that we can access them in the same way that we do a Web page. The difference between a Web page and a Web service is that any interested party can access a page, whereas the service is accessed only by programs upon request.

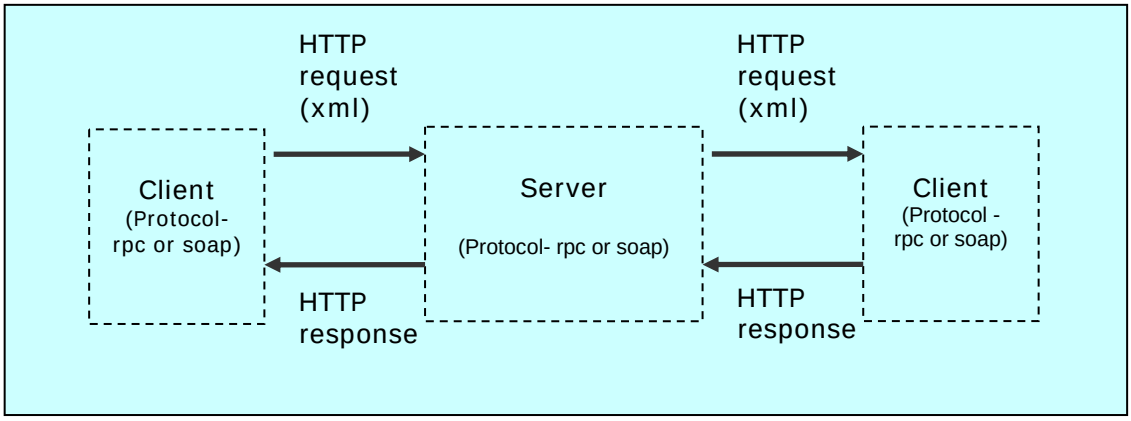
It is possible to set up a simple Web service on a system, but at the risk of not complying with communication standards. Therefore, we should understand that is necessary to work with standard protocols to set up a fully functional Web service.

There are two dominant trends: XML-RPC protocol for remote processing (RPC: Remote Procedure Calling), and SOAP (Simple Object Access Protocol). When programming for a Web service, it is important to decide which protocol to use because they are incompatible. In other words, if we program a Web service using XML-RPC, we cannot invoke it from a programming language that works with SOAP, such as .Net from Microsoft.

The difference between SOAP and XML-RPC is in its complexity. XML-RPC is designed to be simple, whereas SOAP was created with the idea of giving complete and full support for Web services.

A Web service is invoked by sending a request in the XML language, and the response is a Web page coded in XML. The process is started from the client application by using routines from the client protocol (SOAP or XML-RPC), which are translated into an XML request for a Web service via HTTP, and the corresponding response is returned and processed.

The following figure shows a simplified version of the process of sending an XML file. As you can see, any client can call up the Web service located on the server by sending a file in XML format. This file is processed on the server and sent back to the client in XML format. Once the client has received the file, it will continue the process outlined below:



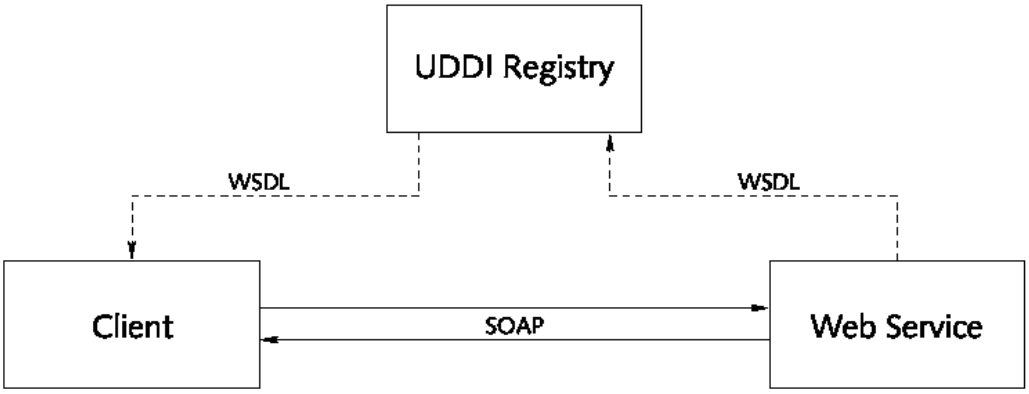
If we enter into a bit more detail, we can say that when sending a message in XML-RPC, this is an HTTP-POST request. The body of the message is in XML, the procedure is executed on the server, and the value returned is in XML format.

If we use SOAP, the protocol is based on XML and consists of three parts: First, defining the message and determining how to process it; second, a rules-based system to express defined types of data; and third, how to response to calls by remote procedures.

SOAP, in contrast to XML-RPC, is embedded in an infrastructure. It is not a mere communications protocol for computers, but is surrounded by terms such as WSDL and UDDI, which mean:

- **WSDL** (Web Services Description Language) is a file in XML format that tells the requesting computer which services are on its website. It also gives an exact reference to its services to be able to call them using the appropriate parameters.
- **UDDI** (Universal Description, Discovery, and Integration). UDDI is a Web service that can be used from applications dynamically to find out what other services are available, all perfectly integrated in a simple XML interface.

The following figure shows the relationship among the technologies:



The UDDI specification, together with Extensible Markup Language (XML), Simple Object Access Protocol (SOAP) and Web Services Description Language (WSDL), are gaining broad support in the framework of Web services.

This technology may be useful for certain areas of programming because it is very flexible. However, despite the fact that it is a technology that tries to share the workload over the Internet and on the Web service server, it still uses the client-server model (based on centralised systems). This leads us to wonder what would happen if the UDDI registry ever failed; even if the system were replicated, it would cause a bottleneck in the servers.

Here are the advantages and disadvantages of this system:

Advantages:

- Interoperability between software applications, independent of characteristics or system platform.
- Web services encourage standards and protocols based on text, which makes it easier to access its content and understand how it works.
- Since it is supported by HTTP, Web services can take advantage of security firewall systems without having to change the filtering rules.

Disadvantages:

- Execution of transactions cannot be compared to more fully developed open standards of distributed programming models.
- Performance is low in comparison to other distributed programming models, such as RMI or Corba. This is one of the drawbacks to adopting a text-based format.
- Since it is supported by HTTP, it can avoid security procedures based on firewall rules which try to block or audit the communication between programs on either side of the wall.

Another option is the use of electronic mail as a tool for exchanging documents. This tool is the most used Internet application in an enterprise environment because it is quick, easy to use, and convenient.

Every e-mail system should guarantee, at a minimum, that the messages arrive at their destination (in both directions), and that they pass through a control point upon arrival to ensure security (anti-virus, firewall). Furthermore, it is important to store the receipt of each message's delivery in a centralised repository that can be accessed for future retrieval of the documents.

The e-mail system has many advantages, such as:
- It is a very well-known tool.
- It is easy to use and is inexpensive.
- It does not require a very complex infrastructure.
- It is an asynchronous system, and does not require simultaneous communication.

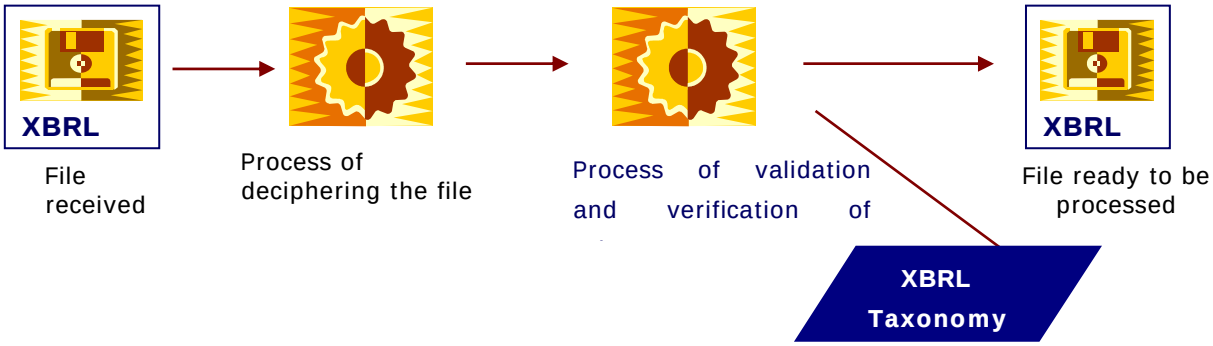
On the other hand, this system is quite insecure, so security policies should emphasize this point to avoid the misuse of information. Methods should be adopted to ensure security, such as encryption and/or digital signatures that comply with standards (see chapter on security). It is also advisable to install a secure server for e-mail. These servers are very functional and high performance machines, offering a wide range of new techniques and protocols to improve efficiency, robustness, flexibility, and security.

A third possibility is the use of an FTP client. This is a transfer protocol for files. It is a very popular way to transfer files in a trustworthy, synchronous manner, and more secure than e-mail because proper authorisation is required to execute tasks such as to display, download, copy, etc. Most errors generated by the coding of various e-mail protocols are also avoided. Furthermore, there are many FTP clients with a display interface that facilitates the use of these programs.

4.4 Receiving and processing information

Once an XBRL file has been received, a series of steps is required prior to processing it. If the information is sensitive material, reasonable security measures should be taken to avoid unauthorized access to the data; it is recommended that the documents have a digital signature from the company, auditor, or a trusted certifying authority.

The diagram below shows a simplified view of these steps:



The file received has been encoded in accordance with security specifications outlined in the chapter on security. Therefore, the first step is to decipher it in order to get a legible file to work with. Then, the document must be validated. For the validation process it will be necessary to have access to the XML schema that describes the validity of labels appearing in the document, as well as xlink information to validate the document's logic. In addition, it is necessary to have an established taxonomy or a set of XML files that describe the line-items or elements in a financial report (these concepts are explained in greater detail in previous sections of this paper).

Once the file has been validated in accordance with specifications, it is possible to go on to the next step of storing and/or processing it.

Storage of a file may be in a relational database or in some directory on the system. It may be decided to store the file directly upon receipt, or first decipher it and then store the processed file in a database table. Please refer to section 4.5 of this paper to find a complete description of the storage process.

On the other hand, and given that XBRL is based on XML concepts, it is possible to use the XSLT specification (XML Stylesheet Transformation Language) to process the file received. XSLT is a language to transform XML files, and is supported by XPath, which is a language for writing search queries based on XML tree nodes. This type of transformation may generate files in its own xml format, in html or txt. It can also generate PDF (Portable Document Format) or RTF (Rich Text Format) by using the style language specification XSL-FO (Extensible Style Language Formatting Objects).

When using Java technology, there are two APIs (application programming interfaces) for processing an XML file. Here is a brief description of the basic characteristics of each one:

- **SAX** (Simple API for XML). This is a series of classes for working with an XML document from programs written in Java – can access data, verify if document is well-formed and has a valid format.

  - The main characteristic is that the document is read sequentially, from start to finish, without loading the entire document into memory.
  - Advantage: efficiency of time and memory used in the analysis.
  - Disadvantage: does not have a tree-like structure for documents.
  - Requires a SAX analyser.

- **DOM** (Document Object Model). This is a series of classes for working with XML documents from programs written in Java.

  - The main characteristic of the DOM is that the entire document is loaded into memory, in a tree-like structure.
  - Advantage: the availability of the document structure allows access to the data based on the hierarchy of elements, and it is possible to modify the content of the documents, even creating new ones from scratch.
  - Disadvantage: time required.
  - Requires a DOM analyser.

From the file transformation process we get a file with with the required format, either in SAX or DOM. Storage of the file will be based on storage policies addressed in section 4.5.

4.5 Security

4.5.1 Security of the application

Restricted access: Access to the application should be restricted to those users who have rights to that information. The following controls should be put in place to ensure restricted access:

- Set up menus to control access to the system functions of the application.
- Restrict information and application functions to authorised users.
- Control the access rights of users.
- Ensure that the output of the application generating the instance document only contains the correct information and is sent only to authorised destinations.

Identification and Authentication: As regards the generation of instance documents, there should be a procedure that calls for the absolute and personalised identification of each user attempting to access the system. It is advisable to keep a list of approved users with access to restricted applications, based on their profile and access to that area of the system.

Furthermore, there should be a limited number of unauthorised attempts to access this part of the system.

Different authentication procedures may be used to approve access to the user. The password process is a common way to identify and authenticate a user. It is also possible to use cryptographic means and authentication protocols.

Identification and authentication can also be achieved with devices like intelligent cards, mini-calculators with stored keys, or biometric technologies that use unique characteristics or attributes of a person. A combination of technologies and devices using a secure connection may provide for a more robust authentication.

Follow-up of access granted to the application: When an instance document is generated, it is necessary to keep a record, or log, of the user's action, including the date and time of instance generation, identification of the source (terminal) from where the instance was generated, if possible, and a record of the number of attempts, both accepted and rejected, and, finally, if the process has been completed satisfactorily.

Location of the application: The application generating the instance document should reside on a server having restricted access.

Furthermore, only authorised personnel should have access to the physical location where the server is installed.

Limitation of access time: For additional security, connection time should be restricted for access to high-risk applications. Limiting the connection time from a terminal reduces the "window of opportunity" for unauthorised users.

Backup: There should be regular security backups performed on the server where the application resides.

Contingency plan: A contingency plan implies an analysis of possible risks to which the application is exposed, including information contained in the various storage units. A risk analysis should be performed to prepare for subsequent procedures, in case risk has actually been identified.

Despite all security measures, disasters can happen. Therefore, it is necessary to have a contingency plan that covers recuperation from disasters. This would include restoring computer service fast and efficiently, at the lowest cost, and with the least possible loss of information.

4.5.2 Security of instance documents

Logical restricted access: Access should only be granted to approved users for the directory containing the instance documents. The following measures and controls provide support for such access restrictions:

- Restrict the availability of this information to authorised users.
- Control the access rights of users.

There should be a mechanism to ensure the personalised identification of every user who tries to access this directory. A list should be maintained with such users and their profiles.

It is also important to limit the number of attempts to access this directory by unauthorised users.

Different authentication procedures may be used to substantiate the user's identity. Passwords are commonly used to identify and authenticate each user. Cryptography and authentication protocols may also be used.

File backup: Regular backup security copies should made of the instance documents generated by the system. A check should also be made to see that the security copy has been properly processed.

Physical security of the equipment: Proper controls should be in place to protect the computer that actually houses the instance documents generated. A secure environment should surround this important resource. Access to the room where the resource is located should be controlled at all times, and restricted only to authorised personnel. Such controls would include a personal ID card for authorisation and to be validated. An audible tape should be kept of every access, appropriately secured.

Access rights to the physical location of this resource should be reviewed and updated regularly.

Furthermore, the resource environment should be physically protected from any potential threat or risk of loss or damage. The installation and availability of the resource is equally important. Special control measures should be in place to ward against unauthorised access and to protect supporting systems, such as the uninterrupted power supply (UPS) and the cable infrastructure.

Encrypting the instance document: A cryptographic technique can be used to protect the confidentiality of the information.

An acceptable level of protection should be required based on a risk analysis, which will include the type and quality of the encrypted algorithm, as well as the length of the cryptographic keys employed.

Adoption of an encryption policy for such techniques should take into account different national regulations and restrictions legislated in other parts of the world relating to the encryption and decryption of data in international transmissions.

Contingency plan: A contingency plan implies an analysis of possible risks to which the directory and the information contained therein may be exposed. Such an evaluation should be performed, as well as a plan to reduce the risk and the procedures to follow in case of a problem.

A contingency plan should include a disaster plan in order to repair and restore the system as quickly and cheaply as possible, mitigating the loss of data.

4.5.3 Security of transmission of documents

For a secure channel of communication, the following is necessary:

- Full authentication on both ends of the communication (origination and destination).

- Confidentiality: Ensure that the information is accessible exclusively for authorised users.

- Integrity: Guarantee the exactness and completeness of the information and processing methods. In practice, to ensure that the information provided in outgoing file at origination is the same as the incoming file at destination.

- Availability: Ensure that authorised users have access to information and related assets whenever they need it.

- Non Repudiation: Ensure that neither end of the communication is able to reject the authorship of a transmission and its contents.

Security procedures:

Digital cerificate: File with information about the originator of the file, specifically to identify the system or application. The digital certificate provides authentication.

Encryption: Electronic information is changed into a secret code that only authorised users can interprete.

Digital signature: Sequence of digitised information added to a transmission or instance document that unequivocally identifies its author. The digital signature is a unique identifier that provides both authentication and integrity.

Access controls: Physical or logical access to information is not allowed to unauthorised users.

Integrity procedures: Control codes are added to a transmission to allow detection of any change or error in the content of the information sent.

Routing controls: Routing systems are used to protect information.

The following table outlines each procedure with the security service provided:

SERVICES	Authentication	Access Controls	Confidentiality	Integrity	Non Repudiation	Availability
PROCEDURES						
Digital certificate:	X					
Encryption:	X		X	X		
Digital signature:	X			X	X	
Integrity procedures:				X	X	X
Access controls:		X				X
Routing controls:			X			

Cryptography

Encryption methods used for digital signatures, digital certificates, and integrity procedures use the science of cryptography as their basis for encrypting and decrypting messages.

Cryptography is the science used for methods and mathematic techniques in the process of encryption or decryption of a message, or instance document, by means of an algorithm using one or more keys. There are various encryption systems that provide assurance for these services: confidentiality, integrity, availability, and non-repudiation of the sender or recipient of the message.

Encryption techniques may be broken down into two categories, depending on the type of key used.

- Symmetric cryptography
- Asymmetric or public key cryptography

Symmetric**:**

Characterised by the use of the same key to encrypt and decrypt.

The entire security process is based on the privacy of the secret key, and is called symmetric because it is the same key for both the sender and the recipient. These systems provide for confidentiality, but not authentication or a digital signature.

Symmetric algorithms are simpler than asymmetric, and that is why the process is less complicated and faster. The most frequently used algorithms are: DES, TDES, IDEA, RC5, RIJNDAEL.

Asymmetric:

Characterised by the use of different keys to encrypt and decrypt: one key is made public and the other key is private for each user. This system is based on every user having access to the public keys, but only knowing their own private key. It provides for confidentiality, authentication, and a digital signature.

The main drawbacks of this system: Difficult to use, and the process is slow.

The main advantage: Authentication and digital signature; no problem distributing the keys, as the public key is visible to everyone, and the private key is never transmitted.

The most used algorithm is RSA.

The algorithm DSS is used only for digital signatures, and has been adopted as a NIST (National Institute of Standards and Technology) standard.

The algorithm Diffie-Hellman is used to distribute symmetic keys, but does not provide for confidentiality, authentication, or digital signatures.

4.5.3.1 Description of the most used algorithms

Symmetric:

- DES (Data Encryption Standard): The most popular symmetric algoriithm in the world. It codes blocks of 64 bits using keys of 56 bits.

Advantages:

- Most prevalent algorithm.
- Very fast and easy to use.

Disadvantages:

- Key is short.
- Cannot use a variable length key.
- Algorithm may break when using a technique of differential cryptoanalysis.

- *TDES (Triple DES)*: Consists of using the algorithm DES several times with different keys. Used to avoid problem of the short key.

- *IDEA (International Data Encryption Algorithm):* Codes blocks of 64 bits using keys of 128 bits. Free of national restrictions and permits, and freely distributed over the Internet.

- *RC5 (algorithm developed by R. Rivest)*: Used by Netscape for its SSL (Secure Sockets Layer) security system.

- *RIJNDAEL (algorithm developed by V. Rijmen and J.Daemen):* Chosen in October 2000 by NIST to become Advanced Encryption Standard (AES) for use in non-military encryption applications. It is a block system of encryption, designed to manage long keys with variable blocks between 128 and 256 bits.

Asymmetric:

- *RSA (algorithm developed by R. Rivest, A. Shamir, L. Adleman)*: The most popular asymmetic algorithm. It is the fastest and easiest one to use. It is used to generate the encryption key used in SSL comunications.

- *DSS (Digital Signature Standard):* A system for digital signatures adopted by NIST as a standard. It uses the SHA (Secure Hash Algorithm) hash function and the DSA (Digital Signature Algorithm) asymmetic algorithm. DSA is an asymmetric algorithm that can only be used with a digital signature. It uses more parameters than RSA, which ensures a higher degree of security.

- *DIFFIE-HELLMAN*: Only used to exchange symmetric keys, but this is one of the main functions for asymmetric algorithms, and is widely used by Internet systems for confidentiality of symmetric keys (VPNs-Virtual Private Networks-, SSL, etc.).

Advantages and disadvantages of symmetric and asymmetric:

Symmetric:

Advantages:

- Encryption speed is fast, and systems with enough space for a long key are very secure.
- Allow authentication of MAC (Message Authentication Code) messages.

Disadvantages:

- Impossible to establish a distribution system and efficient key management system between senders and receivers.
- Lack of broad-based system of digital signatures.
- Security depends only on the secret key.

Asymmetric:

Advantages:

- Sender and recipient of a message use different keys for encryption, decryption, and signatures.
- Due to the complexity of the math used to define these keys, having a pair of keys with asymmetric algorithms is more robust than using symmetric algorithms.
- Allows for digital signatures: authentication of message and sender.

Disadvantages:

- Necessary to rely on certification mechanisms to ensure the accuracy of public keys: ACs (Attribute Certificates).
- Very slow encryption methods.

Hash functions:

One-way and irreversible encryption function. Used to guarantee the integrity of information in transit.

This type of algorithm does not have a decryption key, so there is no guarantee of getting the original string of the hash or checksum. The way to approach this algorithm is to produce collisions to test that the string produces the same hash value.

The most used algorithms are MD5 and SHA-1.

- *MD5 (Message-Digest Algorithm 5):* One of the most popular and widely-used algorithms for this type of function, due in great part to the inclusion of PGP (Pretty Good Privacy) in the first few versions. It processes incoming messages in blocks of 512 bits and produces outgoing messages in 128 bits.
- *SHA-1 (Secure Hash Algorithm-1)*: Produces signatures of 160 bits from blocks of 512 bits of the original message.

Digital signatures:

Digital signatures offer a data authentication service, ensuring that the message actually originated in a certain entity and has not been altered.

For reasons of efficiency, digital signatures have a hybrid nature: First, they use a hash function for data that will be signed, which generates a hash value for the message; then, an asymmetric signature algorithm is applied to the hash value , using the private key of the signer. The digital signature is formed by this signature hash algorithm, and attached to the message.

The testing process repeats the first step to obtain the hash algorithm for the message received, and then applies the asymmetric signature algorithm to the digital signature, using the public key of the signer. This results in, and is compared to, the original hash algorithm.

Any changes made to the message received will change the hash algorithm. In addition, any deliberate change to the signature hash algorithm is mathematically impossible without having access to the private-key.

4.5.3.2 Digital certificates

A digital certificate is essentially a public key and an identifier, issued digitally by a trusted certifying authority, and is used to confirm that a public key belongs to a specific user.

The internationally accepted standard for digital certificates is X.509, developed by the International Telecommunications Union (ITU).

Standard X.509 only defines the syntax of the certificates, so it is not bound to any one algorithm. It is based on the following fields:

Version – serial number of the certificate – identifier of the algorithm used for digital signature – name of the certifying authority – period of validity – name of user – public key of the user – unique identifier of the certifying authority – unique identifier of the user – extensions – digital signature of the above generated by the certifiying authority.

4.5.4 Communication channels

4.5.4.1 VPNs (Virtual Private Networks)

A virtual private network (VPN) is the extension of a private data network that includes links to a shared or public network like the Internet. With a virtual private network, it is possible to send data between two computers across a shared or public line that emulates a private end-to-end communication. This is possible by encapsulizing the data in a header that contains the routing information for the data to run over the public network and reach its destination as it were going through a tunnel. The data is encoded to ensure the confidentiality within the encapsulated connection.

Given a remote access and routing connections, an organisation may use VPN using long distance or local communications, or by using an Internet Service Provider (ISP).

The security of VPN remote access improves when using secure authentication with electronic certification.

Types of VPN:

- *Systems based on hardware*: These systems are routers that also encrypt. They are secure and easy to use. They provide good performance because they do not misuse the processor by overworking the operating system. This is dedicated hardware, very fast and easy to install.

- *Systems based on firewalls*: These systems use firewall software. They have the advantages of security mechanisms that use firewalls, including restricted access to the internal network/intranet. They can also translate NAT addresses (Network Address Translation) for an even better authentication.

- *Systems based on software*: These systems are ideal for situations where the two connection points of the VPN are not controlled by the same organisation, or when different firewalls or routers are used by each organisation. Traffic is sent through the tunnel by using addresses or protocols, whereas when using a VPN based on hardware, all the traffic must be routed through the tunnel. We can see a much easier and intelligent routing.

In reality, the two techniques most used to create VPNs are different protocols, or combinations of protocols: PPTP (Point to Point Tunneling Protocol) and L2TP (Layer Two Tunneling Protocol).

PPTP: Point to Point Tunneling Protocol:

PPTP is a protocol developed by Microsoft and available on all Windows platforms. It is simple and easy to use, but it provides less security than L2TP. The three basic characteristics of PPTP security are authentication, data encryption, and packet filtering.

- *Authentication.*

- *Data encryption:* Network packets are encrypted at the origination point of the connection, then travel through the tunnel, and are deciphered at the destination point. Like all data that flows through the tunnel, it is invisible to the rest of the world. Encryption of data in the tunnel adds an additional level of security.

- *PPTP packet filtering*: This option enhances the performance and reliability of network security, if activated in the PPTP server.

L2TP: Layer Two Tunneling Protocol:

This is an open standard available on most platforms, including Windows, Linux, Mac, etc. It uses IPSec (Internet Protocol Security) and provides a high level of security. Security certificates using public keys can be used to encrypt data and guarantee the user's identity over VPN.

Comparison between PPTP y L2TP:

- With PPTP, the encryption of data starts after the connection has been made (and, of course, after the PPP (Point to Point Protocol) authentication). With L2TP/IPSec, the encryption of data starts before the PPP connection is made by negotiating an association with IPSec security.

- The PPTP connections use MPPE (Microsoft Point to Point Encryption), an encyption method based on the algorithm developed by Rivest-Shamir-Aldeman (RSA) RC-4, that uses keys of 40, 56 or 128 bits. The L2TP/IPSec connections use the Data Encryption Standard (DES), with keys of 56 bits for DES or three keys of 56 bits for 3-DES. The data are encrypted in blocks (blocks of 64 bits in the case of DES).

- The PPTP connections only require authentication at the user level by means of an authentication protocol based on PPP. The L2TP/IPSec connections require the same level of authentication at the user level, but also need authentication of the system itself using digital certificates.

VPN Technology: Security of VPN

- IPSec (Internet Protocol Security) with encryption.
- SSL 3.0 (Secure Sockets Layer) or TLS (Transport Layer Security) with encryption.

IPSec with encryption:

- This is the most popular technique for VPNs. It supports a wide variety of encryption algorithms (DES, 3DES, AES, RC4), as well as data integrity algorithms (MD5, SHA-1). It also supports digital certificates X.509, and it offers a great variety of solutions in many modes: Client, Server, and Gateways.

Advantages of IPSec VPN:

- All types of IP and services are supported.
- Once the key exchange has been completed, many connections may use the tunnel set up.
- It is the same base technology to work between client-client, client-site, site-site.
- It supports robust authentication techniques and directory management.

Disadvantages of IPSec VPN:

- The VPN client needs to have software installed; not all operating systems can support this requirement.
- The VPN client needs to have the client configured before the tunnel is established.
- Connectivity may be affected negatively by NAT (Network Address Translation) or proxy mechanisms between the client and the gateway.
- Connectivity may be affected negatively by firewalls between the client and the gateway.

SSL VPN (Clientless VPNs):

- Currently, the SSL (Secure Sockets Layer, also called Clientless VPN) is competing with the traditional IPSec VPN for remote acess, or frequent usage of the extranet. SSL VPN is basically the possibility of getting VPN from a Web interface, via an encrypted format, to reach certain resources of the organisation.
- It supports a wide range of encryption algorithms (DES, 3DES, AES, RC4, DSA), in addition to data integrity algorithms (MD5, SHA-1)

Advantages of SSL VPN:

- HTTPS client is supported by all operating systems.
- The majority of the most popular applications for client/server systems support SSL.
- Operates transparently over NAT and proxy mechanisms.

Disadvantages of SSL VPN:

- Only supports TCP services, and frequently only HTTP or POP3/IMAP/SMTP over SSL.
- Does not use VPNs for site-to-site.
- Data is secure in transit, but there is no control over security of the data in the client's system.
- Multiple SSL sessions may be required for one session.

In response to the question "Should I use SSL VPN or IPSec VPN?", we have put together the following brief list of advantages of each technology:

SSL VPN is useful when:

- The traffic is HTTP or e-mail.
- Access is required to get information on any device (laptop, PC, PDA) from any part in the world (office, home, cybercafe) .
- It is not possible to install software with VPN access on the remote PC.

IPSec VPN is useful when:

- The organisation needs an infrastructure that accepts traffic from many protocols, not only HTTP and e-mail.
- The organisation has control over the remote access devices.
- There are strict security restrictions for remote access, for example, from cybercafes.

4.5.4.2 Web encryption

Enables a secure connection over HTTP for an application's environment.

Currently, the most used protocol is SSL (Secure Sockets Layer).

SSL provides security services using two different encrypting techniques: Cryptography for a public key (asymmetric) and cryptography for a secret key (symmetric). For the exchange of data between the server and the client, you use symmetric-encrypted algorithms, such as DES, Triple DES, RC2, RC4 or IDEA. For authentication and for encryption of the session key used by prior algorithms, you use a public key algorithm, typically RSA. The session key is used to encrypt data that comes and goes to the server, once a secure communication channel has been established.

In order to establish a secure communication using SSL, it is necessary to follow a series of steps:

First, it is important to establish the parameters that the session will use during a handshake or exchange of keys. Several objectives are met during the handshake: Certificates are used for authentication of the server, and at times of the client, to determine which encryption algorithms will be employed, and a secret key is generated to be used during the exchange of subsequent messages during the SSL communication:

- *Hello Client*: The purpose of "hello from client" is to inform the server which encyrpted algorithm is being used and to request a verification of the server´s identity. The client sends a set of encrypted, compressed algorithms that it supports and a random number. The purpose of the random number is so that in case the server does not have a certificate to confirm its identity, it can still establish a secure communication using a different set of algorithms. Within the set of cryptographic protocols, there is a protocol for key exchange that defines how the client and server can exchange information, as well as the secret key algorithms that define which methods may be used, and a one-way hash algorithm. Until this point, no secret information has been exchanged, only a list of options.

- *Hello Server*: The server responds by sending its digital identifier which includes its public key, a set of compressed cryptographic algorithms, and a random number. The decision of which algorithm to use is based on the most robust one that both the client and server can support. In some circumstances, the server may also request that the client identify itself using a digital identifier.

- *Client Approval*: The client verifies the validity of the digital identifier or certificate sent by the server. This is done by decrypting the certificate using the sender's public key and determining if this comes from a trusted certifying authority. Then, a series of verifications are made on the certificate, such as the date, URL of the server, etc. Once the authenticity of the identity has been verified, the client generates a random key using the server´s public key and the compressed encrypted algorithm previously agreed upon. This key is sent to the server and will be used for future messages during the session, if the handshake has been successfully executed.

- *Verification*: At this point, both parties know the secret key: the client because it generated it, and the server because it was sent and received by using its public key (the only possible way to decrypt it is by using the private key of the server). The last verification step is to confirm if the information transmitted until this moment has not been

altered. Both parties send a copy of prior encrypted transactions with the secret key. If both parties confirm the validity of the transactions, the handshake is complete. If not, the entire process must be repeated.

Now, both parties are ready to exchange information in a secure manner, using the secret key agreed upon and the compressed encrypted algorithms. The handshake is only done once and the secret key is used for the entire session:

- Exchange of data: Now that a secure SSL transmission channel has been established, it is possibe to exchange data. When either the server or the client needs to send a message to the other one, a checksum is generated (using a one-way hash algorithm agreed upon during the handshake), the message is encrypted and sent with the checksum; each messge is verified by using the checksum.

- Terminating an SSL session: When the client leaves an SSL session, the application usually shows a message warning that the connection is no longer secure and confirms that the client wants to leave the SSL session.

Advantages:

- Ease of use when deploying this solution.
- Minimum requirements (does not need a client) for starting a session.

Disadvantages:

- Only useable in a Web application (cannot be used for encryption of other protocols like POP3, FTP, ETC.).

4.5.5 Tools

Here are some security tools for XML:

XML Security suite: → https://secure.alphaworks.ibm.com/tech/xmlsecuritysuite

- This is a tool that provides security functions such as digital signature, encryption, and acess control over XML documents. They go further than protocols at the SSL transport level.

VORDEL DIRECTOR → http://www.vordel.com/products/vordeldirector/index.html

- VordelDirector provides security services for the network, and include encryption, digital signature, XML threat analysis for authentication, authorisation and logs. Designed to be used in service oriented architecture (SOA).

VORDEL SECURE: -> http://www.vordel.com/products/vordelsecure/index.html

VordelSecure is a gateway for XML, and serves as an intermediary element to filter XML traffic. When installed properly behind the network firewall in a DMZ (demilitarised zone), it is often called the "XML firewall".

4.5.6 Bibliography

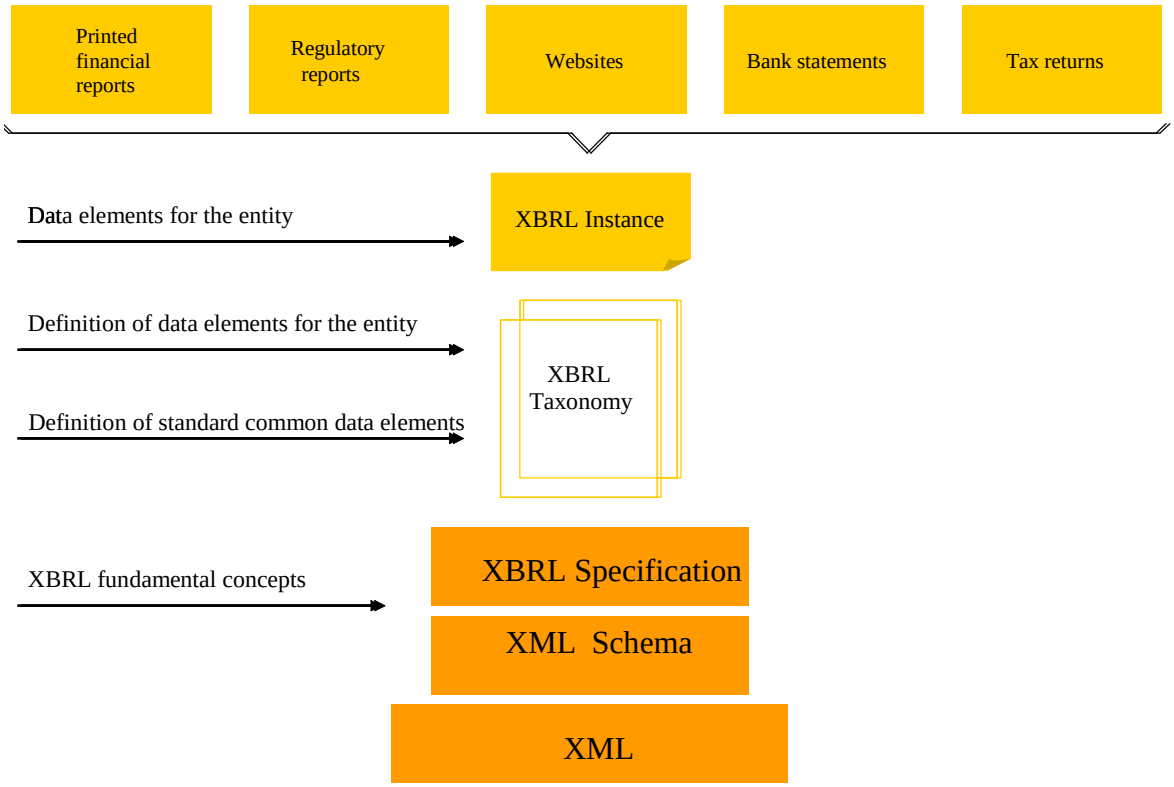
- VPN Technologies: Definitions and Requirements.
- IPSEC Versus "Clientless" VPN for Remote Access.
- Hackers 3: Secretos y soluciones para la seguridad de redes.
- Criptografía y Seguridad en Computadores. Manuel José Lucena López.
- Análisis de Seguridad de la familia de protocolos TCP/IP y sus servicios asociados. Raúl Siles Peláez.
- UNE-ISO/IEC 17799: Código de buenas prácticas para la Gestión de la Seguridad de la Información.
- Virtual Private Networks (VPN /PPTN) -http://www.helmig.com/j_helmig/vpn.htm
- Understanding Virtual Private Networks (VPN) http://www.sans.org/infosecFAQ/encryption/understanding_VPN.htm
- Understanding PPTP and VPNs - http://www.aliceinwondeland.com/library/website_archives/rhino9/pptp.htm
- PoPToP – The PPTP Server for Linux - http://poptop.lineo.com
- PPTP - http://www.cas.mcmaster.cal.~wmfarmer/SE-4C03-01/papers/Silva-PPTP.htm
- Acceso remoto por VPN (Red Privada Virtual) - http://www.uv.es/ciuv/cat/vpn
- Cisco – Layer 2 Tunnel Protocol - http://www.cisco.com/warp/public/cc/pd/iosw/tech/l2pro_tc.htm
- VPN FAQ - http://kubarb.phsx.ukans.edu/~tbird/vpn/FAW.html

4.6 XBRL Architecture

4.6.1 Reference architecture

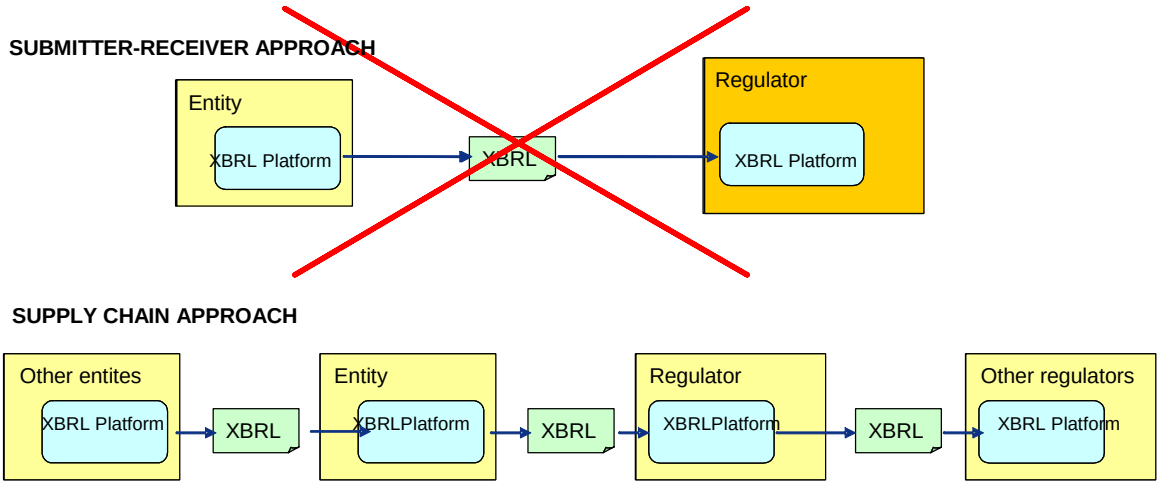
XBRL is a derivation of XML (eXtensible Mark-up Language). XML, as a platform-independent standard format for communicating, was developed to facilitate the exchange of information between applications over a corporate network or the Internet. XML is supported by practically all software vendors, and, as such, is emerging as the main way to transfer data.

XBRL was created specifically to optimize XML technology within a supply chain application for financial information.



An initial overview of the first approaches taken by entities to adopt XBRL as a standard to exchange financial information, basically driven by the regulatory agencies, is based on a scenario where there are entities that receive/collect XBRL data (regulators), and others that send/transfer XBRL data (submitters).

However, a closer look at the process reveals that the entity reporting to a regulator may also be receiving data from its own subsidiaries, and that the regulating agency may also need to consolidate data from its submitters and then report to another international regulator. Therefore, any organisation that processes financial data is a potential link in the supply chain of financial information. An entity may receive, generate, manipulate, and publish financial reports in XBRL format, and that is why it needs a system to handle these requirements.



Therefore, any system architecture that is designed to manage and process data in XBRL format should support the essential characteristics of financial reporting:

- Creation, distribution, collection, and manipulation of multiple taxonomies.
- Creation, publishing, receiving, validation and analysis of instant documents.
- Repository for storage and queries.

In addition, the system architecture may include the following complementary features:

- Security mechanisms: Athough it is true that the XBRL specification does not include any requirement or reference to security matters, the exchange of data in XBRL format does require, in the majority of cases, strict compliance with standards of confidentiality, integrity, authentication, and non-repudiation.

- Tools for developing taxonomies in a collaborative environment in order to complete the life cycle of the application.

- Tools to monitor and control the processing of XBRL instance documents.
  Flexibility to extract, convert, and adapt non-XBRL data to XBRL format.

- Analytical functions for XBRL information.

- Batch processing of XBRL instances.

A graphic representation follows:

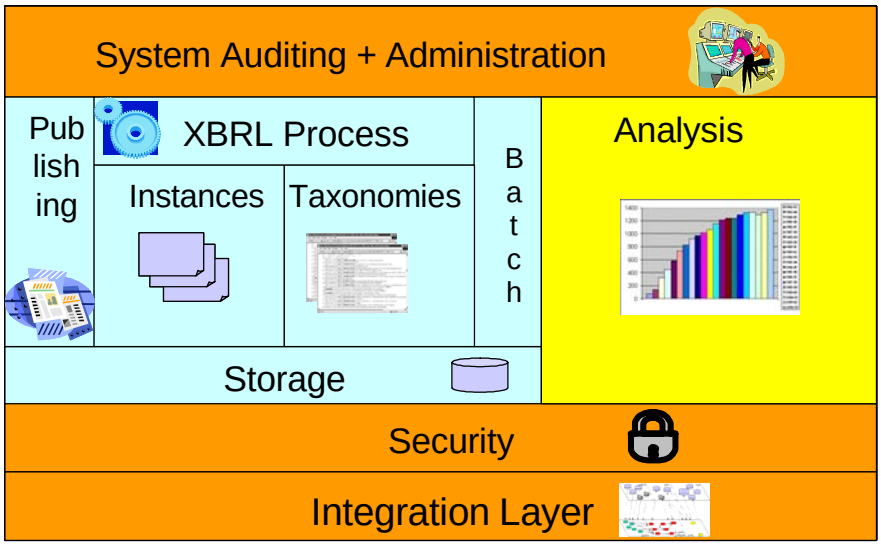


Figure 4.6-3

The functional architecture described above must be supported by a technical architecture that addresses the scalability and performance requirements. The following graph shows a proposed technical architecture that covers these requirements, based on a service-based, modular design.

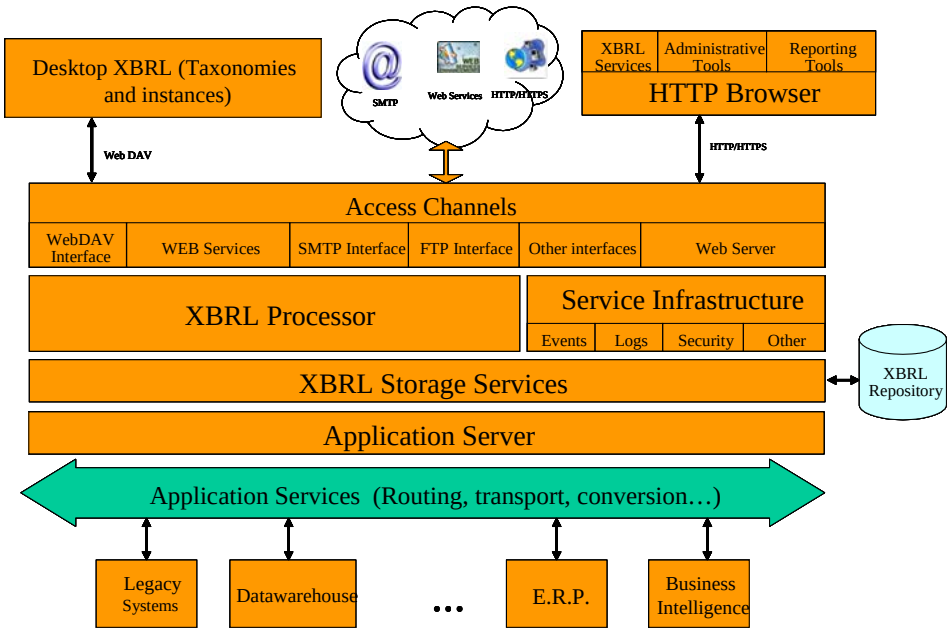


Figure 4.6-3

Here is a breakdown of the architecture by levels:

- *XBRL Desktop*: Groups together and integrates XBRL tools for the creation and editing of taxonomies and instances, in addition to other related utilities. These tools may work in a local mode or connected to a repository of XBRL documents residing on the server (WebDAV).

- *Web client*: Interface of the main user, offering:

- Management and control of services and operations directly associated with the processing of XBRL documents.
    - Tools for the design, generation and manipulation of XBRL documents.
    - Options for the administration and operation of the system.

- *Access channels:* Procedures set up for the collection and transfer of XBRL instance documents, in addition to loading and distributing XBRL taxonomies. They should support protocols like SMTP, FTP and HTTP(S), and provide a series of web services for the distribution of data.

- *XBRL processor:* This is the nucleus of the architecture. It provides the basic procedures to process XBRL documents.

- *Infrastructure services:* Group together all those services which support the functions that are not strictly XBRL related, such as:

- Notices, events and logs.
    - Procedures for security and control.

- *XBRL storage services*: Storage of XBRL documents, both taxonomies and instances, with the ability to manage historical data, version control and perform data searches. The availability and function of these services depend on the type of storage facility that has been set up, for example:

    - Object repository
    - Relational database
    - XML data repository
    - File system

    The level of granularity provided by the repository also depends on the type of storage facility. Granularity may be at the file level (schema and linkbases), or reach down to the level of an element in the taxonomy. It is also wise to provide for the storage of test cases, either in the same repository or in another place.

- *Application server:* The use of an application server is optional, depending on if the architectural modules require it or not.

- *Services bus:* Controlled by the application server, it brings together all the interfaces with external systems, such as the mainframe, datawarehouse systems, ERPs, business intelligence systems, etc. This level is for routing, transport, and converters for mapping, in addition to the extraction of non-XBRL data and transformation into XBRL.

4.6.2 Performance

Architectures designed for processing reports in XBRL format should be able to process large volumes of information at good performance levels.

With respect to managing the volume of data, the following considerations should be take into account:

- Taxonomies and their DTSs (*Discoverable Taxonomy Sets*) are created with multiple levels of imports, and, in some cases, with complex data models.

- International taxonomies are created with many elements (for example, the IFRS-GP taxonomy has more than 4.000 elements).

- Instance documents may be created with multiple contexts.

- For the same financial report, the XBRL version represents a volume increase of 3 to 1 when compared to the traditional (plain text) report.

- Although the financial reporting process does not imply a continuous stream of dataflow, it does tend to concentrate traffic on certain key dates.

Therefore, it is necessary to prepare for simultaneous processing of large-size XBRL instance documents.

Here are some design recommendations to optimize the performance of XBRL applications:

Use XBRL processors that provide the following:

- Possible reuse of data models for taxonomies. Expressed in another way, use cache mechanisms for taxonomies in memory: The creation in memory of the data model for a DTS is one of the most difficult parts associated with processing XBRL documents. By using this technique, the taxonomy will load up once in memory, and then be available to process all instances linked to it.

- Techniques for serialisation/deserialisation: An interesting alternative to caching taxonomies is to store objects in serial form. According to studies, the time to load up a taxonomy is directly proportionate to the square root of links to its linkbases, while the loading time for a serialised (deserialisation) taxonomy is directly proprotionate to the number of links to its linkbases.

- Possibility to precompile XML schemas.

- Techniques to efficiently access and save XBRL documents.

- Use of XML *parser* (DOM o SAX), depending on each situation.

- Only process the linkbases required for each situation.

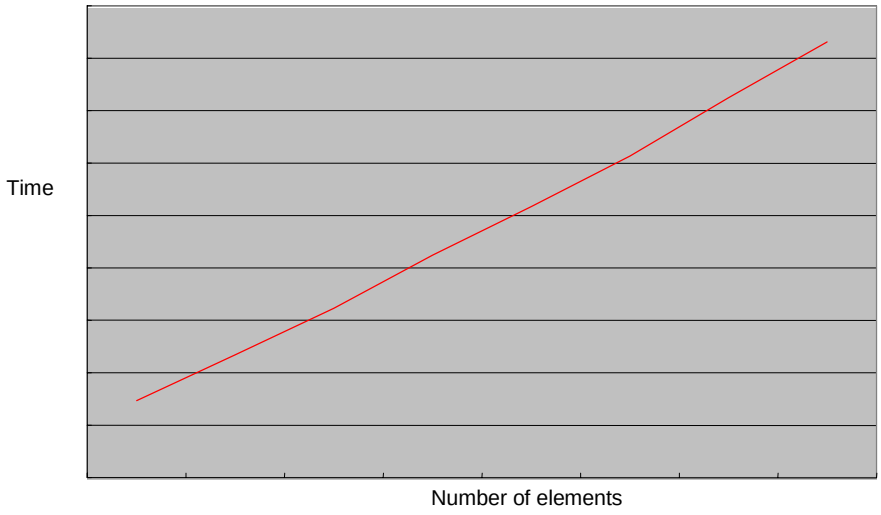
- For taxonomies that import other taxonomies, disable the inherited calculations that do not make sense in the new taxonomy (by using the attritubte *use = "prohibited"* in the calculation arcroles to be disabled). In this way, although the size of the linkbase increases, the validation time of the instances is considerably reduced.

- When selecting the type of repository for storing XBRL documents, consider the performance functions for managing XML and XBRL data.

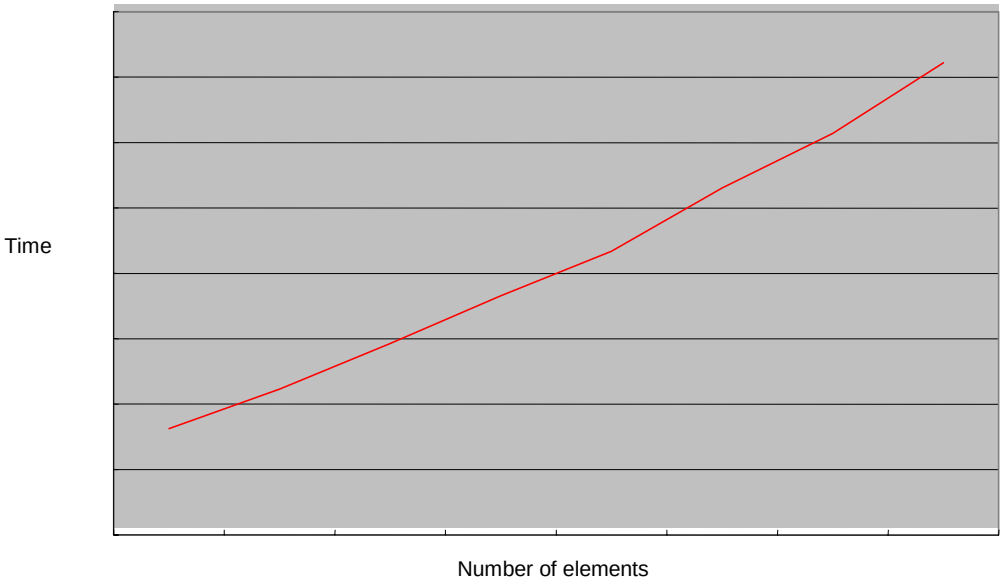
- Design scalable architectures.

- Minimise the length of the XML/XBRL labels when designing stylesheets (XSLT) for the transformation of XBRL documents.

1 LOADING (AND VALIDATION) OF TAXONOMY IN MEMORY.

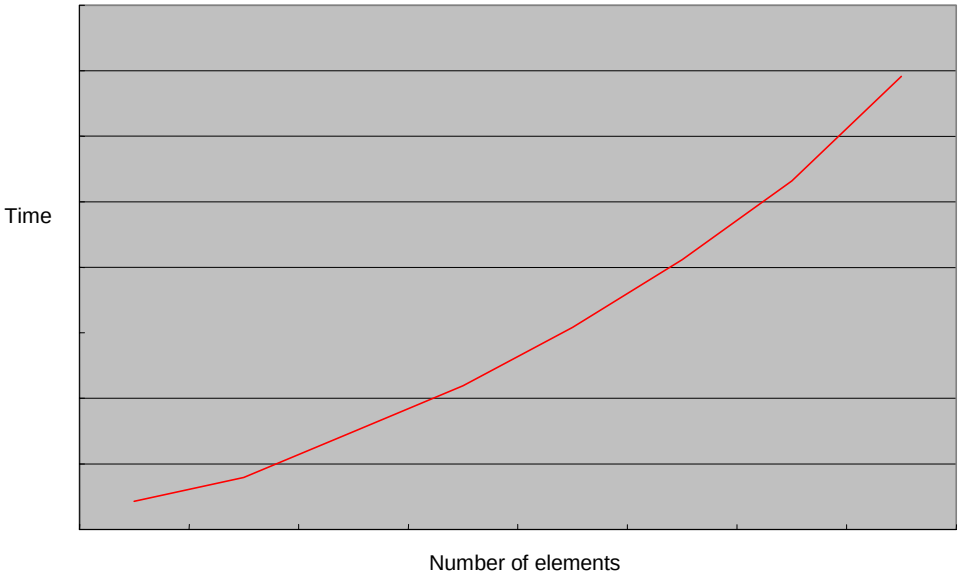


2 LOADING (AND VALIDATION) OF INSTANCE IN MEMORY.



3 SERIALISATION OF A TAXONOMY.

4 DESERIALISATION OF A TAXONOMY.



4.7 Versioning

Taxonomies may be modified from time to time for various reasons, such as:

- Changes in legal regulations relating to reporting requirements, such as in official forms or accounting standards.

- In the case of a taxonomy having been extended from another one, a change in the base taxonomy may originate a change in the extended one.

- Changes due to corrections originated by the technology used to build taxonomies.

- Changes in best practices, as based on supporting standards (FRTA, ISO, etc.).

- Change in the latest version of the XBRL specification, or the supporting XML specifications (XLink, XML Schema, etc.)

The reason for versioning taxonomies is to allow for comparison, analysis, and correct identification of the data that has been changed in the base XBRL documents of the taxonomies. Therefore, it is necessary to clearly document the information about the differences between them.

Those taxonomy users who automate the reporting process are directly affected by version changes because they have to make changes in their internal systems. Proper documentation of taxonomy changes allow such changes to be made in an orderly fashion from version to version, and there exists a history of which elements either change or disappear.

With regard to versioning, here are some best practices:

- Document any changes made between two different versions of a taxonomy, identifying each change or revision with a unique code that can be referred to in other documents (Control Log of Changes).

- Change the namespace of each new version of a taxonomy. The namespace is the unique identifier for the version of a taxonomy. Two versions of a taxonomy should not have the same namespace.

- Identify changes in the data dictionary of elements of the taxonomy with codes such as 'copied', 'eliminated', 'renamed', 'moved´, etc., in the control log of changes.

- Indicate the published launch date of the taxonomy in the file name the taxonomy, as well as in the namespace identifier.

- Indicate the version in the heading of the XML taxonomy documents: data dictionary and referenced linkbases.

- Indicate precisely which elements of the taxonomy documents have been affected by the version changes: linkbases and data dictionary.

- Specify which version of the standards (FRTA Candidate Recommendation 5, FRIS, etc.) and specifications (XBRL 2.1, Xlink 1.0) was used for the new version of the taxonomy.

- In the case of regulatory agencies, maintain a public respository of taxonomies that users may reference and use as a place to find all taxonomies issued to date.

- Ensure the stability, without variations, of the taxonomy during a reasonable time, such as a fiscal year.

5 Products and services

When an organisation is interested in implementing XBRL, the marketplace offers many solutions and tools specifically designed for the XBRL language, in addition to those already available for XML technologies.

These tools may vary by the technical level of a XBRL solution desired, and consequently the functionality required. For example:

- Tools for editing taxonomies, generating instance documents, etc.

- Tools for developing custom-made applications for processing XBRL documents.

- Global platform for financial reporting in XBRL, including the creation, administration, validation, management and storage of XBRL documents, as well as the preparation of business reports.

A few examples of the main vendors of XBRL tools:

Batavia **www.batavia-xbrl.com**
Batavia XBRL Products is a series of products for XBRL, based on Java. *Editor Online* for instance documents, API Java for XBRL applications, and the tool *ReportLink* for XBRL integration.

Blast Radius **www.blastradius.com**
Developed *Flexible v2.1-compliant XBRL processor* and other related products, such as *XBRL Web Express* (validates online instances), *XBRL Office Express* to import, create, validate, and export XBRL instances to MS Excel. Also offers *XMetal customization* for financial reporting using XBRL.

DecisionSoft **www.decisionsoft.com**
Provides products and services, such as *True North* for processing and validating XBRL documents. Also has an API for XBRL applications.

Fujitsu **software.fujitsu.com/en/interstage-xwand/activity/xbrltools/index.html**
Offers broad range of tools for the creation, editing of taxonomies and instances, plus the processing, validation and integration of XBRL documents.

Hitachi **www.hitachi.co.jp/XBRL**
XiRUTE ToolSet offers an API for application development using the XBRL DOM standard with the same interface as the W3C DOM.

Microsoft **www.microsoft.com/latam/office/solutions/xbrl/overview.mspx**
Has a prototype of *XBRL Office Tool* than can be used with Microsoft Office Word 2003 y Microsoft Office Excel 2003 to create and analyze documents formatted in XBRL.

Rivet Software **www.rivetsoftware.com**
Dragon Tag 1.0 is a tool to easily create XBRL instance documents from MS Word and Excel.

Semansys Technologies BV **www.semansys.com**

Offers solutions such as *Next Generation XBRL productivity tools,* for the creation, analysis and processing of taxonomies. Also provides solutions for XBRL integration and financial reporting.

Ubmatrix **www.ubmatrix.com**

Solutions for the reporting, validation, and analysis of XBRL documents. *Automator XBRL Professional* is for the creation of taxonomies, *UBS* is a Web Server for the integration of XBRL and *XBRL Converter*, and they also have an API for XBRL applications called *XBRL ToolKit*.

The following functions have been evaluated:

1 *Create and edit taxonomies*: Visualise, create and modify taxonomies.

2 *Create and edit XBRL instances:* Create and modify instances based on a specific taxonomy.

3 *Extraction of information from XBRL instances*: Process instance documents to find and collect information.

4 *Validation of XBRL instances*: Syntactic validation in accordance with XBRL specification (minimum conformance) and/or semantic validation conforming to a taxonomy (complete).

5 *Validation of taxonomies*: Syntactic validation of taxonomies.

6 *Screen view of XBRL* (XML-HTML): Ability to show HTML report based on presentation linkbase.

7 *Conversion of version 2.0 -> 2.1*: Ability to convert a document in specification XBRL 2.0 to XBRL 2.1.

8 *Integrated version control*: Maintain a history of XBRL documents and their modifications.

9 *Management of internal repository*: Ability to store XBRL documents in a repository.

10 *Integration with Excel*: Ability to export/import from MS Excel.

11 *Comparison of taxonomies*: Ability to see the differences between one taxonomy and another.

Products	Create and edit XBRL taxonomies	Create and edit XBRL instances	Extraction of information from XBRL instances	Validation of XBRL instances	Validation of XBRL taxonomies	Screen view of XBRL (XML-HTML)	Conversion of version 2.0 -> 2.1	Integrated version control	Management of internal repository	Integration with Excel	Comparison of taxonomies
Batavia XBRL		X	X	X							
Blast Radius (Xmetal) XBRL Office Express		X		X	X					X	
Blast Radius (Xmetal) XBRL Web Express				X	X						
DecisionSoft XBRL Toolkit API		X	X	X							
Fujitsu Interstage XBRL Processor		X	X	X							
Fujitsu Business Rule Editor	X					X				X	
Fujitsu Converter (Tool List v2.1)							X				
Fujitsu Intance Creator (Tool List v2.0)		X								X	
Fujitsu Intance Creator (Tool List v2.1)		X				X				X	
Fujitsu Taxonomy Diff	X										X
Fujitsu Taxonomy Editor (Tool List v2.0)	X										
Fujitsu Taxonomy Editor (Tool List v2.1)	X				X					X	
Fujitsu Validator (Tool List v2.0)				X	X						
Fujitsu Validator (Tool List v2.1)				X	X						
Fujitsu Versioning & Mapping			X					X			
Fujitsu XBRL Adapter		X								X	
Fujitsu XBRL Manager	X				X			X	X	X	
Fujitsu XBRL Sheet Mapping		X								X	
Hitachi System and Services (XiRUTE)	X	X								X	
Herramienta de MS Office para XBRL (Prototype)	X	X				X				X	X
Rivet Software		X		X						X	
Semansys Technologies XBRL Composer	X	X									
Semansys Technologies XBRL Integrator				X							
Ubmatrix Automator	X	X		X	X	X	X	X		X	
Ubmatrix Universal Business Server	X	X		X				X	X	X	
Ubmatrix Universal Converter		X								X	
Ubmatrix XBRL ToolKit	X	X		X						X	

6 Training

From a technology point of view, XBRL makes significant use of existing XML specifications. Chapters 3 and 4 of this paper have outlined the basic concepts of XML and XBRL.

In this chapter, we will give an overview of the structure of XML and related specifications. This information will serve as a basis for embarking on any XBRL project.

XBRL Spain has published a study on XBRL Training and Best Practices, which is available on the XBRL Spain website (www.xbrl.org.es). This document gives a detailed view of the training requirements for an XBRL project.

6.1 Technologies

XBRL uses the XML Schema specification for the structure of element definitions used in a taxonomy.

The XML Schema is found on the W3C website (http://www.w3.org/XML/Schema). Information on the XML Schema specification includes terminology, structure definition,

content and semantics for XML documents. The XML Schema was approved as a recommendation by the W3C on May 2, 2001.

The specification is presented in two documents:

- XML Schema Part 1: Structures. This part contains the core of the specification, describing the procedures provided by the XML Schema to represent XML structures. It is available in English at the following link: http://www.w3.org/TR/xmlschema-1/ .

- XML Schema Part 2: Datatypes. This part complements Part 1 and defines all the datatypes for elements that are permitted by the specification. It is also available in English at the link: http://www.w3.org/TR/xmlschema-2/ .

The specification for Namespaces is used in XML technologies to resolve the problem of a collision of XML documents, specifically for those XML documents that may clash because they use the same element names but with a different meaning. XBRL uses it extensively. It is located at: http://www.w3.org/TR/REC-xml-names/ .

XBRL uses the specification XLink to permit relationships between elements defined in the Schema, and this is possible by using XBRL Linkbases. Please refer to the W3C link: http://www.w3.org/XML/Linking

XPath is another W3C specification which is used in XBRL to process XML documents by referring to different parts of the document. It is a query language, and can be found at: http://www.w3.org/TR/xpath. XPath is a candidate technology to be used in the forthcoming XBRL formula linkbase.

XQuery is currently a Working Draft for a specification used for developing query languages for XML documents: http://www.w3.org/TR/2005/WD-xquery-20050404/.

6.2 XML Manipulation Techniques

XML documents must be accessed to obtain the data contained. The XML specification has sufficient information to be able to develop software applications for processing such content, specifically by using interpreters, or "parsers".

Parsers confirm that the XML documents are well formed, in accordance with the specification, and consequently the document will be validated if it provides proper information about its XML schema.

There are two types of parsers, DOM and SAX:

- DOM Parsers (Document Object Model)
    - DOM parsers scan the XML document and create an internal tree, based on the hierarchical structure of the XML data,. You can navigate and manipulate this tree from your software, and it stays in memory until you release it. Please refer to the W3C website: http://www.w3.org/DOM/.

- SAX Parsers (Simple API for XML)

- SAX parsers are based on events. The parser calls handler functions when certain events take place, as defined by the SAX specification. More information can be found at: http://www.saxproject.org/.

Parsers are found in all software applications, such as Java, .Net, PHP, etc. Here are some examples:

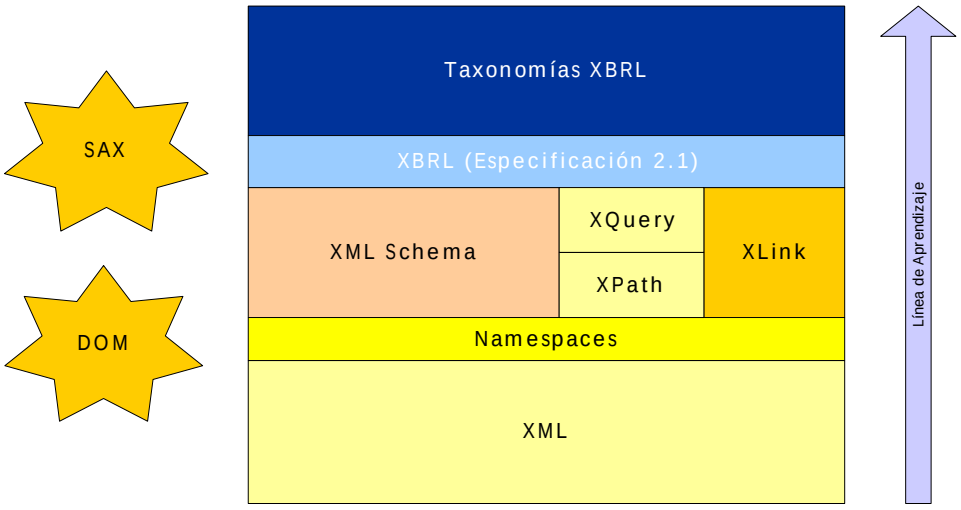
- Xerces (Apache)
- XJ (Data Channel)
- MSXML (Microsoft)

XBRL parsers can use the above methods, to which specific XBRL requirements are added.

XML technologies also include handlers that can transform an XML document into different formats. This technology is called XSL, which is an XML language for stylesheets that is used by a processing engine to perform the transformation of XML documents into different formats, such as another XML document, or PDF, CSV, XHTML, etc. This process may also be used with XBRL instance documents.

6.3 The learning curve

The following graph describes the learning curve for XBRL, starting with XML, and working up through Namespaces, XML schema, XLink, XPath, XQuery and the XBRL specification, to obtain an understanding of XBRL taxonomies.



7 Reference material and examples

7.1 Framework for use cases and implementation of XBRL in a service oriented architecture (SOA)

This example shows a series of steps and patterns used in the architecture of an XBRL project that has been developed for the purpose of automatically transmitting XBRL instance

documents, defining and distributing taxonomies, and transforming XBRL documents into specific back-office applications.

Due to the relatively short time that XBRL has been in use, and because we want to limit the scope of the project, we begin by outlining the major XBRL roles to be used in the system, as follows:

- Definition of XBRL taxonomies
- Creation of XBRL taxonomies
- Creation of XBRL instantes
    - Automatic creation
    - Manual creation
- Delivery of XBRL instance.
    - Automatic delivery
    - Manual delivery
- Analysis of information

The first two roles are developed by the users based on a requirements study of the business case, determining what information needs to exchanged, how it should be standardised, and how it should be implemented in an XBRL model. In short, we are referring to data modeling and its implementation.

Experience has shown that while the work of data analysis requires a good understanding of the business at hand, the actual implementation of XBRL is best done by computer specialists with experience in system analysis, using appropriate software development tools:

- Analysis and normalisation of data
- Definition of the data model
- Versioning requirements for the taxonomies under development
- Requirements of the working group
- Testing and validation of taxonomies under development
- Environmental requirements for development, pre-production, and production

Strictly speaking, these first steps pertain to the “XBRL Designers” and require regular software development tools, in some cases adapted to XBRL technology.

The remaining steps are directly related to the manipulation of operational information, i.e., the transmission of information, delivery and analysis of information, relating typically to the “XBRL Users”.

Once the designated parties and their tasks have been determined, related services and features are defined for the software application:

- Service oriented architecture.
- The application must be highly scalable and extensible because the volume of information to be exchanged, stored and analysed is potentially great.
- Isolation and encapsulation of specific features of XBRL for back-office applications.
- Independence of the XBRL specification to allow for modifications to be made with affecting the design of the software application.
- Independence of communication channels for entry and exit of XBRL instances.
- Incorporation of XBRL services in the service architecture of the general infrastructure.

These guidelines have led to the development of a series of central services and specialized tools for users to generate taxonomies and perform analyses.

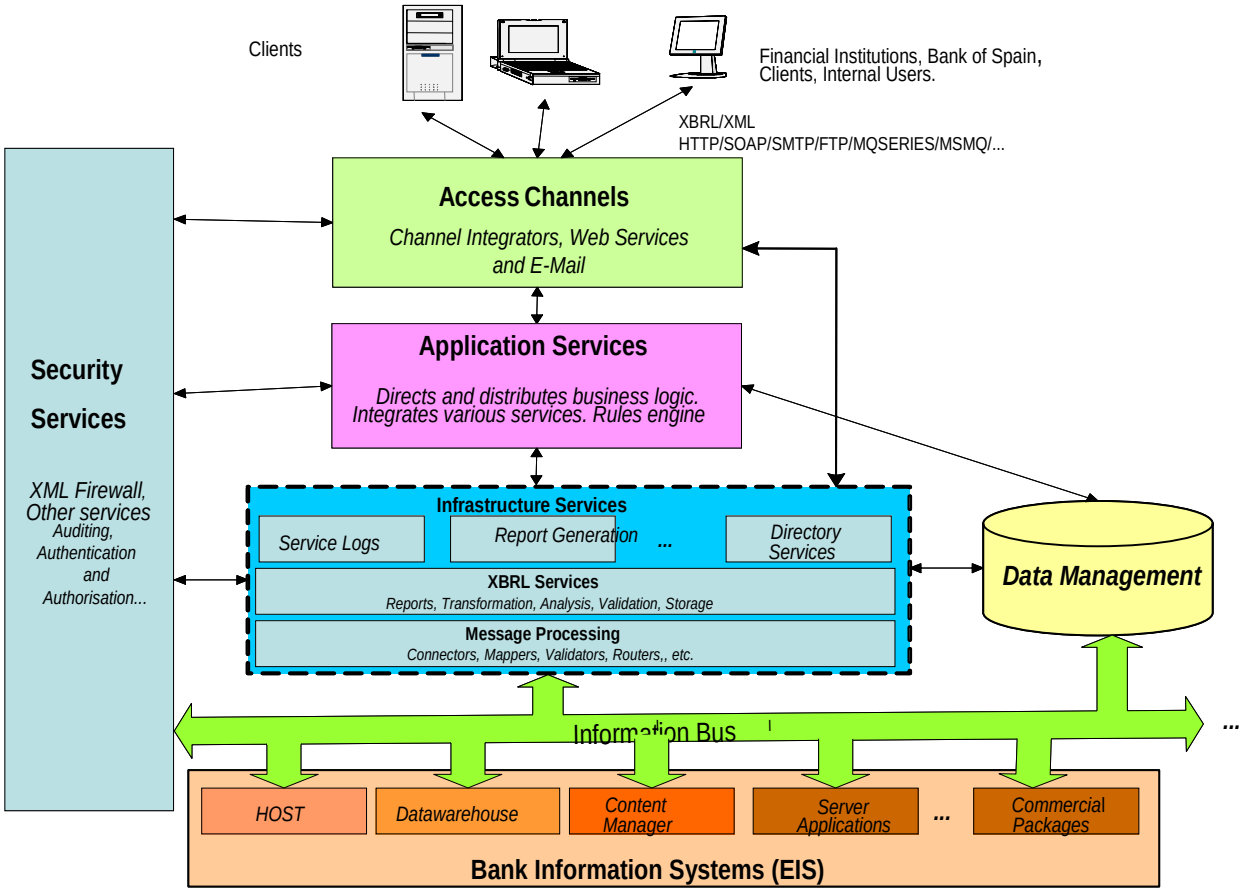
The central services are directed mainly at automating the processing of XBRL documents (generation of XBRL instances, validation of XBRL instances, reliable transmission and delivery of the data – encrypted and decrypted – storage of XBRL documents, etc.)

The following services have been developed:

- Services for taxonomies
    - Development of taxonomies
    - Publication and subscription of taxonomies
- Services for pre-validation and validation of XBRL instance documents
- Services for the transformation of XBRL instances into other friendly formats
    - By role, language…
    - To a flat file, or specific format for the back office…
- Services for general infrastructure
- Security (authentication, electronic signature, encryption and decryption, etc.)
- Services to manage the processes and integration of applications
- Reports for the user (HTML, PDF, Excel, …)
- Services to maintain data (databases for taxonomies and DTS, database for XBRL reports)
- Ensure the use of other services of the system

Considering the newness of XBRL, and the need for a project manager to be both the leader and proponent of the new technology, it was considered important to externalize some of the services for use by third-party data transmitters by giving them access to pre-validation, validation, and transformation of XBRL documents without the need to invest heavily in the technology.

The following diagram shows a high-level architectural rendering of the main services provided by the system.


Clients
Financial Institutions, Bank of Spain, Clients, Internal Users.
XBRL/XML
HTTP/SOAP/SMTP/FTP/MQSERIES/MSMQ/...
Access Channels
Channel Integrators, Web Services and E-Mail
Application Services
Directs and distributes business logic. Integrates various services. Rules engine
Security Services
XML Firewall, Other services
Auditing, Authentication and Authorisation...
Infrastructure Services
Service Logs
Report Generation
...
Directory Services
XBRL Services
Reports, Transformation, Analysis, Validation, Storage
Message Processing
Connectors, Mappers, Validators, Routers,, etc.
Data Management
Information Bus
...
HOST
Datawarehouse
Content Manager
Server Applications
...
Commercial Packages
Bank Information Systems (EIS)