UNIVERSIDADE NOVA DE LISBOA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

TUPLE-BASED MORPHISMS FOR INTEROPERABILITY ESTABLISHMENT OF FINANCIAL INFORMATION MODELS

POR:

MIGUEL ALEXANDRE SOUSA FERRO DE BEÇA

DISSERTAÇÃO APRESENTADA NA FACULDADE DE CIÊNCIAS E TECNOLOGIA DA UNIVERSIDADE NOVA DE LISBOA PARA OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA ELECTOTÉCNICA E COMPUTADORES

ORIENTADOR: PROF. DOUTOR RICARDO JARDIM GONÇALVES

CO-ORIENTADOR: MESTRE CARLOS AGOSTINHO

JURI:

PROF. DOUTOR JOSÉ BARATA

PROF. DOUTOR ANTÓNIO GRILO

PROF. DOUTOR RICARDO JARDIM GONÇALVES

MESTRE CARLOS AGOSTINHO

SETEMBRO 2010

UNIVERSIDADE NOVA DE LISBOA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

TUPLE-BASED MORPHISMS FOR INTEROPERABILITY ESTABLISHMENT OF FINANCIAL INFORMATION MODELS

POR:

MIGUEL ALEXANDRE SOUSA FERRO DE BEÇA

DISSERTAÇÃO APRESENTADA NA FACULDADE DE CIÊNCIAS E TECNOLOGIA DA UNIVERSIDADE NOVA DE LISBOA PARA OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA ELECTOTÉCNICA E COMPUTADORES

ORIENTADOR: PROF. DOUTOR RICARDO JARDIM GONÇALVES

CO-ORIENTADOR: MESTRE CARLOS AGOSTINHO

JURI:

PROF. DOUTOR JOSÉ BARATA

PROF. DOUTOR ANTÓNIO GRILO

PROF. DOUTOR RICARDO JARDIM GONÇALVES

MESTRE CARLOS AGOSTINHO

SETEMBRO 2010

ACKNOWLEDGEMENTS

I would like to express my appreciation to all those who, in some way or another, were involved in the making of this dissertation.

First, I would like to thank Prof. Dr. Ricardo Gonçalves for his assistance as dissertation advisor, as well as, for the encouragement and support which he provided through this journey.

Secondly, I would like to thank my colleagues at GRIS for their support. A special word of thanks to João Sarraipa for his close contributions and encouragement, and also, to Carlos Agostinho for his technical expertise and advice. Finally, I am also grateful to Pedro Maló for his support and for allowing me to coordinate my work at GRIS along with my dissertation.

Last, but not least, I would like to thank my family for all their support and encouragement throughout these long years at school.

ABSTRACT

The current financial crisis has demonstrated that there is a need for financial accounting data in a format which can be rapidly analyzed and exchanged. The appearance of XBRL in 2000 has helped create a ‘de facto’ standard data format for the exchange of financial information. However, XBRL by itself is not capable of ensuring a common semantic for the exchange of accounting information. Additionally, the existence of different accounting standards in different countries is a hindrance to efficient analysis and evaluation of companies by international analysts or investors. Therefore, there is a need to not only use a more advanced data format, but also for tools which can facilitate the exchange of accounting data, in particular when different accounting standards are used. This dissertation presents a tuple-based semantic and structural mapping for interoperability establishment of financial information models based on the use of ontologies and a ‘Communication Mediator’. It allows the mapping of accounting concepts of different accounting standards to be stored in the ‘Communication Mediator’. The mapping stored contains an ATL code expression, which with the aid of model transformation tools, can be utilized to perform the mapping between two different accounting models.

Keywords: XBRL, Ontology, ATL, Models, Model Morphisms

RESUMO

A presente crise financeira demonstrou que existe a necessidade de haver informação contabilística e financeira num formato em que possa ser rápidamente analisada e partilhada. O surgimento do XBRL em 2000 ajudou a criar uma norma de dados 'de facto' para a partilha de informação financeira. Contudo, o XBRL por si só não é capaz de assegurar uma semântica comum para a partilha de informação contabílistica. Adicionalmente, a existência de diferentes normas de contabilidade em diferentes países é um entrave a uma avaliação e análise de companhias eficiente por parte dos analistas e investidores internacionais. Assim, existe a necessidade não só de um formato de dados mais avançado, mas também de ferramentas que facilitem a partilha de dados contabílisticos, em particular, quando se utilizam diferentes normas de contabilidade. Esta dissertação apresenta uma forma de mapeamento semântico e estructural baseado em tuplos para estabelecer interoperabilidade de modelos de informação financeiros, sendo baseado na utilização de ontologias e de uma ontologia de mediação, a 'Communication Mediator'. A forma de mapeamento apresentada permite que o mapeamento de conceitos contabilísticos pertencetes a normas contabiliísticas diferentes possa ser armazenados no 'Communication Mediator'. O mapeamento armazenado contém uma expressão com código ATL, que com a ajuda de ferramentas de transformação de modelos, pode ser utilizada para executar o mapeamento entre dois modelos de informação contabilística diferentes.

Keywords: XBRL, Ontology, ATL, Models, Model Morphisms

CONTENTS

LIST OF FIGURES

LIST OF TABLES

SYMBOLOGY AND NOTATIONS

AICPA American Institute of Certified Public Accountants

API Application Programming Interface

ATL Atlas Transformation Language

CPA Certified Public Accountant

CRM Customer Relationship Management

CWM Common Warehouse Metamodel

ERP Enterprise Resource Planning

GUI Graphical User Interface

IASB International Accounting Standards Board

IDE Integrated Development Environment

IFRS International Financial Reporting Standards

M2M Model to Model

MDA Model Driven Architecture

MOF Meta Object Facility

MoMo Model Morphism

oAW openArchitectureWare

OCL Object Constraint Language

OMG Object Management Group

OWL Web Ontology Language

PDF Portable Document Format

PIM Platform-Independent Model

PSM Platform-Specific Model

QVT Query/View/Transformation

RDF Resource Description Framework

RDFS RDF Schema

SEC Securities and Exchange Commision

SPARQL SPARQL Protocol and RDF Query Language

UML Unified Modeling Language

URL Uniform Resource Locator

US GAAP United States Generally Accepted Accounting Principles

W3C World Wide Web Consortium

WWW World Wide Web

XBRL eXtensible Business Reporting Language

XFRML Extensible Financial Reporting Markup Language

XMI XML Metadata Interchange

XML Extensible Markup Language

1. INTRODUCTION

1.1. GENERAL INTRODUCTION AND MOTIVATION

The current financial crisis has its roots on human greed and irresponsibility on the part of many actors. It is not the first time that the world is faced with the dishonest actions of a few which caused an impact in the rest of the world.

The beginning of the current fallout has already been discussed many times over: financial institutions made loans to customers who could not afford to pay them back, and in turn, these financial institutions packaged and sold these loans to other financial institutions as securities. This "scheme" worked as long as real estate prices continued climbing. However, as housing prices fell, so did the value of these securities.

One of the main issues that faced regulators and financial institutions when the current crisis started to unfold was their inability to properly evaluate the market value of these securities. As each packaged security was based on hundreds of individual mortgages, the task of evaluating them was an extremely difficult one, and that is assuming that the financial institutions would be able to correctly identify the "contents" of each one of their securities.

However, this sort of accounting problem is just one side of the issue. One must not also forget the many accounting scandals which took place almost a decade ago. The early 2000's saw another barrage of financial and accounting scandals. The accounting frauds perpetrated by Enron and MCI are the first that come to mind. However, in 2002 alone, there were more than 20 known accounting scandals [1], ranging from inflated revenues and sales, to overstated assets and understated liabilities.

In response to these, stricter accounting rules and regulations, such as the "Sarbanes-Oxley Act", were passed into law in the United States. Unfortunately, a few years later, the United States and the rest of the world face an even bigger financial crisis, which has not only proved that the earlier regulatory mechanisms were not enough, but that other methods may have to be used to prevent future financial calamities and restore investor confidence in the financial sector.

In the article "How to Fix Financial Reporting", Bogoslaw [2] discusses some of the measures that could be implemented in order to help restore confidence in the equity markets. From changes in accounting rules, to increased transparency in the derivatives market, one of the solutions according to this article is the move towards digital financial data [2]:

"Financial reporting also needs to move away from big documents to electronic database formats that relieve analysts of the need to re-enter all the numbers in their own spreadsheets, and would free up analysts and regulators to actually analyze the data, says

Philip Moyer, chief executive of EDGAR Online (EDGR), a Web site that publishes corporate filings to the SEC."

This move towards digital financial data has already started in 2000, with the introduction of eXtensible Business Reporting Language (XBRL), a standard specification which is mainly used to exchange accounting information, such as financial reports and statements.

As it shall be discussed further ahead, XBRL has brought tremendous advantages as it has become the 'de facto' standard format for exchanging accounting data. However, some challenges still remain, mainly regarding the semantics of the contents being exchanged.

An important distinction must be made between the contents being exchanged and their respective semantics. Just because data can be exchanged between two systems with the same data format it does not necessarily mean that each system will give the same meaning to the data being exchanged. This is referred to as 'semantic heterogeneity.'

This issue, in the context of exchanging XBRL data, is raised by Madnick and Zhu [3] and they refer to it as "Ontological Heterogeneity", which is "the case where the elements in different taxonomies that appear to refer to the same concept actually have subtle differences." The authors show that the concept of 'Operating Profit' contained in XBRL taxonomy according to US GAAP, may have some differences from its Chinese equivalent, 'Ying Ye Li Rum', as exemplified below:

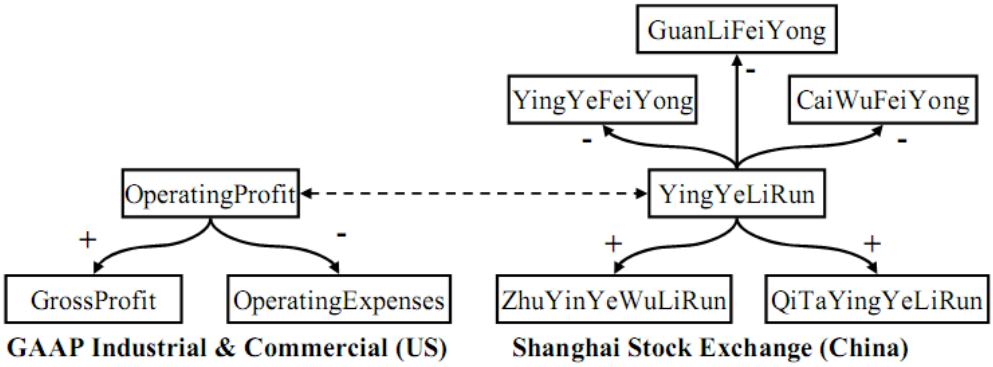

FIGURE 1.1– ONTOLOGICAL HETEREGONEITY EXAMPLE [3]

This issue arises because of the differences in accounting standards between two countries. Although XBRL may be used to represent a taxonomy according to many different accounting standards, currently there are no tools which allow the correct mapping of accounting data defined according to different standards.

Therefore, the exchange of accounting data still requires a set of tools which will enable the mapping or the transformation of data between different accounting standards.

1.2. PROBLEM DESCRIPTION AND CONTEXT

This dissertation intends to focus and address the problems related to the interoperability establishment of financial information models. Thus, its focus is to propose a solution that could help on solve interoperability issues related to the exchange of the current existent financial information models. The proposed technical solution is mainly related to business information models (e.g. XBRL) and information semantic solutions (e.g. taxonomies; ontologies; semantic mappings; model morphisms).

The dissertation proposes a solution, which is demonstrated by the development of tools which can assist in the mapping and transformation of data, in particular, accounting data represented according to two different accounting standards. However, such results of the work performed, may be transferred to other domains, such as the exchange of product data.

Ontologies are used to build an information model which allows the exploration of the information space in terms of the items which are represented, the associations between the items, the properties of the items, and the links to documentation which describe and defines. Thus, this dissertation studies ontologies and their capacity to represent different accounting standards, but also as an active mechanism to solve the existent interoperability issues mainly related to semantics. One of its main uses in this dissertation is for the representation of semantic mappings and model morphisms.

Regarding model morphisms, the present dissertation builds upon the “Knowledge Enriched Tuple”, a formalism which has been defined, as a means of representation of semantic mappings. In the context of the solution to be presented, this formalism is used to exemplify how it can be used to store mappings between two different ontologies, in way which can also be utilized by other applications or even model transformation code generation.

Ontology mapping is something that can be achieved already through various existing tools. However, according to [4], the variety of ontology languages, as well as, the diversity of ontology tools presents a serious problem of lack of interoperability. Additionally, [5] states the following: “There are many existing ontology development tools, and they are used by different groups of people for performing diverse tasks. Although each tool provides different functionalities, users tend to use just one tool, as they are not able to exchange their ontologies from one tool to another.”

Therefore, there is a need for some harmonization with regards to ontology mapping, a need which can be fulfilled through the use of Model-Driven Architecture (MDA) paradigms.

In the particular context of ontology mapping, the model transformation languages, such as ATL, provided by the MDA paradigm can serve as tool which can assist in the mapping of concepts between two different information models. The proposed solution aims to provide a bridge between the conceptual representation of the model mappings, as described above, and

translate it into model transformation code, which can be utilized to perform the actual transformation with existing instances of those models.

1.3. RESEARCH METHOD

The present dissertation will use the following scientific research method [6]:

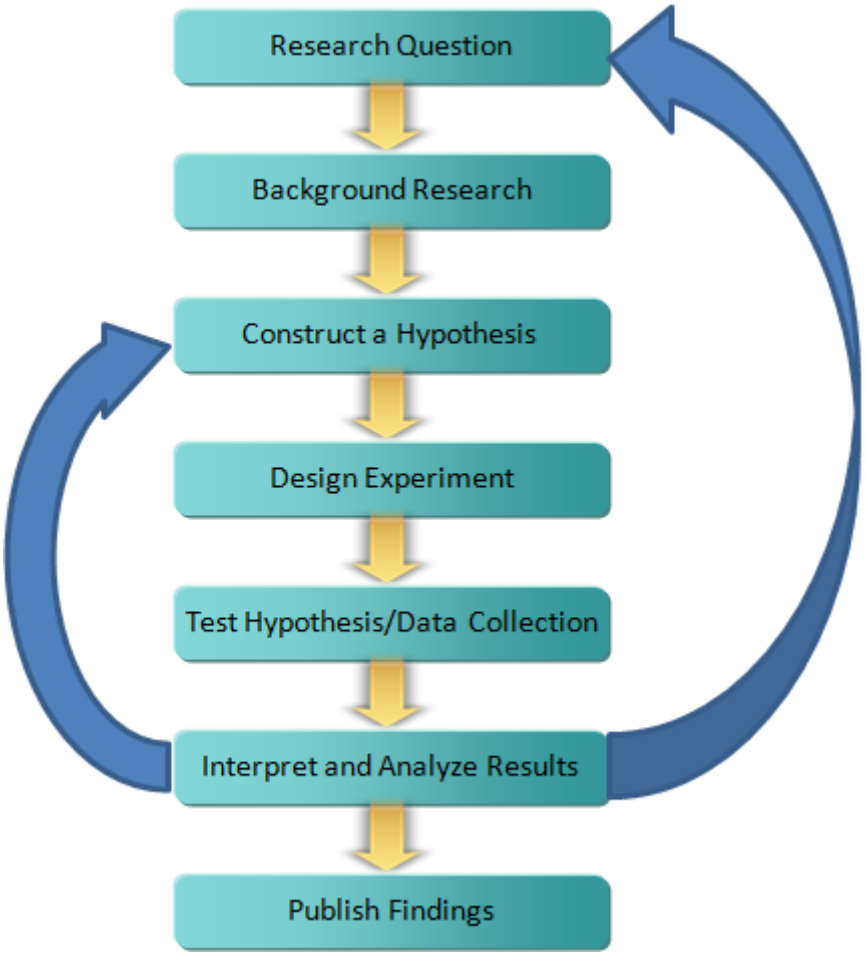

FIGURE 1.2 – STEPS IN THE SCIENTIFIC METHOD

1. Ask Research Question – The first step in the method, is also the most important, as it defines the boundaries of the work to be performed. It cannot be too broad otherwise, it becomes unfeasible or unreasonable to reach a conclusion. It must be focused and measurable, meaning that the claim being researched must be answered in a way that is logic and asserted with concrete facts. The research question is usually accompanied by some secondary questions, which can assist in narrowing the focus. The research question for the present dissertation is defined in sub-section 1.4.
2. Perform Background Research – There is no need to reinvent the wheel in science, therefore, all scientific work must be built upon previous work. Hence, the second step of the scientific method consists of performing background research on the subject under question. This research consists of developing a state-of-the-art on relevant matters which relate to the research question. Once one has a good grasp of what has

been accomplished so far in the subject, one is therefore much better able to determine what and how the research work being developed will complement the work done previously, and how it will serve to advance the knowledge in the fields associated with it. The background research consists mostly of literature review of papers, articles, books and other relevant materials which are available on the subject under question. The background research is detailed in sections 2, 3, 4 and 5.

3. Construct a Hypothesis – Once sufficient knowledge on the subject has been obtained it is then possible to formulate a hypothesis, based upon the information collected in step 2. The hypothesis is then a compilation of facts, which serve as the basis for further study. It aims to predict a specific outcome and it must be brief, focused and stated in a declarative form. The purpose of the hypothesis is to give clarity and focus to the research problem. The hypothesis for the present dissertation is defined in sub-section 6.1.3.
4. Design experiment – This step consists in outlining what is necessary to properly evaluate the hypothesis under consideration. Within the realm of computer science, this usually requires the design of a prototype or the development of a system architecture. In order to be valid, the experiment must be measurable, meaning that under this step, it is important to identify which variables will be under evaluation. Finally, the validation of the hypothesis must be planned in such a way which can be easily replicated by others. The theoretical design and the proof-of-concept implementation are described in section 6.2.
5. Test hypothesis / Collect Data: Once the experiment has been designed, it is then necessary to perform the testing of the hypothesis and gather all relevant data. In order to properly test the hypothesis, a battery of tests should be defined and performed. For each test scenario, data should be collected to perform the validation of the hypothesis. The test methodology is discussed in section 6.3.
6. Interpret and analyze the results: Once all the tests have been performed and the associated data has been collected, it is then necessary to interpret the results. It is at this stage where it is possible to evaluate whether the hypothesis under evaluation is valid, or if the test data is unfavorable, if it needs to be refined or completely redefined. In some instances, it may even be necessary to reformulate the research question, and start from the top. If the test results do validate the hypothesis, it is then possible to move on to the next step. The interpretation of analysis of the test results is detailed also in section 6.3.
7. Publish findings: The final step in the scientific method is the publication of the findings which results from the research work developed. The findings publication serves as a contribution to the scientific community and can be used to further advance other relevant research in the subject field. In accordance to the type of research performed, the findings should be published in scientific papers.

1.4. Research Problem and Questions

The research problem and questions to be addressed by the present dissertation are:

- Shall it be possible to achieve harmonization of global accounting standards through the interoperability of accounting data, achieved through model transformation and supported by ontologies?

1.5. Hypothesis

The hypothesis to be validated by the dissertation is the following:

- XBRL can be the reference standard data format for the exchange of accounting data between global accounting systems.
- It is possible to enhance the interoperability between heterogeneous accounting systems through the use of an ontology-based harmonization tool.

1.6. Dissertation Outline

The dissertation begins with an overview of Financial Reporting and XBRL as a Financial Reporting Data Standard. This section provides an introduction to the financial reporting domain, as well as, a detailed description on the inner workings of XBRL. The description of XBRL serves a means of better understanding the capabilities and, more importantly, the limitations of XBRL and, therefore, the need to adopt a different information modeling mechanism The next section, 'Knowledge Representation & Interoperability Semantics', discusses the various knowledge representation paradigms available, and how ontologies are the most appropriate mechanism for the representation of financial reporting data. Then, the dissertation describes the major types of Model Morphisms and introduces the "Communication Mediator" ontology, as mechanism to be included in the proposed solution, when integrated in a Model-Driven Architecture, focusing on how Model Transformation Languages can assist in implementing model morphisms.

Section 6 discusses the proposed solution for validating the hypothesis raised, based on the technologies described in the earlier sections, followed by a discussion on the Proof-of-Concept implementation, by elaborating on the actual technology solutions used and respective implementation steps. Finally, it describes the testing procedures which were performed to validate the implementation of the proposed solution.

The final section of the dissertation discusses the conclusions reached and foreseen future work.

2. Overview of Financial Report & XBRL as a Financial Reporting Standard

2.1. Overview of Financial Reporting

2.1.1. Introduction to Financial Reporting

The accounting software in use by most large companies today, is extremely sophisticated. Usually, Enterprise Resource Planning (ERP) applications, such as SAP, provide a variety of modules that integrate many types of data, and which also integrate themselves with other company applications, such as Customer Relationship Management (CRM) applications.

However, this sophistication has yet to make its transition to other types of accounting information to be used by other parties, in particular, financial accounting reporting data.

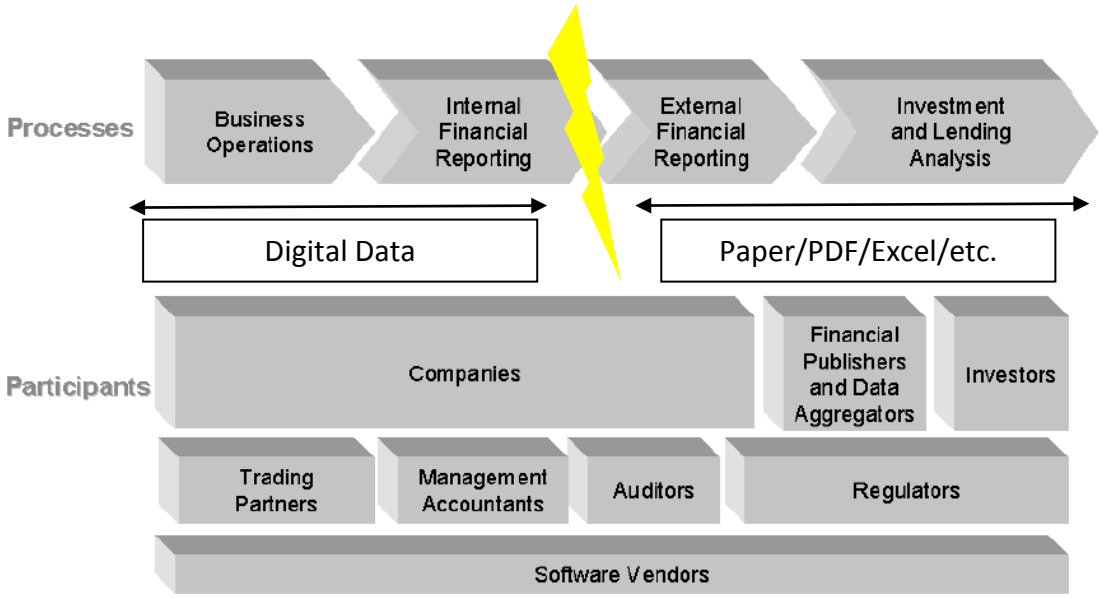

FIGURE 2.1 - BUSINESS REPORTING SUPPLY CHAIN [7]

Figure 2.1 exemplifies the business reporting supply chain from a 'traditional perspective'. The sophistication of IT accounting systems has for many years been limited to the internal processes of the reporting supply chain. Within company walls, accounting data flows in digital format.

However, when accounting reporting data must be released to external parties, the flow of digital data is stopped. Data from accounting systems must be converted into other formats, traditionally a paper report.

Even with the advent of the Internet, a few more options appeared, as most companies now publish their reports in a digital format, usually in PDF file format, while some even provide accounting data such as, Balance Sheet, Income Statement and Cash-Flow Statement in Excel, thus saving investors and analysts some precious time while doing their own analysis.

Besides the potential for errors, there is above all, the time spent unnecessarily, copying data from one type of document to another. David Harper, an investment analyst for Harbinger Research claims that, “crunching numbers and generating graphs occupies at least 50% of my time” when writing a company report. [8]

Additionally, even digital formats, such as PDF and Excel files, do not allow much in terms of customization nor do they provide much assistance in terms of their integration with other tools and/or other types of data. The typical “cut & paste” of data is still prevalent when one has to perform financial analysis using these formats.

Also, the information provided is static, not dynamic, meaning that it soon gets outdated, as market conditions change every day. The financial information provided by a public company in December, can be quickly outdated by the end of January.

Finally, one must also take into account that certain companies, such as multinationals, may be subject to multiple accounting standards. For example, a Portuguese company listed on the New York Stock Exchange is required to file financial reports according to IFRS, while it may also be required to provide financial information according to the Portuguese GAAP. In this scenario, although the raw data is the same, it must, however, be presented according to two different standards.

2.1.2. How XBRL can assist financial reporting

XBRL is a digital data format which aims at solving many of the problems of today’s financial reporting, while adding new opportunities for more advanced tools of financial analysis.

By becoming a data standard, XBRL will enable the appearance of XBRL-enabled tools to analyze XBRL data, thus taking away the need to import data from one digital format to another. As other tools also become XBRL compatible, this will allow for even further integration.

In addition to the above, XBRL is dynamic as data published in XBRL can be generated directly from a company’s internal accounting system. The paradigm of continuous accounting, which shall be discussed below, can become a reality.

Finally, XBRL offers flexibility, as different taxonomies can be used, thus facilitating the generation of financial reports according to various accounting standards.

2.1.3. Customizable Data, Continuous Accounting & Auditing

As discussed in the previous section, digital data flows freely within a company’s accounting system. At any time, an accountant or any member of the management team can get access to real-time information on a company’s financial situation.

Again, as it was already mentioned, today's financial reporting system does not provide the same sort of ability to those outside the company. At best, analysts and investors are only able to obtain financial reports on a quarterly basis, at best.

The shift to an electronic data format, such as XBRL, could enable a paradigm shift in terms of seamless access to data.

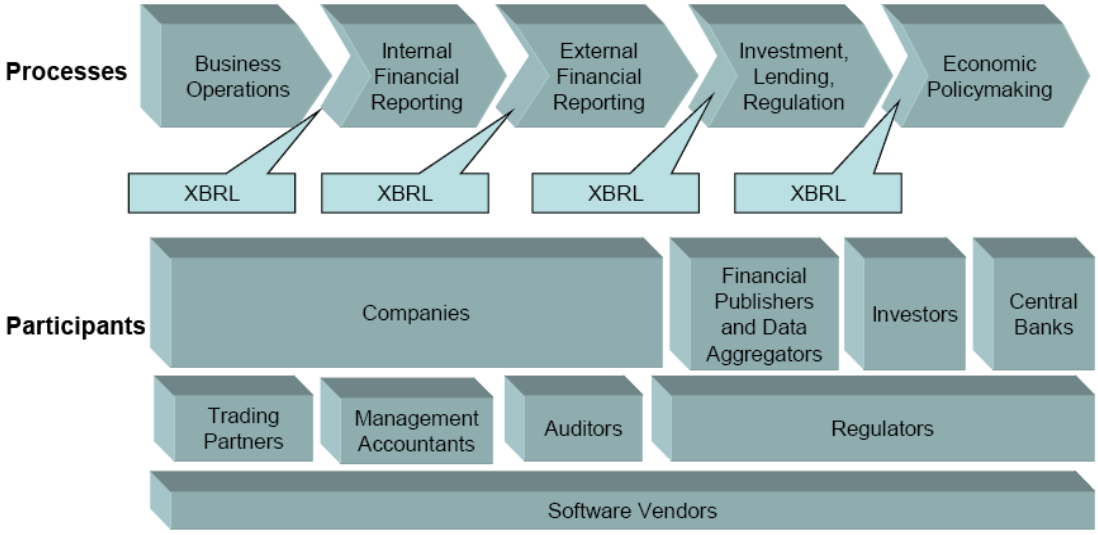

FIGURE 2.2 - BUSINESS REPORTING SUPPLY CHAIN ENABLED BY XBRL [9]

The 'XBRL-enabled' reporting supply chain, shown in Figure 2.2, exemplifies what could be the future of accounting reporting.

By utilizing XBRL as the data exchange standard in IT accounting systems, both within the company and outside, here are some of the consequences of this paradigm shift:

1. Customizable accounting data – In utilizing a common standard for accounting data, there would be no more need to convert internal accounting data into external accounting reports. The source of data would be the same, however it would be customizable according to the needs of each participant. For example, analysts would obtain financial reporting data, whilst regulators could have access to more in-depth information financial data.
2. Real-time accounting data could become a reality – With the implementation of an XBRL reporting supply chain, the possibility of real-time accounting data [10] may become a reality. Thus, instead of financial reporting data only being available on a quarterly basis, such as, the typical quarterly report, investors, analysts and regulators could have access to real-time accounting information in the near future. The implications of such a paradigm shift could be tremendous, as companies would be forced to have greater transparency. However, as it was the case with many accounting scandals, there are times when companies may wish to cut back on their information

disclosure. But, on the other hand, investors would benefit from this, as greater transparency would enable them to make more sound investment decisions.

3. Continuous auditing – Rezaee, Elam and Sharbatoghlie define continuous auditing as "a process of gathering and evaluating evidence to determine the efficiency and effectiveness of Real-Time Accounting in safeguarding assets, maintaining data integrity, and producing reliable financial information" [10]. If the XBRL enabled business supply chain is to become a reality, the auditing process must also shift from "the manual audit of accounting systems with paper documentation to on-line, continuous electronic audit of EDI, paperless systems." [10] The implications of this change are tremendous as continuous auditing will allow auditors "to test a larger sample (up to 100 percent) of clients' transactions and data faster and more efficiently", as well as, reduce the time and costs required by audits and, it may "also increase the quality of financial audits by allowing auditors to focus more on understanding a client's business and industry and its internal control structure" [10].

A concrete example of how continuous reporting can assist companies is provided in a study by Searcy, Ward and Woodroof [11]. In their paper, "Continuous reporting benefits in the private debt capital market", the authors found that "that high risk companies providing financial information to the lender on a daily basis have a higher probability of loan acceptance than do companies providing financial information to the lender on a quarterly basis." [11] Indeed, this shows how greater transparency can be beneficial to companies by providing greater confidence to third-parties, such as, lenders, regulators and investors.

2.2. XBRL AS A FINANCIAL REPORTING DATA STANDARD

2.2.1. BRIEF INTRODUCTION AND HISTORICAL BACKGROUND

eXtensible Business Reporting Language (XBRL) is a standard specification, based on XML for the purposes of exchanging accounting information, more specifically, financial reports and statements [12], as well as, other non-financial information. Some examples include the following [13]:

Financial information:

- Balance Sheets
- Cash flow Statements
- Income Statements

Non-financial information:

- Performance Measurements
- Regulatory Reporting Forms
- Loan Applications
- Statistics

The beginnings of XBRL can be traced back to April 1998, when Charles Hoffman, a Certified Public Accountant (CPA), started investigating the usage of eXtensible Markup Language (XML) in financial information reporting. Mr. Hoffman started by developing prototypes of financial statements and audit schedules in XML. [14]

In July 1998, Mr. Hoffman contacts the American Institute of Certified Public Accountants (AICPA), in particular, the AICPA's High Tech Task Force, regarding the potential of XML in financial reporting. [7]

The objective was to engage the AICPA in developing a "standard, XML-based language for digitizing business reports in accordance with the rules of accounting in every country around the world." [15]

Just two months later, a Product Description is crafted, which proposes the creation of a prototype set of financial statements. After receiving funding, just one month later, the prototype is completed in December 1998. Then, in January 1999, it is presented within the AICPA, which recognizes the value and the importance of the standard to the accounting profession.

In June 1999, a business plan is created for the development of XFRML (Extensible Financial Reporting Markup Language), and in July 1998, the AICPA agrees to provide funding for the new endeavor. Soon after, twelve companies joined in as members of the Steering Committee, including Deloitte & Touche, Ernst & Young, KMPG, PricewaterhouseCoopers and Microsoft, amongst others.

The organization changed its name to XBRL Steering Committee in April 2000, and the first XBRL specification was released in July 2000. XFRML, the code name for the language, would now become XBRL.

Today, the XBRL Consortium has more than 550 members, including companies, governmental entities, universities and other institutions. These members are spread out through several established jurisdictions. Jurisdictions are composed by all the members involved in the development or deployment of XBRL at a national level. There already fifteen countries with established XBRL jurisdictions, while seven more countries have Provisional jurisdictions.

2.2.2. TECHNICAL SPECIFICATIONS OF XBRL[1]

XBRL reports are made with separate entities, each with its specific purposes. As it shall be seen further ahead, it is this separation between components that gives XBRL much of its appeal.

[1] The structure of this section is based on IASB's 'XBRL Fundamentals' web page [17].

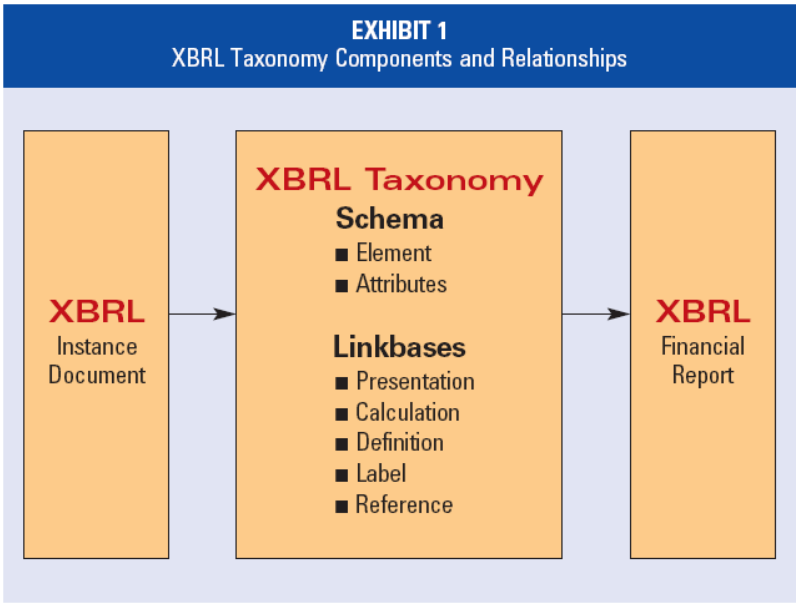

FIGURE 2.3 - XBRL REPORT COMPONENTS [16]

As figure 2.3 exemplifies, the two main components of an XBRL report are:

1) Taxonomy – In its essence, it provides the structure of the financial report. It includes the definitions of the concepts that are to be included in the report, as well as, the relations between them. However, the taxonomy document does not include any values for any of the concepts [13]. Some examples of some concepts that can be included in a balance sheet report, may include:
    - Assets
    - Currents Assets
    - Accounts Payable
    - Equity
2) Instance Document –Basically, an instance document is a collection of facts, which in this case, are the values for the concepts that are defined in the taxonomy. However, the instance document does not include any information regarding the relations between the facts. As mentioned previously, all this is contained in the taxonomy document.

This separation of a business report into two entities is what gives XBRL great flexibility. For example, a company may need only to develop a taxonomy document once, as the definitions will most likely be the same, year after year. Once a taxonomy document is developed, a company filing an XBRL report needs only to create an instance document with the new values.

Additionally, this also allows for the creation of national taxonomies, whereby all firms in a country can submit their financial reports according to the standard national taxonomy. However, given the specificities of certain sectors, there are usually several national taxonomies, each specific to a sector, such as banking, industry and others.

Finally, the use of standard taxonomies allows for easy comparison of financial reports belonging to different firms.

2.2.3. Components of Taxonomy Documents

A taxonomy derives its name from the Greek, being the junction between the verb tassein (to classify) and nomos (law or science) [17].

Therefore, a taxonomy would mean the law or science of classification, however, it is usually interpreted as being the classification of knowledge in a particular domain.

In the XBRL environment, the taxonomy is the classification of the various accounting terms in a structured way.

XBRL Taxonomies can be decomposed into the following:

1) Taxonomy Schema – It includes the definitions of the elements that are contained in the taxonomy.
2) Taxonomy Linkbases – The Linkbases, on the other hand, include the relationships between the different elements of the taxonomy.

This separation between definitions and Linkbases is traced back to the XML standard, which specifies such separation.

2.2.4. Taxonomy Schema

As mentioned above, the taxonomy schema contains the definitions of all the elements that make up the taxonomy, and their respective attributes such as their identifications, names and other characteristics.

The schemas are created according to the XML Schema format and are usually stored as an .xsd file. In the particular case of XBRL, the schemas are tailored to the specific needs of financial reporting.

The code example below will assist in illustrating the structure of an XBRL taxonomy schema:

```
<schema
xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:xbrli="http://www.xbrl.org/2003/instance"
xmlns:link="http://www.xbrl.org/2003/linkbase"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:ifrs-gp="http://xbrl.iasb.org/int/fr/ifrs/gp/2005-05-15"
xmlns:ifrs-gp-rol="http://xbrl.iasb.org/int/fr/ifrs/gp/2005-05-15/roles"
xmlns:samp="http://www.iqinfo.com/xbrl/taxonomy"
targetNamespace="http://www.iqinfo.com/xbrl/taxonomy"
elementFormDefault="qualified"
attributeFormDefault="unqualified">
```

The schema always begins with the root element, the tag <schema> and always ends with the closing tag </schema> (not shown). Following this, there are several namespaces.

The namespaces help identify where the various elements derive their definitions from. For example, the definition of “Assets” in US GAAP may be slightly different than the one provided by the Australian GAAP. Therefore, the namespaces must be unique and that explains why they are defined with Uniform Resource Identifiers (URI’s), similar to the Uniform Resource Locators (URL’s) that are common for locating web addresses.

As it can be seen from the example above, the namespaces included in an XBRL schema are various, each with its own specific role, as described below:

- The ‘xmlns’ namespace is a reference to the XML Schema, since XBRL derives from XML.
- The ‘xmlns:xbrli’ namespace contains the elements and attributes that are unique to XBRL. It extends the XML Schema and it is required in all XBRL taxonomies.
- The ‘xmlns:link’ namespace is the XBRL implementation of XML linking. It is required in all XBRL taxonomies, as it used to relate the various XBRL elements contained in the taxonomy.
- The ‘xmlns:xlink’ namespace refers to the native XML Linking elements and attributes. It is different from the XBRL linking discussed above and it is also required.
- The ‘xmlns:ifrs-gp’ namespaces are references other taxonomies. In this example, the taxonomy being described will be defined as an extension to the IFRS taxonomy.
- The ‘xmlns:samp’ is the namespace for the taxonomy under definition. When one builds a new taxonomy, one is also building a new schema which must also have a unique reference. This namespace will then be the reference for the new concepts that will be defined in this new taxonomy.
- Finally, the ‘targetNamespace’ declares the ‘samp’ namespace defined previously. Both of these must be unique to each taxonomy schema.

2.2.5. TAXONOMY ELEMENTS

In addition to the root element, the schema contains the definitions of the various elements.

Basically, an XBRL element is a business concept, such as “Assets”, “Cash” or “Liabilities”, defined according to a set of rules and having certain characteristics, as illustrated below:

<element name=”Assets” id=”Assets” periodType=”instant” balance=”debit” abstract=”false” substitutionGroup=”item” type=”monetaryItemType”/>

The element definition is composed of several attributes, the most relevant being:

- ‘element name’ – The name must be unique, as there cannot be two different elements with the same name. Additionally, the name must meet certain requirements, as it

cannot have spaces or other reserved XML characters (such as <, > or /). Finally, names are case sensitive in XML, meaning that 'Assets' is a different element from 'assets'.

- 'periodType' – This attribute refers to the " accounting distinction flows and resources." (IASB n.d.) Since Assets are valued at a specific point in time (i.e. year-end Balance Sheet), the periodType is considered to be 'instant', thus referring to a particular point in time. On the other hand, Cash Flow elements would have 'duration' as periodType.
- 'balance' – This attribute defines the balance nature of the element, according to double entry accounting rules. Assets appear as a debit on the Balance Sheet, hence the respective 'debit' attribute. Obviously, on the other hand, a Liabilities element would be characterized as a 'credit'.
- 'Abstract' – This attribute defines whether this element is to be used in calculations or not. For example, an element such as "Balance Sheet" could be defined in order to assist in the visualization of the final report. However, the element "Balance Sheet" would not have any monetary value associated to it, as it is only meant to be used as a label and it would therefore be described as an abstract element.
- 'substitutionGroup' – This attribute defines whether the element is an 'item' or a 'tuple'. An 'item' refers to a single reporting concept, while a 'tuple' refers to a collection of items. A possible example of a tuple element could be the company information, such as company name, company address, company telephone, etc.
- 'type' – Finally, the type attribute refers to the kind of value that will be stored in the element. In this case, the element will contain a monetary value, hence the 'monetaryItem Type'. There are several other types, such as 'sharesItemType' for shares based values, 'decimalItemType' for decimal values or stringItemType for string values, among others.

2.2.6. TAXONOMY LINKBASES

Besides the schema, as it was mentioned earlier, any XBRL taxonomy contains linkbases. The elements contained in the Schema do not contain any information whatsoever regarding their hierarchy or relationships with other elements. Therefore, it is the role of the linkbases to establish the relationships amongst the various taxonomy elements.

As stated in the IASB XBRL site [17], "the creation of an XBRL taxonomy, regardless of its purpose, also involves performing (the) following actions:

- labelling elements in specified languages in order to make taxonomy readable for humans;
- referencing elements to the external resources that justify their existence and that contain an explanation, definition or example of the use of the particular financial concept,

- defining relations between elements according to different criteria."

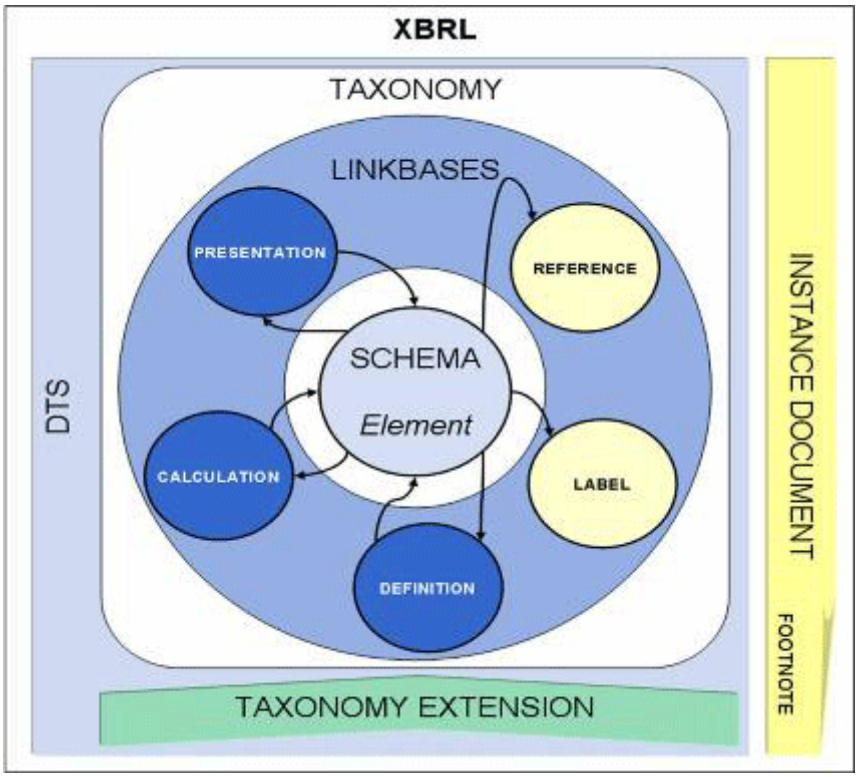

FIGURE 2.4 - THE ROLE OF LINKBASES IN A XBRL TAXONOMY [17]

Figure 2.4 provides a graphical representation of the role of the various linkbases.

For instance, the calculation, presentation and definition linkbases are described with bidirectional arrows, meaning that they help define relationships between elements.

On the other hand, the reference and label linkbases are represented by unidirectional arrows, meaning that they provide links to resources that are external to the taxonomy.

Linkbases are based on two XML languages:

- XLink – Also known as XML Linking Language, XLink "allows elements to be inserted into XML documents in order to create and describe links between resources." [18]
- XPointer – Or, XML Pointer Language, which "allows for examination of a hierarchical document structure and choice of its internal parts based on various properties, such as element types, attribute values, character content, and relative position." [19]

Basically, in order to create a relation, we need to point to elements or resources that we are interested in and define the type of relationship. A simplified example of a hierarchical relation from a presentation linkbase is provided below [17]:

```
<loc xlink:type="locator"
 xlink:href="schema.xsd#Assets"
 xlink:label="Assets_Locator"/>
```

```
<loc xlink:type="locator"
 xlink:href="schema.xsd#CurrentAssets"
 xlink:label="CurrentAssets_Locator"/>

<presentationArc xlink:type="arc"
 xlink:arcrole="http://www.xbrl.org/2003/arcrole/parent-child"
 xlink:from="Assets_Locator" xlink:to="CurrentAssets_Locator"/>
```

In this particular example, two locators are created using Xlink. Locators help identify reporting concepts in a taxonomy. These locators indicate that the elements 'Assets', in the first locator, and 'Current Assets', in the second locator, are in the schema.xsd file, under the references '#Assets' and '#CurrentAssets' respectively. Each locator has also its own label, in this case, 'Assets_Locator' and 'CurrentAssets_Locator'.

The last section of code describes the relationship between the two elements. The 'presentationArc' reference means that this is a presentation linkbase relationship, whereas the 'arcrole' attribute defines the relation as a 'parent-child'. The last line of code refers that the 'Assets' element (from) is the 'parent' of the 'Current Assets' element.

There are five different types of linkbases in a XBRL taxonomy:

- Presentation
- Calculation
- Definition
- Label
- Reference

Each of these linkbases will be explained in further detail in the following sections.

2.2.7. PRESENTATION LINKBASE

The presentation linkbase, as its name implies, stores the relations between the various taxonomy elements, so that they may be properly organized within the taxonomy. By organizing the various taxonomy elements, they can then be presented in a structured way.

For example, a Balance Sheet contains information regarding Assets, which may be decomposed into Current Assets and Non-current Assets. As seen in the example above, the presentation linkbase will store parent-child relations, defining the relation between these two elements.

2.2.8. CALCULATION LINKBASE

The calculation linkbase is used to add some sort of validation rules to XBRL reports. This linkbase contains "basic validation rules, which apply to all instance documents referring to a particular taxonomy" [17]. Whereas the presentation linkbase provides a hierarchy of how the elements should be presented, the calculation linkbase provides a structure where lower level

elements are summed or subtracted from one another, so that the higher level element is the result of these calculations.

The following is a basic example of how the calculation linkbase works:

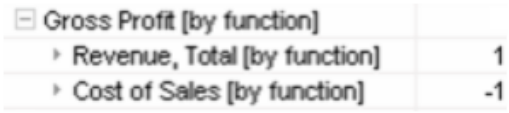

FIGURE 2.5 - CALCULATION LINKBASE EXAMPLE [17]

In the example shown in Figure 2.5, the high-level element 'Gross Profit' is obtained by the addition of 'Total Revenue', which is represented with weight '1', and the subtraction of 'Cost of Sales', which is defined with a '-1' weight.

The XBRL code for the above example is as follows [17]:

```
<calculationArc xlink:type="arc"
 xlink:arcrole="http://www.xbrl.org/2003/arcrole/summation-item"
 xlink:from="GrossProfit" xlink:to="RevenueTotal"
 order="1" weight="1" use="optional"/>

<calculationArc xlink:type="arc"
 xlink:arcrole="http://www.xbrl.org/2003/arcrole/summation-item"
 xlink:from="GrossProfit" xlink:to="CostOfSales"
 order="2" weight="-1" use="optional"/>
```

One must notice the similarities between the calculation linkbase and the presentation linkbase. The calculation relation is defined with a 'calculationArc', where the 'arcrole' attribute is defined as a 'summation-item'. However, the negative weight in the 'CostOfSales' is what indicates that this element must be subtracted, instead of added to the 'Gross Profit' element.

While getting a better understanding of the calculation linkbase, one also gets a better idea behind the need for the presentation linkbase:

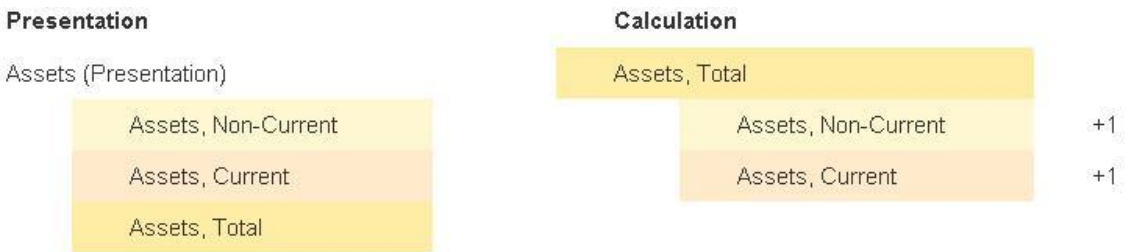

FIGURE 2.6 - DIFFERENCE BETWEEN PRESENTATION AND CALCULATION LINKBASES [17]

As stated earlier, in the calculation base, the high-level element is the sum of all lower level elements. However, the way a financial report is structured, requires that Total Assets appear at the bottom of all elements, thus to indicate that is a sum of the elements above it.

Despite the added advantage of having some data validation built-in into the XBRL taxonomy itself, the calculation linkbase does have its limitations. The major limitation behind the calculation linkbase is that it does not allow elements which have different 'periodType' attributes. One significant example of this limitation is for example, the inability to perform calculations between elements from the Balance Sheet, which have an 'instant' 'periodType', with elements contained in the Income Statement or Cash-Flow Statement, which have a 'duration' 'periodType.'

However, in June of this year, the XBRL consortium released a new Formula Linkbase, XBRL Formula 1.0, which makes it "possible to define logical and mathematical relations expressing sophisticated rules and checks between business concepts." [20]

Given its recent introduction, it may take still a few more months before existing taxonomies are revised and updated to include this new linkbase.

2.2.9. DEFINITION LINKBASE

The definition linkbase allows the ability to create relationships which are not covered by either the calculation or presentation linkbases.

The four types of standard relationships supported by the definition linkbase are:

- 'general-special' – this type of relationship helps distinguish between concepts that have a generic or a more specific (special) meaning. For example, 'ZIP Code' is a specific definition of the more general concept of 'Postal Code'.
- 'essence-alias' – This relationship allows matching different elements that have same meaning. For example, an airline company may use the concept of 'Planes' in its taxonomy, while another airline may use 'Aircraft', even though for accounting purposes, the concepts are the same and they can be used interchangeably.
- 'similar-tuples' – This relationship is similar to the essence-alias. While the 'essence-alias' applies only to single elements, the 'similar-tuples' applies to two different tuples that have equivalent meanings.
- 'requires-element' – As the name itself states, this sort of relationship is used when the value of one element requires the value of another element. An example of this relationship can be when a regulatory authority may require the disclosure of certain component of Assets, if that particular component appears on the Balance Sheet.

2.2.10. REFERENCE LINKBASE

The purpose of the reference linkbase, as stated by its name, is to provide a reference mechanism to the elements contained in the taxonomy.

Most taxonomies are based on existing accounting standards or regulations, be they international, such as the IFRS, or national, such as the POC in Portugal. Hence, XBRL taxonomies allow for the inclusion of a reference to those specific documents or items, in order to provide some sort of guidance to would be XBRL instance creators.
Examples of reference relationships are shown below [17]:

```
<reference xlink:type="resource"
  xlink:role="http://www.xbrl.org/2003/role/presentationRef"
  xlink:label="CashFlowsFromUsedInOperationsTotal_ref">
      <ref:Name>IAS</ref:Name>
      <ref:Number>7</ref:Number>
      <ref:Paragraph>14</ref:Paragraph>
</reference>
```

```
<reference xlink:type="resource"
  xlink:role="http://www.xbrl.org/2003/role/measurementRef"
xlink:label="CashFlowsFromUsedInOperationsTotal_ref">
      <ref:Name>IAS</ref:Name>
      <ref:Number>7</ref:Number>
      <ref:Paragraph>18</ref:Paragraph>
      <ref:Subparagraph>a</ref:Subparagraph>
</reference>
```

The first segment of XBRL code refers to a presentation reference, meaning that it references a document that explains “how and where the element should be presented in terms of its placement and labeling” [17]. In this case, we can find that reference in the IAS 7, paragraph 14.
The second type of reference relationship is a measurement reference. Therefore, the resource referenced will explain how the value for that particular element is determined and how it should be calculated.

2.2.11. LABEL LINKBASE

Finally, the label linkbase provides a mechanism to make XBRL taxonomies “multi-lingual”, meaning that a single element can have multiple labels, in various different languages, as shown below [17]:

```
<label xlink:type="resource" xlink:role="http://www.xbrl.org/2003/role/label"
 xlink:label="ifrs_AssetsTotal_lbl" xml:lang="en">Assets, Total</label>

<label xlink:type="resource" xlink:role="http://www.xbrl.org/2003/role/label"
 xlink:label="ifrs_AssetsTotal_lbl" xml:lang="de">Vermögenswerte, Gesamt</label>

<label xlink:type="resource" xlink:role="http://www.xbrl.org/2003/role/label"
```

xlink:label="ifrs_AssetsTotal_lbl" xml:lang="pl">Aktywa, Razem</label>

The three label relationships above all refer to the same element, ‘Assets’, where the first label is in English, followed by the German and Polish translations.

2.2.12. TAXONOMY EXTENSION

As its name implies, the first attribute of XBRL is its “extensibility”, meaning that given the variety of industries and businesses, some companies are required to modify existing “standard” taxonomies (i.e. national taxonomies, IFRS taxonomies) in order to suit their own particular needs, such as adding new elements not contained in the original taxonomy, or changing the relationships between existing elements, with regards to their order, addition or deletion.

XBRL does allow such flexibility, however such extensions must not modify the base taxonomy. This is usually not possible, because as seen in the schema section of this chapter, the URI’s for the standard taxonomies are usually web addresses that are out of reach to taxonomy creators.

2.2.13. XBRL INSTANCE DOCUMENT

After the taxonomy is created, it is then possible to create an instance document, in accordance to the specified taxonomy.

The instance document will contain the values for the elements specified in the taxonomy, together with an explanation of the context in which they are specified.

Finally, XBRL instance documents also allow the inclusion of footnotes, whenever further information is needed regarding a particular element.

2.3. LIMITATIONS OF XBRL AND THE NEED FOR ONTOLOGIES

In order to ensure the successful exchange of data, there are two interoperability areas which must be addressed:

- Common syntax – This relates to the way that the data is structured in order to be exchanged.
- Common semantics – Even if data can be exchanged, the semantics of the data being exchanged must be common, that is, the meaning behind the data concepts must be the same in order to enable complete interoperability between systems.

In the realm of accounting reporting data, XBRL has provided a common syntax for the exchange of financial accounting data, and thus becoming the ‘de facto’ standard for

exchanging accounting data in most countries around the world. However, it has some limitations, both technical and conceptual.

As mentioned in section 2.2.8, one of XBRL's technical limitations is in its calculation linkbase which does not allow calculations between elements which have different 'periodType' attributes, such as the elements from the Balance Sheet, which have an 'instant' 'periodType', with elements contained in the Income Statement or Cash-Flow Statement, which have a 'duration' 'periodType.'

On the other hand, a major conceptual limitation has to do with the variety of XBRL taxonomies, many of which are based on different accounting standards, which vary from country to country. Bansón et al, state that these various taxonomies, "may seem an advantage, but, actually, they represent an impediment for achieving the full, comprehensive expansion and application of the standard. If the bases on which the XBRL taxonomies rest are different, users will not be able to compare the financial information corresponding to companies from different countries." [21]

Therefore, as it currently stands, XBRL by itself is not capable of ensuring a common semantic for the exchange of financial reporting data on a global scope. However, the issue of semantic interoperability is one that is not so easy to solve, for three major reasons:

- Existence of different accounting standards - This the most common source of discrepancies, as national accounting standards differ from country to country. For example, these discrepancies may make it difficult for an analyst to compare companies which have financial data according to different accounting standards (ie US GAAP vs IFRS).
- Differing terms may describe the same concept – Even in those situations where the accountings standards being used are the same, it is common to find the same concept being described through different terms, such as, Turnover/Revenue, Net Income/Net Profit, etc.
- Different users may have different views on the same data – Accounting reporting needs may also vary upon the specific needs of the user. For example, internal accounting data (managerial accounting) needs are different from those of external data consumers, such as regulators, investors, analysts, etc. (financial accounting)

As XBRL focuses on financial accounting only, the issues behind the mapping of concepts between the managerial accounting and the financial accounting realms shall remain outside the scope of this work. However, the same principles which are to be applied to map accounting concepts belonging to different accounting standards are exactly the same as those which could be used to map managerial and financial accounting concepts.

An example of the discrepancies generated by different accounting standards can be shown below, where some of the financial reporting data of a company must be adjusted in order to be reported under a different accounting standard:

	Net Income	Shareholders' Equity
IFRS	$100.000	$1.000.000
US GAAP Adjustment		
- Reversal of depreciation	$10.000	$200.000
- Reversal of revaluation Surplus for fixed assets	$0	-$700.000
US GAAP	$110.000	$500.000

FIGURE 2.7 – IFRS TO USGAAP RECONCILIATION [22]

However, even when the same accounting standards are used, such as IFRS, there can be discrepancies on the way the financial reports are structured and on the terms utilized to describe the same concept, as exemplified below:

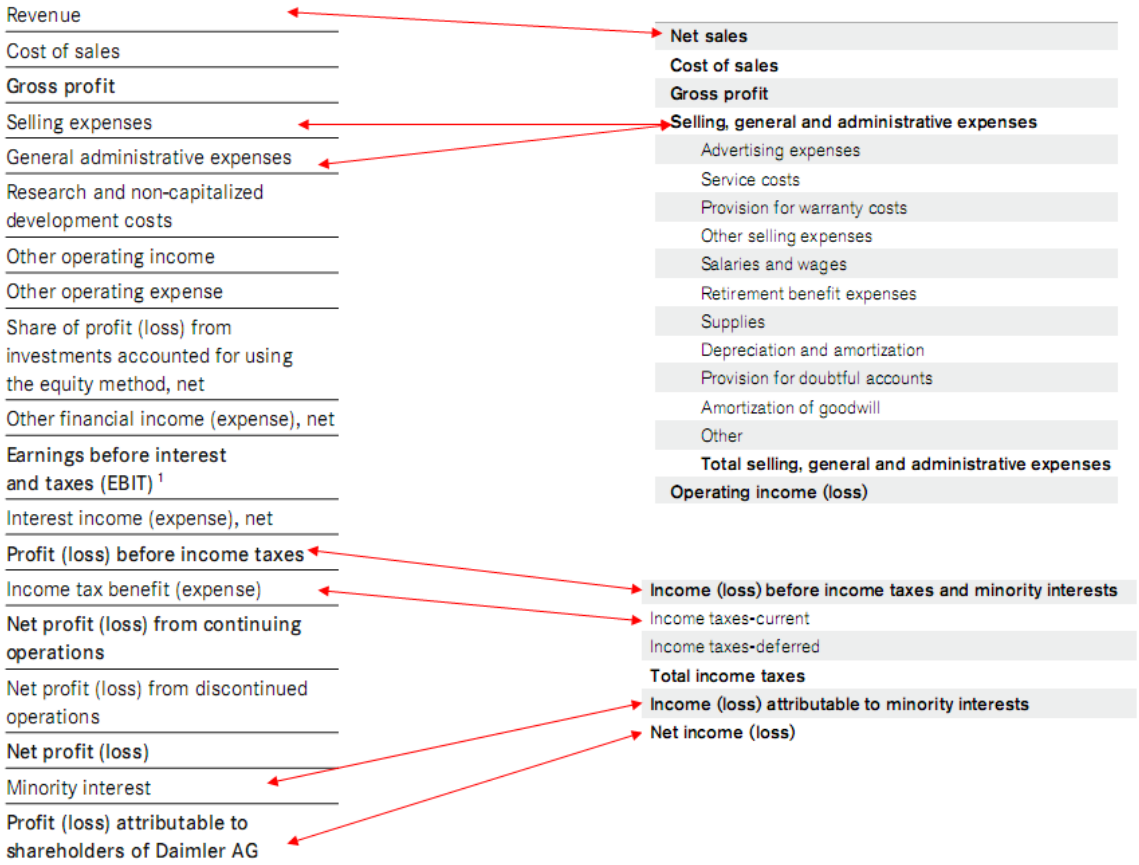

FIGURE 2.8 – INCOME STATEMENT ITEMS FROM DAIMLER [23] (LEFT) AND NISSAN [24] (RIGHT)

Figure 2.8 shows the items reported on the Income Statements of two automotive manufacturers, Daimler and Nissan. The red arrows relate items which are conceptually equivalent, but have different descriptions (Revenue vs Net Sales). Additionally, it is interesting

to notice how certain items may be grouped together, such as ‘Selling, general and administrative expenses’ on Nissan’s Income Statement, or they may be listed separately, as in Daimler’s case.

Therefore, there is a need for tools which will enable the mapping of concepts from one accounting standard to another. This would provide analysts and investors the ability to harmonize financial data according to a specified standard (ie IFRS), in order to better perform analysis and comparisons between different companies from different countries.

Finally, one last issue has to do with the availability of software tools for financial analysis for XBRL. The vast majority of tools for XBRL tend to be related to the generation, editing and validation of accounting reports. However, there are very few, if any, publicly available tools which can be used to perform comparisons or infer knowledge, or perform other sorts of financial analysis on XBRL reports. Therefore, there is an additional need to for the translation of financial accounting data into an information model, such as ontologies, which can take advantage of wider set of reasoning and analysis tools, query languages, etc.

3. Ontologies and Interoperability Semantics

Given some of the identified conceptual weaknesses around XBRL, in order to implement a proper methodology which will facilitate the mapping of accounting concepts, the adoption of a new form of data representation is proposed. The purpose of this section is to introduce ontologies and how they can assist in the realization of this goal.

3.1. Subject-Based Classification Methods

Before introducing ontologies, it is first necessary to understand the concept of subject-based classification methods.

Subject-Classification is “any form of content classification that groups objects by the subjects they are about.” [25] There are various methods to perform subject-classification, which in turn can be combined with other techniques. The most common ones are the following:

1. Controlled Vocabularies
2. Taxonomies
3. Thesauri
4. Ontologies

The following subsections shall provide a brief overview of these methods.

3.2. Controlled Vocabularies

Controlled vocabularies, as their name implies, are closed lists of named subjects which can be used for classification purposes. They are composed by ‘terms’, which can be defined as particular names for particular concepts [25]. Although concepts may have multiple names, in a controlled vocabulary each term is meant to refer to only a single concept, in order to avoid duplicate terms.

The purpose of controlling the vocabulary is therefore “to avoid authors defining meaningless terms, terms which are too broad, or terms which are too narrow, and to prevent different authors from misspelling and choosing slightly different forms of the same term.” [25]

3.3. Taxonomies

The word ‘taxonomy’ derives its meaning from two greek words, ‘taxis’ which means classification, and ‘nomos’, meaning law or science. Therefore, it is possible to define taxonomy as the “law or science” of classification.

According to Garshol [25], another definition of taxonomy can be the “subject-based classification that arranges the terms in the controlled vocabulary into a hierarchy without doing anything further, though in real life you will find the term "taxonomy" applied to more complex structures as well.”

Therefore, we can understand the taxonomy as the natural evolution of the controlled vocabulary, whereby the terms are now arranged into a tree-like structure, as exemplified in figure 3.1:

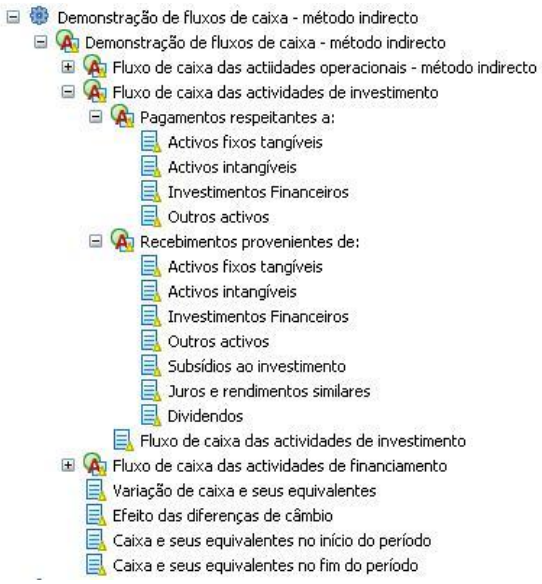

FIGURE 3.1 – EXAMPLE OF AN XBRL TAXONOMY

Taxonomies permit the arrangement of terms in a structured form, whereby it is possible to infer parent-child or type-subtype relationships.

XBRL utilizes taxonomies, as the items in financial accounting reports must be organized in a hierarchical form and certain items can also contain references to other sub-items.

However, apart from the parent-child relationships, taxonomies do not permit the construction of more advanced relationships between terms. Within the context of financial reporting, XBRL has tried to 'work around' some of these limitations through the use of linkbases, however even so, these do not permit more complex relationships between terms, and in particular, when the relationships are between terms that belong to different taxonomies.

3.4. THESAURI

Thesauri represent the next step in the evolution of subject-classification methods. They build upon the structure of the taxonomy, by allowing "other statements to be made about the subjects." [25]

While taxonomies, like controlled vocabularies, only allow single terms to define a concept, thesauri allow the inclusion of additional terms in the description of each concept, thus allowing a "much richer vocabulary for describing the terms than taxonomies" and thus "are much more powerful tools." [25]

3.5. ONTOLOGIES

The final step in the evolution of subject-classification methods is the ontology.

The earliest definition of ontology within the realm of IT was made by Gruber in 1993, who defined it as “an explicit specification of a conceptualization” [26]. However, other more descriptive definitions include:

1. A model for describing the world that consists of a set of types, properties, and relationship types [25]
2. A specification of the concepts of a domain and their relationships, structured to allow computer processing and reasoning [27], [28]

The ontology builds upon the thesaurus by permitting the definition of various types of relationships between concepts, thus providing for much richer descriptive capabilities and for increased information processing capabilities.

Some of the common components in ontologies include the following:

- Classes – These are the building blocks of ontologies and refer to the concepts, sets, collections, types of objects that are part of the domain to be represented by the ontology.
- Individuals – They are specific instances or objects of the classes described in the ontology
- Attributes – They can be the various properties, features or characteristics that are related to either classes or objects
- Relations – These define how classes and individuals can relate to each other
- Restrictions – They provide formal descriptions on what sort of assertions can be accepted as input in the ontology
- Rules – These are “if-then” statements which describe the logical inferences that can be drawn from an assertion
- Axioms – Finally, these refer to the logical expressions which can be used to describe complex relations between elements in an ontology.

The ability of being able to define all of the above characteristics allows one to not only define and describe concepts within a domain, with greater detail and specificity, but it also allows the usage of querying and reasoning tools to infer facts from a given ontology.

3.6. ONTOLOGY APPLICATIONS

Since an ontology can provide a common definition for the various concepts which refer to a specific domain, its usage can facilitate the exchange of knowledge, information and data.

Some examples of how ontologies can facilitate such exchange can include the following:

- Cooperation between people
- Integration of ontologies with software tools
- Integration with semantic web

At the level of cooperation between people, ontologies can facilitate cooperation at three levels:

1. Internal cooperation – By providing a common set of concepts to be used within an organization
2. External cooperation – This relates to cooperation between different organizations
3. Ontology integration – Ontologies can be used to integrate other ontologies, from the same domain, or even from different domains

With regards to their integration with other software tools, ontologies can be used in the following applications:

1. Design and development of software systems – Ontologies have a relevant role in the specification, relation and reutilization of software systems.
2. Communication – Ontologies facilitate the exchange and reconciliation of data
3. Interoperability – Finally, ontologies can also aid in software systems interoperability at various levels:
    a. Data Interoperability
    b. Function interoperability
    c. Process interoperability

Finally, ontologies are one of the building blocks of the Semantic Web, as they provide a common semantic for the exchange of data and information from different sources.

3.7. Application of ontologies in Financial Reporting

Bao, Rong and Ding [29] point out some of the weaknesses of XBRL with regards to concept matching, namely the fact that "while we can declare the equivalency of two concepts in XBRL using arc roles, there is no means in XBRL to infer new relations from the equivalency relation." This has to do with the fact that, according to the authors, XBRL "remains largely to be a structural model of financial reports, without addressing the logic model of these reports."

On the other hand, ontologies provide a better mechanism to perform the required mapping of accounting concepts, not to mention, the fact that they also provide better integration with other sources of information.

With the availability of tools which already perform the transformation of XBRL taxonomies and instances into ontologies, the present dissertation will work at the ontology level and it will utilize example ontologies as basis for focusing on the demonstrator.

4. MODEL MORPHISMS (MOMO)

In mathematics, “Morphism” is an abstraction of a structure-preserving map between two mathematical structures. It can be seen as a function in set theory, or the connection between domain and co-domain in category theory [30]. Recently, this concept has been gaining momentum applied to computer science, namely to systems interoperability. This new usage of “morphism” specifies the relations (e.g. mapping, merging, transformation, etc) between two or more information model specifications (M as the set of models). Therefore, a MoMo describes a model operation.

In this context, the research community identifies two core classes of MoMo: non-altering and model altering morphisms [30, 31]. As evidenced in Table 4.1, in the non-altering morphisms, given two models (source A and target B), a mapping is created relating each element of the source with a correspondent element in the target, leaving both models intact. In model altering morphisms, the source model is transformed using a function that applies a mapping to the source model and outputs the target model [32]. Other relations, such as the merge operation, can also be classified as model altering morphisms, however they are not detailed in this dissertation. The focus shall be on the mapping operations.

TABLE 4.1 – CASES OF MODEL MORPHISMS

MoMo	Formalization	Classification
Mapping: $\theta(A,B)$	$\forall A,B \in M: \theta(A,B) \subseteq Sub(A) \times Sub(B)$	Non-altering
Transformation: $\tau: A \times \theta \rightarrow B$	$\forall A,B \in M: if\ \exists \theta(A,B)\ then\ \tau(A,\theta) = B$	Model altering

4.1. SEMANTIC MISMATCHES

Mismatches are inconsistencies of information that result from “imperfect” mappings. Due to the differences among models referred before, almost in every case, a MoMo leads to a semantic mismatch, which can either be lossy or lossless depending on the nature of the related model elements (see Table 4.2): In lossless cases, the relating element can fully capture the semantics of the related; while in lossy mismatches a semantic preserving mapping to the reference model cannot be built [33].

TABLE 4.2. SEMANTIC MISMATCHES (BASED ON [31])

	Mismatch	Description
Lossless	Naming	Different labels for same concept
	Granularity	Same information decomposed (sub)attributes
	Structuring	Different design structures for same information
	SubClass-Attribute	An attribute, with a predefined value set (e.g. enumeration) represented by a subclass hierarchy
	Schema-Instance	An attribute value in one model can be a part of the other's model schema
	Encoding	Different formats of data or units of measure (e.g. kg and lbs)
Lossy	Content	Different content denoted by the same concept
	Coverage	Absence of information
	Precision	Accuracy of information
	Abstraction	Level of specialisation (e.g. "Car" and "Ford")

This notion of mismatch can bring a semantic meaning to the type of the relationship being established in the mapping. However, the envisaged semantic "link" between two different models needs to account for more than inference of a meaning. It needs to be represented through a formal expression that is traceable and parseable by an intelligent system that can deduce and recommend mapping readjustments, which might even change the mismatch type. Figure 4.1 provides some mismatch examples:

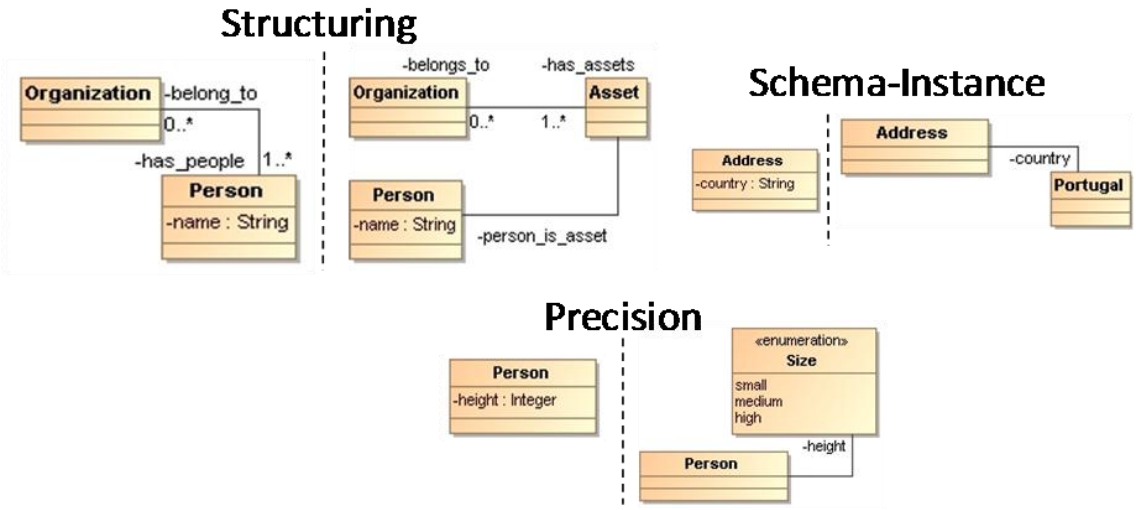

FIGURE 4.1 MISMATCH EXAMPLES

4.2. MoMo Formalisms

Model Morphisms are intended to introduce a method of describing relationships/transformations among models. Originally graph theory has been used, but other and theories can be considered to achieve the envisaged goals:

4.2.1. CLASSICAL MATHEMATICS: GRAPH & SET THEORY

Graphs are a common way to graphically present models, where the nodes are considered as a domain entity and the edges as relations between them. For the purposes of MoMo, model operations such as the ones of Table 4.1 can be described using a 6-tuple labelled oriented multigraph (*LDMGraph*) of the form *G=(V,E,s,t,lv,le)*, where: *V* is the vertex set of *G*; *E* is the edge set of *G*; *s: E → V*, is a function that associates an edge with its source vertex; *t: E → V*, is a function that associates an edge with its target vertex; *lv: V → ∑ V*, is a function that associates a vertex with its label; *le: E → ∑ E*, is a function that associates an edge with its label [30], [34]. This abstract view of models allows formal reasoning on their properties and on the properties of the model operations needed for their effective management.

As graphs, also sets can be used to represent models and operations using first-order logic, algebra and axioms. Being defined as a collection "*M*" of distinct objects "*m*", a set can represent objects, numbers, other sets, etc. [35] Operations such as membership "$M1 \subseteq M2$", power "$P(M)$", union"$M1 \cup M2$", intersection "$M1 \cap M2$", complement "$M1 \setminus M2$", or cartesian product "$M1 \times M2$" are already well defined.

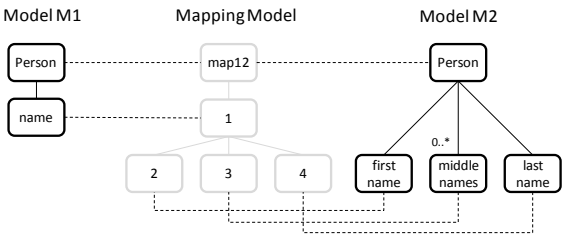

FIGURE 4.2 - MAPPING AS A MODEL (MAP12)

4.2.2. MAPPING AS A MODEL: MODEL MANAGEMENT [36]

This theory defends that a mapping between models *M1* and *M2* should be a model "*map12*" and two morphisms (one between "*map12*" and *M1* and another between "*map12*" and *M2*). Thus, each object "*m*" in the mapping can relate a set of objects in *M1* to a set of objects in *M2*. In this approach, instead of representing a mapping as a pair of objects, a mapping is represented as a set of objects (see Figure 4.2). Using concepts from classical mathematics, this approach enables to define complex algebra to describe major model operations such as match, compose, diff, model gen, or merge.

4.2.3. MAPPING AS A COMPLEX TUPLE: MATCHING [37]

The match operator takes two graph-like structures and produces a mapping between the nodes of the graphs that correspond semantically to each other. Mappings between these elements can be described using set-theoretic semantic relations instead of using traditional numeric coefficients. The meaning of concepts (not labels) within a model can determine equivalence "≕", more "⊒" and less "⊑" general, as well as disjointness "⊥" relationships. Having

this, a mapping element can be defined as a 4 level tuple <*IDij, ai, bj, R*> where: *IDij* is a unique identifier of the given mapping element; *ai* is the i-th node (or vertex) of the first tree; *bj* is the j-th node of the second tree; and *R* specifies the semantic relation which may hold between them.

The above methodologies seem to be powerful in terms of expressiveness of the morphism. However others exist, such as the composition of complex operations based on a catalogue of primitive transformations [38]. However, this approach is more focused on model altering morphisms.

4.3. KNOWLEDGE ENRICHED TUPLES FOR MAPPING REPRESENTATIONS

In the paper "Tuple-based semantic and structural mapping for a sustainable interoperability", Agostinho et Al., propose a "Knowledge Enriched Tuple" [39] to perform mapping representations.

The authors propose a 5-tuple mapping expression (equation 1), reusing some of the concepts explained in section 3, that formalizes the morphism between two model elements (*a* and *b*) and is enriched with semantic information that enables fast human readability, where $\forall\, A, B \in M, \exists a \in A\ and\ \exists b \in B\colon if\ M\ is\ an\ LDMGraph\ then\ a \in V(A) and\ b \in V(B)$.

$$Mapping\ Tuple\ (MapT) : < ID, MElems, KMType, MatchClass, Exp > \qquad (1)$$

- *ID* is the unique identifier of the MapT and can be directly associated with the *a*'s vertex number: $IDi.j\_x\colon 1 \leq i \leq |V(A)|\ and\ 1 \leq j \leq \left|V\left(Sub(B)\right)\right| and\ x \in \mathbb{N}$. The depth of the sub-graph detail used in the mapping is not limited, and *x* is a counter for multiple tuples associated with the same concept;
- *MElems* is the pair (a, b) that indicates the mapped elements. If the *ID* specifies a mapping at the n-th depth level of the graph, a should be at the same level, i.e. *a.ai* (for *i =1..n*);
- *KMType* stands for Knowledge Mapping Type, and can be classified as: "Conceptual" if mapping concepts and terms; "Semantics" if mapping model schemas; and "InstantiableData" if the mapping is specifying instantiation rules.
- $KMType = \{Conceptual, Semantics, InstantiableData\}$;
- *MatchClass* stands for Match/Mismatch Classification and depends on *KMType*, such as $\forall (a, b) \in MElems$:
- $\forall KMType$, if a=b, the mapping is absolute and $MatchClass = Equal$;
    - if $KMType = Conceptual$, the mapping is relating terms/concepts, and $MatchClass \in \begin{Bmatrix} Equal, Naming, Coverage, \\ MoreGeneral, LessGeneral, Disjoint \end{Bmatrix}$ depending on the coverage of the relationship;
    - Otherwise, the mapping is structural or non-existent and $MatchClass \in Table\ 4.2 \cup \{Equal, Disjoint\}$;
- *Exp* stands for the mapping expression that translates and further specifies the previous tuple components. It can be written using a finite set of binary operators derived from the mathematical symbols associated with the mapping types and classes (e.g. "$=, \sim, \subseteq, \supseteq, \perp, +, -, \times, \div, concatenate, split$").

This mapping tuple which represents $\theta(a, b)$, can also be used to generate a transformation function τ, where $\tau(a, \theta) = b$, being $(a, b) \in MElems$. Therefore, when used by intelligent systems such as those similar to complex adaptive information systems, the tuple's information

enables automatic data transformations and exchange between two organizations working with/on different information models, thus achieving an interoperable state among them and supporting the recovery from any harmonization breaking situation.

4.4. COMMUNICATION MEDIATOR

The proposed Communication Mediator is defined by an ontology (knowledge base) in OWL format which was built as an extension to the Model Traceability Ontology defined in [40], addressing traceability as the ability to chronologically interrelate the uniquely identifiable objects in a way that can be processed by a human or a system. The structure of the evolved communication mediator is presented in Figure 4.3, shown below:

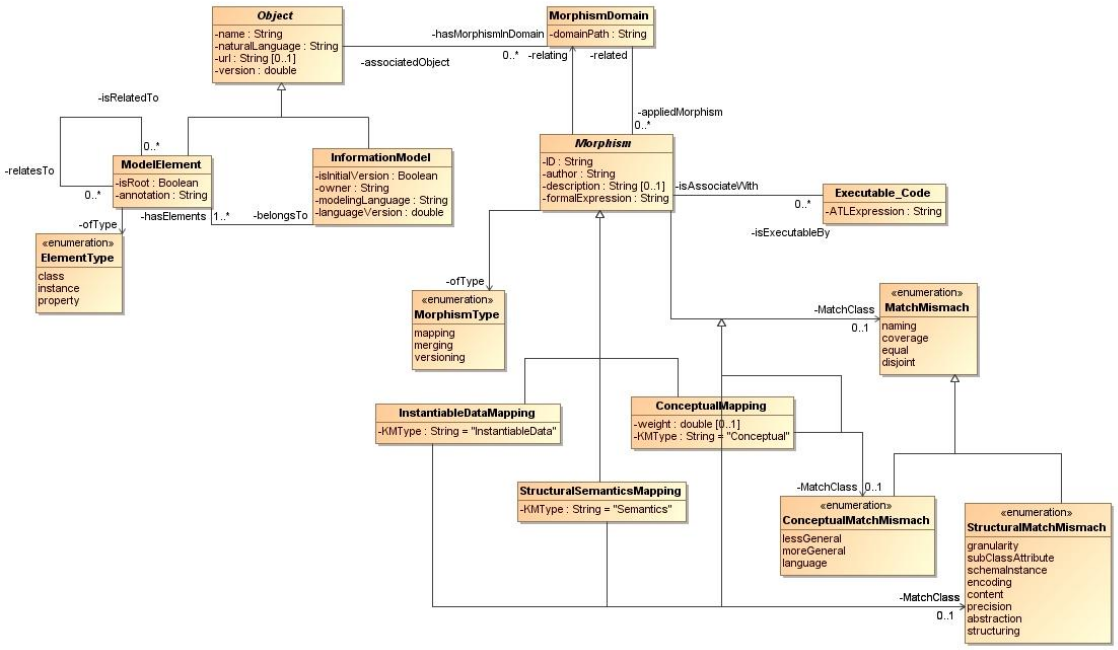

FIGURE 4.3 - STRUCTURE OF KNOWLEDGE BASE FOR COMMUNICATION SUPPORT (CM)

The structure of the ontology base rests on two main classes:

- Object – The Object class is the parent class of two other major classes: Information Model, which represents the model/ontology itself, and the ModelElements class, which represents the building blocks of the Information Model, such as classes, properties and instances.
- Morphism – The Morphism class represents the MapT described in the previous section. It describes the morphism relationship by associating two Objects (related and relating – Melems in MapT), and classifying it according to the MorphismType and KnowledgeMappingType properties (when the morphism describes a mapping), and respective Match/Mismatch class (MatchClass in MapT).

Finally, the ontology is also prepared to store the associated model transformation code, which shall be written in Atlas Transformation Language. The code shall be stored in the ExecutableCode class.

5. Model-Driven Architecture

Model-Driven Architecture (MDA) is a software development paradigm which was introduced in 2001 by the Object Management Group (OMG), a consortium involved in setting standards in the software industry.

According to OMG [41], MDA "separates business and application logic from underlying platform technology."

The MDA architecture relies on the following types of models:

- Platform-Independent Models (PIM) – These are usually expressed in languages, such as UML or MOF.
- Platform-Specific Models (PSM) – These refer to the specific implementations of a PIM, according to a specified language, such as Java.

The goal behind MDA is that "automated mappings can be used to move from a PIM to a PSM and to round-trip between a PSM and a code" [4]. In MDA, the models are the starting point to code generation. Once a system is modeled, it can then be implemented and mapped into various specific software implementations.

In order to accomplish this goal, MDA relies on several technologies, such as MOF, XMI, UML and QVT, some of which shall be explained in the following sections.

5.1. Meta Object Facility

The OMG specification states that "Software development in the MDA starts with a Platform-Independent Model (PIM) of an application's business functionality and behavior, constructed using a modeling language based on OMG's Meta Object Facility (MOF)." [43]

In order to better understand the above requirement, the MOF metamodeling hierarchy is shown below:

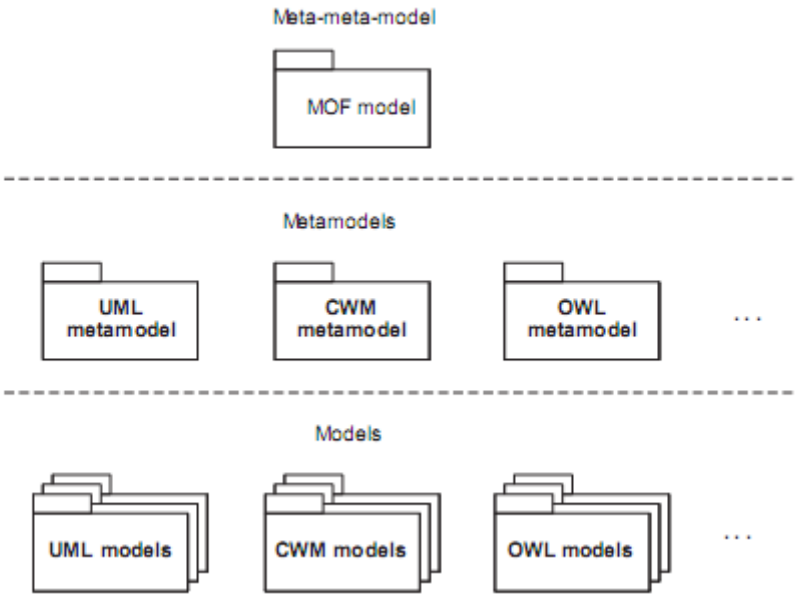

FIGURE 5.1 – MOF METAMODELLING HIERARCHY [4]

As it is shown in Figure 5.1, the PIM's represent the lower level layer, represented by the various models which can include UML, CWM or OWL models. In order to define these models, a modeling language must be used, such as UML or OWL. The definition of these languages is described through their metamodels, represented by the middle layer of the hierarchy. Finally, as the OMG specification states, the metamodels must be defined according to a specific standard, the Meta Object Facility (MOF) Model. Hence, the MOF model can be considered as the metamodel of the various modeling languages.

As an OMG standard, MOF is an abstract language for the specification of metamodels, as it "defines the essential elements, syntax, and structure of metamodels that are used to construct object-oriented models of discrete systems." [44]

5.2. UNIFIED MODELING LANGUAGE

Unified Modeling Language (UML) is another OMG standard, which is used in modeling discrete systems. It is OMG's most-used specification [45] and has become a "de facto" industry standard.

UML derives its notation and unifies the notations of three object-oriented design and analysis methodologies [46]:

- Grady Booch's methodology for describing a set of objects and their relationships
- James Rumbaugh's Object-Modeling Technique (OMT)
- Ivar Jacobson's approach which includes a use case methodology

In the context of MDA, UML is OMG's the language of choice for the specification of PIM's, however, other modeling languages may also be used.

5.3. XML METADATA INTERCHANGE

XML Metadata Interchange, more commonly known as XMI, is an XML-based standard used in the exchange of metadata information for metamodels expressed in MOF.

One of XMI's main applications is to allow the easy exchange of data between UML-based modeling tools and MOF-based metadata repositories. Additionally, XMI is also used as the method through which models are transferred from modeling tools to software generation tools, within the realm of MDA.

5.4. QUERY/VIEW/TRANSFORMATION

Query/View/Transformation, usually referred to as QVT, is an OMG standard for model transformations. It is based on another OMG standard, Object Constraint Language (OCL), which is a declarative language used in describing rules for UML models.

One of the issues with QVT, is its complexity, as QVT actually defines three domain-specific languages, as follows:

- Relations (Descriptive language)
- Core (Descriptive language)
- Operational Mappings (Imperative language)

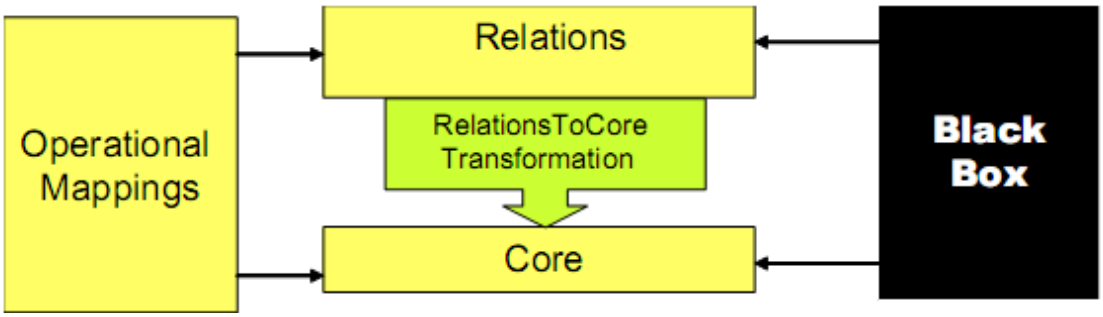

FIGURE 5.2 – RELATIONSHIPS BETWEEN QVT METAMODELS [47]

The Relations language is utilized to provide a declarative specification of the relationships between MOF models. It supports "complex object pattern matching, and implicitly creates trace classes and their instances to record what occurred during a transformation execution" [47]. Additionally, it can "assert that other relations also hold between particular model elements matched by their patterns." [47]

According to OMG's QVT Specifications [47], the Core language is a "small/model language that only supports pattern matching over a flat set of variables by evaluating conditions over those variables against a set of models. Compared to the Relations languge, it is simpler but equally powerful. However, given its simplicity, its transformation descriptions are "more verbose". [47]

Finally, the Operational Mappings language serves as a mechanism to invoke imperative implementations of either Relations or Core. The QVT Specification states that “Mappings operations can be used to implement one or more Relations from a Relations specification when it is difficult to provide a purely declarative specification of how a Relation is to be populated.” [47]

The major problem that QVT has is that, so far, it is only a standard which has little practical implementation, because, there is not yet a full implementation of the QVT standard. However, there are few usable implementations, namely:

- medini QVT – QVT-Relations
- Eclipse M2M – QVT-Operational Mappings
- Smart QVT - QVT-Operational Mappings

In addition to the above, there are other criticisms to QVT:

- QVT only addresses model to model transformations, not model-to-text or vice versa. Therefore, it is limited to performing XMI to XMI transformations
- The specification is so extensive and complex, that it is doubtful that it will ever be fully implemented

Because of these issues, it was therefore necessary to study other available transformation languages, as described in the following section.

5.5. Analysis of Model Transformation Languages

As mentioned previously, given some of the issues surrounding QVT, it was necessary to perform an evaluation on whether it was the most appropriate language to be used within the specific context of the present dissertation.

In order to perform this evaluation, some relevant criteria were identified, as well as, two other transformation languages, in order to perform a comparison with QVT.

The other transformation languages which were evaluated are described below.

5.5.1. Atlas Transformation Language

Atlas Transformation Language (ATL) is a model transformation language developed at INRIA, as an answer to OMG’s QVT Request for Proposal. Hence, it is “inspired” by QVT, however, it does not comply with its specification.

Unlike QVT, which as described previously defines three specific languages, ATL is a single language which incorporates both declarative and imperative characteristics. The preferred method for the development of transformations is declarative, however, imperative constructs

are also allowed, in order to handle some mappings which are too complex to be developed in a declarative form.

While QVT allows for bidirectional transformations, ATL is unidirectional. Therefore, in the cases where bidirectional mapping is necessary, ATL requires separate descriptions for each model transformation direction.

One of ATL's major strengths is the fact that it is widely used, and has very good documentation and support from wide user community, as part of the Eclipse Modeling Project.

5.5.2. XTEND (OPENARCHITECTUREWARE)

OpenArchitectureWare (oAW) is an "open source tool set for defining and processing models". [48]. oAW provides a comprehensive set of tools, which can assist developers in all the steps of the model transformation chain, such as, defining metamodels, building editors for the corresponding models, checking the validity of the models created, and, finally, generating code (the XML files).

oAW focused on having a modular approach, thus allowing some of its tools to be used in other contexts and also, "making it possible to integrate tools that originate outside of oAW." [48]. Although its main focus is the generation of code through the use of MDA, it did include Xtend, a model transformation language, which could be used to perform model transformations.

Xtend does not follow any standard or specification, however, in September 2009 the main components of openArchitectureWare's framework have been integrated into the Eclipse Modelling Project, where generator components were merged into the model-to-text (M2T) project. [48].

Two of the major reasons which led to the evaluation of Xtend, as possible choice to perform model transformation were:

- Simplicity – Although also inspired by OCL, the syntax of Xtend is much simpler than that of QVT or even ATL
- Java integration – As it was built on top of Eclipse and with a modular architecture, Xtend provides much easier integration with Java.

Further details on how ATL and Xtend fared when compared with QVT, are provided in the next subsection.

5.5.3. QVT IMPLEMENATIONS EVALUATED

Given that the QVT is split into three different sub-languages, two different QVT implementations were evaluated, namely:

- medini QVT – QVT-Relations

- Smart QVT - QVT-Operational Mappings

medini QVT is an implementation of OMG's QVT Relations specification developed by german developers ikv++ technologies. It is available as open source software, and it was developed as an Eclipse plug-in. [49]

Smart QVT implements of OMG's QVT Operational Mappings and it was developed by France Telecom. Just like medini QVT, it is also available as open source software in the form of an Eclipse plug-in. [50]

Despite being separate implementations of two QVT sub-languages, they both have the same issues, as both implementations suffer from having very little documentation and user support available. Given the complexity of QVT, neither implementation seemed like a good choice upon which an implementation could be build upon.

5.6. Evaluation criteria and analysis

The criteria which were deemed relevant, and their respective weight in this evaluation, were the following:

1. Java Integration (20%)
2. Documentation, tools and available support (30%)
3. Language capabilities (20%) (declarative vs hybrid)
4. Simplicity (20%)
5. Standardization (10%)

5.6.1. Java Integration

The first criterion that was taken into consideration was the Java Integration abilities of the transformation language in question. Since the language is to be included into a tool to be developed in Java, it was of extreme importance that the language should provide an API for easy integration with Java code.

All the 3 languages do provide Java integration, however, XTend is the one that integrates itself more fully with Java. Both QVT and ATL also provide API's so that one can call either language from Java code.

5.6.2. Documentation, tools and available support

Since model transformation is a relatively new field and that many of the tools are still under constant development, this criterion was considered to be the most important one.

Indeed, during the early stages of the research work behind this dissertation, much time was spent on trying tools and tutorials that simply did not work, or for which there was not much information available.

In this criterion, ATL is definitely the better language. The fact that it is widely used has led to the existence of a wide variety of model transformation examples, and the language also has a lively online forum where one can get assistance from other ATL users. Additionally, the language is easily integrated into Eclipse.

As for Xtend, it also has a good online user forum which is constantly updated and a user manual. However, some of the examples provided in the user manual have not been yet updated to the newer version of the tool and, beyond the user manual and the forum, there is not much more documentation available.

As for QVT, the fact that there are different versions of the language, as well as, different implementations of each language, makes the existence of a wide user base somewhat more difficult. The 2 versions of QVT that were tested provided little documentation and the user support base did not seem to be as active as that of other languages.

Additionally, although QVT is a standardized language by OMG, apart from the standard itself, there are fewer examples and tutorials available when compared to ATL.

5.6.3. Language Capabilities

The Language Capabilities criterion refers to the particular abilities that each transformation language offers.

In this category, QVT is undoubtedly the most powerful language. Its structure provides both declarative and imperative languages, and, unlike the other two languages, it provides the ability to perform bidirectional transformations. However, the fact that the declarative and imperative language capabilities are provided through different languages, also adds to the complexity.

Unlike QVT, ATL only allows the user to perform unidirectional transformations. However, this disadvantage relative to QVT is balanced with the fact that ATL is hybrid language, meaning that it allows for both declarative and imperative operations. The hybrid nature of ATL is a major advantage in that it allows one to use mostly declarative programming and using imperative programming whenever necessary, thus making it a very flexible programming language.

Finally, XTend allows for the same level of model transformation as ATL, therefore, it receives the same mark.

5.6.4. Simplicity/Complexity

This criterion is related to the level of complexity of the language, regarding its syntax and the amount of code to perform a certain task.

Undoubtedly, QVT is the most complex language. As mentioned earlier, being also the most powerful of the three, the additional capabilities that it offers, also translates itself into additional complexity.

On the other hand, Xtend is definitively the simpler language and the one that is easier to use.

As for ATL its complexity level lies somewhere in between that of Xtend and QVT. Even though it syntax is somewhat similar to QVT, its hybrid nature simplifies some of the syntax necessary to perform a given transformation.

5.6.5. STANDARDIZATION

Finally, the standardization criterion tries to assess whether a transformation language follows any existing standards or specifications.

Out of the three languages under comparison, only QVT is a standard language, as defined by the Object Management Group. However, although the QVT specification exists on paper, not all QVT implementations are fully QVT compliant. Still, both QVT implementations that were tested for this comparison were fully compliant with standard.

As for ATL, it is considered to be a “QVT-like” language, as the language was written in response to a Request for Proposal from OMG for QVT. However, although it has many similarities with QVT, ATL does not comply with the OMG standard. But, given the large user base and the large amounts of example transformations available, there is a possibility that ATL could become the ‘de facto’ standard for transformation languages.

Finally, we have Xtend which does not follow any standard or specification. Therefore, it receives the lowest mark out of the three languages.

5.6.6. LANGUAGE OF CHOICE

In order to make a fair evaluation of all three languages, each language was graded on each criterion on a scale between 1 (Lowest) and 5 (Highest). The results were as follows:

TABLE 5.1 - COMPARISON OF TRANSFORMATION LANGUAGES

	Java Integration (20%)	Documentation & Available Tools (30%)	Language Capabilities (20%)	Simplicity (20%)	Standardization (10%)	TOTAL
ATL	4	5	4	4	3	4,2
QVT	4	2	5	3	5	3,5
Xtend	5	3	3	5	2	3,7

When one takes into consideration all the factors and criteria explained above, ATL seems to be the most appropriate choice.

The hybrid nature of the language, which gives it much flexibility, together with an abundance of example transformations, and a lively user base, make it an appealing choice.

The Xtend language also seemed like an appropriate choice, however, being the newer language of the three, does not allow it to yet have the amount of materials and support which already exists for ATL.

Finally, although QVT looks good on paper, practical implementations of the language are still somewhat lacking. The documentation and available examples were poor, and for a language that is the most complex of all three, it is certainly a major drawback.

6. Proposed Solution, Implementation and Testing

6.1. Proposed Solution

The scenario which shall be the focus of the present dissertation is the case where a financial analyst has to evaluate the financial reports of companies which utilize different accounting standards.

This sort of situation is very common not only amongst financial analysts working for investment firms, but also among international investors, ratings agencies, regulatory authorities and in other situations where there is a need to harmonize accounting data which is described according to differing accounting standards.

6.1.1. AS-IS Scenario

Currently, the most common method for achieving harmonization of accounting data is through manual adjustment. Usually, a financial report in one accounting standard is loaded up in a spreadsheet, and then manually adjusted in order to comply with the target accounting standard.

Obviously, while this sort of method allows for great control of the data, it has some obvious disadvantages:

1. Greater probability of errors – Any time there is human intervention in the exchange of data, there is a greater propensity for data errors caused by human interaction
2. Longer processing times – The need for human intervention also adds to the processing time required to perform the evaluation work.

Although the mapping of accounting items in accordance to different standards is not a trivial matter, there are however certain adjustments which could be mapped only once and then automated, thus saving time and the potential for errors which can occur due to manual intervention.

6.1.2. TO-BE Scenario

The envisaged To-Be scenario aims to circumvent some of the major disadvantages of the current scenario described earlier. The goal is to permit some sort of automation of the mapping tasks, thus reducing the potential for errors and reducing processing times.

Therefore, the proposed solution is the development of a tool which can assist in the mapping of concepts between different accounting standards, and make those mappings accessible to third-parties.

The tool should permit the storage and retrieval of accounting concept mappings, which could provide the following advantages:

- Mappings only would need to be done once – The manual adjustments which are already done when there is a manual adjustment of accounting data would still be required. However, this task would only have to be done once, as the mapping could then be stored and retrieved, whenever necessary.
- Integration with other tools to permit automation – Once the mappings of accounting concepts are stored and made available, it would then be possible to build other software tools which could assist in the automation of the transformation of accounting data between one accounting standard to another.

The mapping of accounting concepts will be achieved through the use of a mediator ontology, which will provide the data structure that will permit the correct and efficient storage of the various mappings which may occur when exchanging data between two different accounting standards.

In addition to the storage of the concept mappings, the mediator ontology will also allow the storage of ATL code. The purpose of this code is to permit the transformation of accounting data from one accounting standard to another.

Although the mapping data can potentially be used by other tools, the inclusion of model transformation code is of great benefit as it also serves to illustrate how the model-transformation paradigm can facilitate the automation of interoperable data.

6.1.3. DEMONSTRATOR TOOL

The Demonstrator tool implemented to evaluate the concepts described earlier, builds upon the following software technologies:

1. Data models based on ontologies
2. Usage of a mediator ontology for storage of mappings between concepts of two different accounting data ontologies.
3. Model transformation code generation in ATL
4. SPARQL query language for retrieval of data stored in mediator ontology
5. Integration with PROTÉGÉ

The above technologies shall be described in further detail.

6.1.3.1. USAGE OF ONTOLOGIES AS DATA MODELS

The proposed solution aims to side-step XBRL and promote the usage of ontologies for the representation of financial data.

In addition to some of the issues related to the usage of XBRL outlined in section 2.14, there are advantages for the usage of ontologies for the representation of financial data. Reyes et al [51], state the following advantages:

- They facilitate the transparent integration of different accounting standards
- Usage of ontologies as means to formalize concepts in the financial domain, in order to create better taxonomies
- Usage of ontology tools, such as Protégé, in conjunction with other XBRL ontologies

Regarding the aforementioned advantages, the usage of ontologies is essential, as the main purpose of the demonstrator tool is the mapping of accounting concepts between two different accounting standards.

Additionally, it is also interesting to point out the apparent lack of tools which can be used to analyze XBRL data. Most of the XBRL tools available tend to be mostly focused on the needs of businesses which need to generate XBRL reports for financial reporting purposes. There are few, if any, suitable tools for the querying and analysis of financial data in XBRL format. Hence, the usage of ontologies for financial accounting data representation could then make use of the many open-source and commercial ontology tools which could then be used to perform data analysis.

6.1.3.2. Usage of a Mediator Ontology

The Demonstrator tool which shall be developed within the scope of the present dissertation is based on an ontology mediator tool, the “Communication Mediator”, proposed by Agostinho et al [39], which shall provide a platform for the mapping of two ontologies. The Communication Mediator was already described in sub-section 4.4.

6.1.3.3. Generation of Model Transformation Code in ATL

The proposed demonstrator tool also aims to show the capabilities offered by MDA technologies, namely, the facilities provided through the model transformation paradigm.

Therefore, the tool will also permit the generation of ATL code which implements the mapping of concepts being stored in the Communication Mediator. However, the code generated is only for a single concept mapping. Therefore, in order to perform the model transformation of a complete data model (ie Income Statement), it is necessary some sort of mechanism which will collect the various mappings and their associated code, before the final model transformation can be executed.

6.1.3.4. Usage of SPARQL for Retrieval of data Stored in Mediator Ontology

The proposed tool must also support some sort of method which allows for the retrieval of data stored in the mediator ontology. This is essential as the proposed solution can serve as a starting point for further enhancements or even other tools which can build upon the demonstrator.

Therefore, the proposed demonstrator shall also implement some basic queries which can permit the retrieval of data stored in the mediator ontology (ie Model Elements, Morphisms, ATL Expressions, etc.) An example of how this feature can be applied is, for example, the development of a method which can retrieve all the mappings belonging to a particular 'MorphismDomain' and their respective 'ATLExpression' attributes, which contain the ATL code necessary to implement a mapping. Upon retrieval of all the 'ATLExpression' strings, these could be combined to automatically generate the final ATL code which can run the complete model transformation between two models

In order to implement this data retrieval capability, SPARQL (SPARQL Protocol and RDF Query Language) was chosen, as it is the most commonly used query language for ontologies, as well as, W3C recommendation. [52]

6.1.3.5. INTEGRATION WITH PROTÉGÉ

Finally, the proposed demonstrator will be developed as a Protégé plug-in (tab). The reason for doing so is that Protégé is one of the most widely used ontology editors, with an Application Programming Interface (API) which facilitates the development of third-party applications/plug-ins for ontology engineering. Additionally, there is already a wide variety of plug-ins available for Protégé, hence, the integration of the proposed demonstration tool with Protégé will allow for easier integration with existing and future plug-ins, does extending its future capabilities and increasing its possible adoption to a wider audience.

6.1.4. APPLICATION SCENARIO

The envisaged scenario aims to illustrate a common situation amongst financial and accounting professionals, namely, the mapping of financial reports from one accounting standard to another.

The application scenario shall focus on the translation of a company's Income Statement, which was prepared in accordance to US accounting standards (US GAAP), into its equivalent in accordance to International Accounting Standards (IFRS). This sort of need could arise in the following situations:

- An American company that is required to publish its financial accounting reports in accordance to international accounting standards, as it has operations in several countries or simply because its international investors are more familiar with IFRS accounting rules.
- A financial analyst or investor who is comparing companies from a given sector (ie oil companies) and, therefore, needs to harmonize the financial reports from companies of various countries and needs a common accounting standard. The usage of IFRS would be a sensible choice, and therefore, the analyst must converge the financial reports of various companies into the common accounting standard (IFRS).

- Other scenarios where conversion financial reports from one accounting standard to another may occur can be related to regulatory requirements (ie companies with stock listed in a foreign exchange) , evaluation by third-parties (ie international banks evaluating a firm, before deciding upon a loan), etc.

Therefore, for demonstration purposes, a simplified Income Statement, as shown in Figure 6.1, shall serve as the basis for the Use Cases which shall be implemented by the demonstrator tool.

US GAAP			IFRS
Sales	1000000	1000000	**Revenue**
Cost of Sales	600000	600000	Cost of Sales
Gross Profit	400000	400000	**Gross Profit**
COSTS AND EXPENSES			**COSTS AND EXPENSES**
Selling expenses	55000	187000	Selling, general and administrative expenses
General expenses	120000	3000	Depreciation
Administrative expenses	12000		
Depreciation	3000		
Operating income	210000	210000	**Operating income**
Non-operating income	23000	23000	Other income
Earnings before Interest and Taxes	233000	233000	**Income (loss) before interest and income taxes**
Interest expense	12000	12000	Interest Income (Expense)
Income before provision for income taxes	221000	221000	**Profit (loss) before income taxes**
Provision for income taxes	55250	55250	Income tax benefit (expense)
		-3000	Reversal of Depreciation
Net income	165750	162750	**Net Profit (Loss)**

FIGURE 6.1 – INCOME STATEMENT ACCORDING TO TWO DIFFERENT ACCOUNTING STANDARDS (US GAAP & IFRS)

This simplified Income Statement aims to illustrate some of the various possible mappings which can occur in terms of concept combinations, namely:

- 1-to-1
- N-to-1

Based on the simplified Income Statement from Figure 6.1, we can extract some examples for the above mapping combinations.

FIGURE 6.2 – EXAMPLE OF A 1-TO-1 MAPPING

An example of a 1-to-1 mapping is shown in Figure 6.2 above, where the concept 'Sales' is mapped directly to 'Revenue'. This sort of mapping is applied when two different concepts have the same meaning but different terminology.

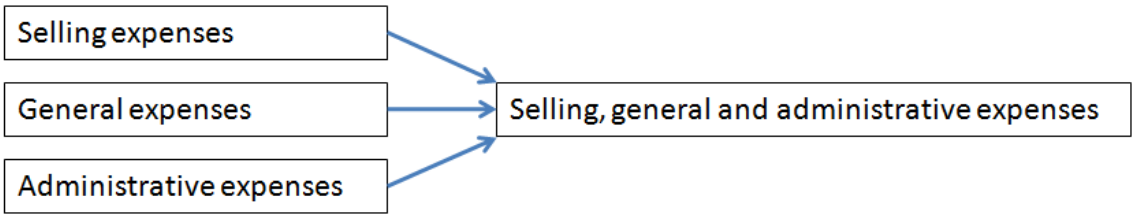

FIGURE 6.3 – EXAMPLE OF A N-TO-1 MAPPING

Figure 6.3 gives an example of an N-to-1 mapping, where several concepts in the USGAAP Income Statement must be combined into a single one in the IFRS Income Statement.

The Demonstrator tool that shall be developed will implement a series of services which will permit the insertion, as well as, the querying of data in the above described ontology. Additionally, the tool will also provide the associated ATL code which, once integrated into a complete ATL transformation code, can perform the morphism between the two ontologies.

6.2. Proof-of-Concept Implementation

The proposed demonstrator solution presented in section 6.3 needed to be validated to ensure its suitability as a possible solution to the issues addressed in earlier sections.

Given the variety of software technologies addressed, as well as, the challenges which are present when working with technologies which are still under development, such as the many MDA software tools, the present dissertation implemented certain parts of the complete solution, which focused on the following areas:

1. Development of methods for populating the Mediator Ontology
2. Development of methods for querying the Mediator Ontology
3. Development of a Graphical User Interface, as a Protégé plug-in
4. Generation of ATL expressions for concept mapping

The following sections will describe in further detail the implementation work which was accomplished.

6.2.1. Implementation Overview and Technology Used

Section 6.1 stated already the characteristics of the proposed demonstrator tool, namely:

1. Data models based on ontologies

2. Usage of a mediator ontology for storage of mappings between concepts of two different accounting data ontologies.
3. Model transformation code generation in ATL
4. SPARQL query language for retrieval of data stored in mediator ontology
5. Integration with PROTÉGÉ

In order to comply with the first requirement, OWL was the language of election for the representation of the financial data models (the simplified Income Statements), as OWL is the now the most commonly used ontology language.

The Mediator Ontology is also represented in OWL, which facilitates the work of querying and storing data for all three ontologies.

With regards to the development of methods for storage and querying of data in the ontologies, these were implemented in Java. The choice of programming language is obvious, as Java is a platform-independent language, meaning that Java code can run on any hardware platform (Windows, Mac, Linux, etc.), thus making it a much more universal choice. Additionally, there are already a variety of existing development API's implemented in Java (Jena, Protégé, etc.), which make it the most sensible choice for development work related to ontologies.

The choice of ATL for the implementation of the model transformation code was already outlined in sub-section 5.6. However, it is important to add again the relevance of choosing Java as the development language. ATL does permit the automatic generation of transformations (once the ATL code for a complete model transformation is available) however, this can only be accomplished via Java methods.

The decision to use SPARQL as querying language was also related in section 6.1. SPARQL is one of the most common querying languages for ontologies, and its similarities to SQL make it a user-friendly choice for the development of ontology querying methods.

Finally, the choice of Protégé as an integration tool was also discussed in section 6.1. The existence of an API for the development of plug-ins, and its wide user base, made it a wise choice as an integration tool and GUI.

6.2.1.1. USE-CASES

In order to evaluate the proof-of-concept, three use cases, consisting of three mapping transformations, were defined. These mapping transformations were deemed to be comprehensive enough, as they illustrate not only the various types of model morphisms possible, but they also illustrate the two main item mapping combinations (1-to-1 and N-to-1).

The transformations are shown below in Table 6.1:

TABLE 6.1 - MAPPING TRANSFORMATION USE-CASES

ID		$Client1.1.1\_1$
$MElems = (a, b)$	a	$USGAAP.IncomeStatement.Sales$
	b	$IFRS.IncomeStatement.Revenue$
$KMType$		$Conceptual$
$MatchClass$		$Naming$
Exp		$"a = b"$
ID		$Client1.1.4\_1$
$MElems = (a, b)$	a	$USGAAP.IncomeStatement.SellingExpenses$
	b	$IFRS.IncomeStatement.SellingGeneralandAdministrativeExpenses$
$KMType$		$Semantics$
$MatchClass$		$Granularity$
Exp		$"a \subseteq b"$
ID		$Client1.1.4\_2$
$MElems = (a, b)$	a	$USGAAP.IncomeStatement.SellingExpenses$
	b	$IFRS.IncomStatemen.SellingGeneralandAdministrativeExpenses$
$KMType$		$InstantiableData$
$MatchClass$		$Granularity$
Exp		$"b = a + USGAAP.IncomeStatement.GeneralExpenses + USGAAP.IncomeStatement.AdministrativeExpenses"$

As it can be seen in Table 6.1, the Use-Cases illustrate the three Knowledge Mapping Types (Conceptual, Semantics and Instantiable Data), as defined by the Mediator Ontology. Additionally, the two top mappings are examples of 1-to-1 transformations, whereas the 'InstatiableData' mapping, provides an example of a N-to-1 transformation.

The simplified Income Statements which serve as the bases for the application scenario do have more mappings than the three use-cases shown above, however, a choice was made not to utilize all the existing mappings, as many would simply be similar in nature to those shown in Table 6.1.

Finally, it is important to point out an exception regarding the generation of ATL Expressions for storage in the Mediator Ontology. It shall only be possible to generate ATL code for 'Instantiable Data' mappings. This has to do with the fact that many of the 'Conceptual' and 'Semantics' Knowledge Mapping Types would not be possible to convert into a factual code expression, as they refer to the mapping of concepts more from a theoretical or semantics point-of-view. However, in the cases where certain 'Semantics' or 'Conceptual' mappings can also be translated into 'Instantiable Data' mappings (as the first example in Table 6.1 illustrates), there can be two separate mappings for the same concept pair.

6.2.2. IMPLEMENTATION STEPS

Having identified the Use-Cases which were implemented, as well as, the technologies which were utilized, this section shall describe the steps which were performed throughout the implementation phase of the demonstrator tool.

6.2.2.1. Step 1- Creation of Simplified Income Statements in OWL

The first step of the implementation phase consisted in the creation of the simplified Income Statements, as shown in section 6.3 in OWL.

The choice of creating the simplified Income Statements from scratch, as opposed to utilizing existing ones in XBRL, has to do with the fact that existing Income Statements in XBRL format would be extremely large and not very practical for demonstration purposes.

Additionally, the utilization of existing Income Statements in XBRL, would require their transformation into OWL. Although this has already been accomplished as published in [29], it would demand several additional steps, which are not really within the main scope of the present dissertation.

6.2.2.2. Step 2- Development of Java Methods for Data Insertion into Mediator Ontology

The second phase of implementation consisted in the development of the Java methods necessary to insert data into the Mediator Ontology.

With the aid of the Protégé API, several methods were developed for the insertion of new data into the Mediator Ontology, such as, methods for the creation of new Morphisms, new Model Elements, new Information Models, etc.

6.2.2.3. Step 3 – Development of Java Methods for Querying the Mediator Ontology

Once the methods for inserting data into the Mediator Ontology were developed, it was then necessary to implement a set of methods which would perform queries on the data stored in the Mediator Ontology.

During this step, the TopBraid Composer tool was utilized. TopBraid is an ontology editor tool, built on top of the Eclipse IDE. One of the features provided by TopBraid is a very user-friendly SPARQL query interface, which allows for quick testing of SPARQL queries. This tool was essential throughout this step, as not only served as learning tool on how to implement SPARQL queries, but it also served as tool for validation of the SPARQL queries once they were implemented through the usage of Java code.

6.2.2.4. Step 4 – Development of ATL Code Generation Methods

Being a relatively new language, ATL has suffered several updates over the time period in which the present dissertation work took place. Given that much of the documentation available was developed in the early stages of the language, many of the tutorials, examples and other learning tools have become outdated throughout the years, thus requiring additional efforts in understanding the workings of the language.

Hence, this step involved many actions, namely:

- Learning the correct installation of the Eclipse IDE in order to support the latest ATL versions, namely all necessary plug-ins, etc.
- Learning about the existing methods which allow for the automatic execution of ATL code, through Java methods which can compile and execute ATL code
- Getting acquainted with the syntax of the OCL language utilized in ATL, through examples and tutorials

After running and performing several tests and examples, it was then possible to get a grasp on how to write the necessary ATL code which is to be generated by the tool.

6.2.2.5. Step 5 – Development of Protégé Tab as an Integration Tool and GUI

The final step in the implementation of the demonstrator tool, consisted in the development of a Protégé Tab.

The purpose of this tab was mainly to serve as user-interface to implement the use-cases which were implemented in this dissertation. Therefore, the tab developed only has very basic functionalities which permit the mapping of the use-cases defined in sub-section 7.1.1.

In order to develop the tab, the code was written in Java using the API provided by Protégé.

6.3. Implementation Testing and Hypothesis Validation

After performing the implementation of the demonstrator, according to the use cases identified earlier, it was necessary to test it in order to verify whether proposed solution meets the requirements outlined, and therefore, validates the hypothesis raised.

In the particular context of software testing, it can be defined as the process of searching for errors in an implementation, so that they may be corrected before making the final software available to the end-user. This search for errors is done by running experiments in a controlled environment, in order to ensure that the software implementation meets the specified requirements, before it can be utilized in a real environment.

Additionally, software testing can be split up into two major categories:

- Functional Testing – This type of testing aims to determine whether the software written implements specific functions, in accordance to user requirements. This type of testing can be done with the aid of Use Cases.
- Non-Functional Testing – The goal of Non-Functional Testing is to evaluate other software related aspects, which are not specific functions, but which are nonetheless relevant, such as, reliability, usability, security, etc.

As the proposed demonstrator was meant for academic purposes only, the main focus of the testing procedures was on functional testing.

The set of tests which were performed were as follows:

- Testing of Java methods for data insertion
- Testing of SPARQL queries
- Testing of Java methods for calling and running SPARQL queries
- Testing of ATL code generated by the application
- Testing of Protégé Tab

The following sub-sections will explain in further detail the tests implemented, and their respective testing methodologies and tools.

6.3.1. Testing of Java Methods for Data Insertion

The first which was conducted was on the Java methods which were developed to insert data into the Mediator Ontology.

The testing methodology consisted of writing a method which, in turn, would call upon the other methods implemented, in order to create insert various instances into the Mediator Ontology of the following classes:

- 'ModelElement'
- 'InformationModel'
- 'Morphism'
- 'MorphismDomain
- 'Executable_Code'

The instances created were varied, in other to represent the major types of Mappings ('InstatiableData', 'Semantics' and 'Conceptual').

Upon running the methods described above, the Mediator Ontology was opened in Protégé, in order to verify if the methods developed inserted the data correctly as expected.

Throughout this testing procedure, it was possible to identify some minor issues which were corrected. The methods now work as expected.

6.3.2. Testing of SPARQL Queries

The second set of tests involved evaluating a set of SPARQL queries, which were necessary to extract data from the Mediator Ontology.

The set of SPARQL queries which was developed involved queries for the following:

- Obtaining all the details associated with a specific Morphism, according to a specified ID
- Obtaining a set of Morphisms associated with a given Model Element or a given Information Model
- Obtaining the ATL Executable Code strings associated with a given Information Model

The queries were first implemented and tested in TopBraid Composer, as described earlier in sub-section 7.2.3. The aim of using a separate tool was to ensure that the queries were well implemented and the results obtained were directly from the queries themselves, and without any 'interference' from other code.

Additionally, the usage of a separate tool also allowed for testing certain improvement in query syntax, thus helping make some of the queries more readable to other parties who may in the future have access to the code.

The set of queries described above was tested successful in TopBraid, before they were integrated into the Java methods.

6.3.3. Testing of Java Methods For Calling and Running SPARQL Queries

Once there were was an assurance provided by the tests from sub-section 8.2 that the SPARQL queries were indeed correct, it was necessary to test their implementation as an integrated part of other Java methods, which could run these queries and provide a means of accessing the query results.

After integrating the queries described in section 8.2 into the appropriate Java Methods, they were tested to ensure that the query results provided were similar to those provided by running the queries in TopBraid Composer. Once more, the tests were successful, as the Java methods returned similar results.

6.3.4. Testing of ATL Code generated by the Application

The next set of tests involved ensuring that the ATL expressions generated by the demonstrator were indeed correct and performed the model transformation once they were inserted and ran in a complete ATL model transformation code.

In order to validate the code correctness, it was necessary to create two Ecore Metamodels, representing the two simplified Income Statements. The purpose for creating these has to do with the fact that ATL works only with Metamodels in Ecore format. Therefore, these metamodels were implemented using the tools provided by the Eclipse Modeling Framework, which allow for the creation of Ecore metamodels, in a way which is very similar to the way Protégé allows a user to develop an ontology. The metamodel can be created, following a tree-like structure, similar to the structure of an ontology. In addition to the metamodels in Ecore, a simple Income Statement in XMI was developed, in order to serve as an input model.

The expressions generated by the application were then incorporated into a complete ATL model transformation code, as the purpose was to validate whether the code generated by the demonstrator tool was correct and did perform the correct transformation in accordance to the morphism expression provided.

The code generated by the Use Cases described in sub-section 7.1 was tested and the transformations generated in ATL produced the correct model morphism, as described in the Use Cases.

6.3.5. TESTING OF PROTÉGÉ TAB

The final testing procedures were focused on the evaluating the performance of the Protégé Tab, as means of integrating all the code and providing a GUI to the user.

The testing involved repeating many of the above procedures, such as testing the insertion methods, the query methods and the ATL code generation methods. However, the testing now involved the invocation of these methods, via the GUI provided by the Tab.

Once the method invocations were tested, it was then necessary to test some of the GUI components (textboxes, etc.), which are used to receive user input to be inserted into the Ontology. These were tested to ensure that the correct data is being extracted and inserted into the Mediator Ontology, as desired.

All of the above testing procedures were successful.

7. Conclusions and Future Work

7.1. Conclusions

The solution proposed by the present dissertation, aimed to provide the groundwork for a framework which can facilitate the seamless exchange and analysis of financial accounting data.

The Use Cases provided, although based on a simplified Income Statement, were however, sufficient to prove that through the use of ontologies, model transformation technologies and the Model Morphism paradigm, it is possible to convert financial accounting data from one accounting standard to another.

Based upon this work, it should be possible to develop a system, a sort Model Transformation Warehouse, where it shall be possible to store complete libraries of Model Morphisms between several accounting standards. If implemented, once such a system is made available for public use, it could greatly facilitate the work of financial analysts, investors and regulators, as they could provide a financial report in one accounting standard (ie US GAAP), and have the system transform the said financial report into its IFRS version.

Even though this vision may still be a few years down the road, the tools which already exist today can certainly make their contribution towards more transparency in the financial sector.

7.2. Future Work

The work accomplished through this dissertation can be improved upon with regards to the following aspects:

1. Further automation of ATL code – The solution provided only generates the ATL code expression for a single mapping. The next step would be to generate the complete ATL transformation code, given the set of Information Models being transformed. Once this has been accomplished, it should be possible to have the application compile and execute the ATL code generated by the transformation, thus making the tool a sort of ‘all-in-one’ application, where ontologies are mapped and the subsequent transformation is accomplished.
2. Transformation of OWL ontologies into Ecore Metamodels – In order to make the full automation of ATL code possible, it is necessary that the OWL ontologies being mapped, also exist in Ecore format. There is some work already done in this arena by Guilluame Hillaret (Ref), hence, it should be possible to build upon it and integrate this feature into the final application
3. Transformation of OWL instances into XMI models – In parallel with the work required in item 2, the final step would be to extract the instances from the OWL models and convert them into XMI models, in order to be compatible with the ATL transformation.

4. Complete Protégé tab – The tab developed only implements some very basic functionalities which were sufficient to cover the use-cases of the dissertation. Therefore, the following step would be to build upon what was implemented and include more functionalities, to allow for more mapping types, more user-friendly interface, work with other types of data models, etc.

As a final note, that parts of the work performed for this dissertation shall be used within the context of CRESCENDO, a FP7 EU-funded project, and ISOFIN, a Portuguese-funded QREN project.

8. BIBLIOGRAPHY

[1] Patsuris, Penelope (2002), The Corporate Scandal Sheet, Forbes, viewed 20 October 2009, http://www.forbes.com/2002/07/25/accountingtracker.html

[2] Bogoslaw, David (2008), How to Fix Financial Reporting, Business Week, viewed 14 June 2009, http://www.businessweek.com/investor/content/nov2008/pi2008119_257282.htm

[3] Zhu, Hongwei and Stuard Madnick (2007), “Semantic Integration Approach to Efficient Business Data Supply Chain: Integration Approach to Inter-Operable XBRL”, MIT Sloan School of Management Research Paper Series

[4] Cranefield, Stephen and Pan, Jin (2007), Bridging the gap between the model-driven architecture and ontology engineering. Int. J. Hum.-Comput. Stud. 65, 7 (July 2007), 595-609.

[5] Knowledge Web, (2005). Benchmarking the Interoperability of Ontology Development Tools, viewed 5 May 2009, http://knowledgeweb.semanticweb.org/benchmarking_interoperability/rdfs/index.html

[6] Camarinha-Matos, Luís (2010), “Scientific Research Methodologies and Techniques - Unit 2: Scientific Method”, PhD Program in Electrical and Computer Engineering

[7] XBRL International (2005), XBRL's History, XBRL International, viewed 10 June 2009, http://www.xbrl.org/history-print.aspx

[8] RR Donnelley (2007), XBRL Reference Guide: 1940 Act Companies, RR Donneley, viewed 20 June 2009, http://www.rrdonnelley.com/wwwFinancial/Downloads/PDF/Investment%20Markets%20XBRL%20Guide%202007.pdf

[9] FFIEC (2006), Improved Business Process Through XBRL: A Use Case for Business Reporting, Federal Financial Institutions Examination Council, viewed 21 June 2009, http://www.xbrl.org/Business/Regulators/FFIEC-White-Paper-31Jan06.pdf

[10] Rezaee, Zabihollah, Rick Elam and Ahmad Sharbatoghlie (2001), “Continuous auditing: the audit of the future”, Managerial Auditing Journal, 16(3):150-158

[11] Searcy, DeWayne L., Terry J. Ward and Jon B. Woodroof (2009), “Continuous reporting benefits in the private debt capital market”, International Journal of Accounting Information Systems, Volume 10, Issue 3, September 2009, Pages 137-151

[12] PC Magazine (2009), XBRL Definition from PC Magazine Encyclopedia, PC Magazine, viewed 22 September 2009, http://www.pcmag.com/encyclopedia_term/0,2542,t=XBRL&i=54994,00.asp

[13] KPMG (2003), Step by Step XBRL 2.0 Tutorial, viewed 10 June 2009, http://www.us.kpmg.com/microsite/xbrl/train/launch.htm

[14] Bryant University (2005), “What is the History of XBRL?”, Bryant University, viewed 11 June 2009, http://web.bryant.edu/~xbrl/xbrl/history.htm

[15] AICPA (2006), XBRL, American Association of Certified Public Accountants, viewed 11 June 2009, http://www.aicpa.org/Professional+Resources/Accounting+and+Auditing/BRAAS/XBRL.htm

[16] Strader, Troy J. (2007), XBRL Capabilities and Limitations, The CPA Journal, viewed 20 September 2009, http://www.nysscpa.org/cpajournal/2007/1207/essentials/p68.htm

[17] IASB (n.d.), XBRL Fundamentals, International Accounting Standards Board, viewed 12 June 2009, <http://www.iasb.org/XBRL/Resources/Fundamentals.htm>

[18] W3C XLink (2001), XML Linking Language (XLink) Version 1.0, World Wide Web Consortium, viewed 14 June 2009, http://www.w3.org/TR/xlink/

[19] W3C XPointer (2001), XML Pointer Language (XPointer) Version 1.0, World Wide Web Consortium, viewed 14 June 2009, http://www.w3.org/TR/WD-xptr

[20] BRAG (2009), XBRL Formulas Trainings, Business Reporting - Advisory Group, viewed 20 September 2009, http://www.br-ag.eu/xbrlformulastrainings

[21] Bonsón E., Cortijo V., and Escobar T. (2009), Towards the global adoption of XBRL using International Financial Reporting Standards (IFRS), International Journal of Accounting Information Systems, Volume 10, Issue 1, March 2009, Pages 46-60

[22] Riordan, Diane and Riordan, Michael (2009), IFRS vs US GAAP: A Sixty Minute Waltz in the Classroom, James Madison University

[23] Daimler, Annual Report (2009), Daimler, viewed 23 March 2010, http://www.daimler.com/Projects/c2c/channel/documents/1813321_DAI_2009_Annual_Report.pdf

[24] Nissan, Annual Report (2009), Nissan, viewed 23 March 2010, http://www.nissan-global.com/EN/DOCUMENT/PDF/AR/2010/AR2010_E_All.pdf

[25] Garshol, Lars Marius (2004), Metadata? Thesauri? Taxonomies? Topic Maps? - Making sense of it all, viewed 2 April 2010, http://www.ontopia.net/topicmaps/materials/tm-vs-thesauri.html

[26] Gruber, T.R. (1993), “A translation approach to portable ontology specifications,” Knowledge Acquisition, Vol. 5, no. 2, pp. 199–220.

[27] Willpower Information (2009), Glossary of terms relating to thesauri and other forms of structured vocabulary for information retrieval, viewed 2 April 2010, http://www.willpowerinfo.co.uk/glossary.htm

[28] Melli, Gabor (2009), Ontology Definition, Gabor Melli's Research Knowledge Base, viewed 3 April 2010, http://www.gabormelli.com/RKB/Ontology

[29] Bao, Jie, Rong, Graham, Li, Xian and Ding, Li (2010), Representing financial reports on the semantic web: a faithful translation from XBRL to OWL. In Proceedings of the 2010 international conference on Semantic web rules (RuleML'10), Mike Dean, John Hall, Antonino Rotolo, and Said Tabet (Eds.). Springer-Verlag, Berlin, Heidelberg, 144-152

[30] INTEROP NoE (2005), Deliverable MoMo.2 - TG MoMo Roadmap. InterOP.

[31] Agostinho C, Sarraipa J, D'Antonio F, et al (2007), Enhancing STEP-based interoperabity using model morphisms. In: 3rd International Conference on Interoperability for Enterprise Software and Applications (I-ESA 2007)

[32] Delgado M., Agostinho C., Malo P. and Jardim-Gonçalves R., (2006) A framework for STEP-based harmonization of conceptual models. 3rd International IEEE Conference on Intelligent Systems (IEEE-IS'06), Westminster, Jul 4-6 , 2006.

[33] INTEROP NoE (2006), Deliverable D.A3.3 - Semantic Annotation language and tool for Information and Business Processes. InterOP.

[34] Delgado, Marco (2008), Harmonisation of STEP and MDA conceptual models using Model Morphisms. MSc thesis presented at Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

[35] Dauben, Joseph W. and Cantor, Georg (1979), His Mathematics and Philosophy of the Infinite. Boston: Harvard University Press, ISBN 978-0-691-02447-9.

[36] Bernstein , Philip A. (2003), Applying Model Management to Classical Meta Data Problems, First Biennial Conference on Innovative Data Systems Research, California, January 2003.

[37] Giunchiglia, F. Yatskevich and M. Shvaiko, P. (2007), Semantic Matching: Algorithms and Implementation, J. Data Semantics 9: 1-38.

[38] Blaha, M. and Premerlani, W. (1996). A catalog of object model transformations. Proceedings of WCRE, Volume 96, pp 87.

[39] Agostinho, C., Sarraipa, J., Goncalves, D. and Jardim-Goncalves, R. (2011). Tuple-based semantic and structural mapping for a sustainable interoperability. In the proceedings of DoCEIS'11 – 2nd Doctoral Conference on Computing Electrical and Industrial Systems, 21-23 February 2011 – Costa da Caparica, Lisbon - Portugal

[40] Sarraipa, J., Zouggar, N., Chen, D and Jardim-Goncalves, R. (2007). Annotation for Enterprise Information Management Traceability. In Proceedings of IDETC/CIE ASME 2007.

[41] OMG (2001), OMG Model Drive Architecture, viewed 5 May 2010, http://www.omg.org/mda/

[43] OMG (2001), MDA Specifications, viewed 5 May 2010, http://www.omg.org/mda/specs.htm

[44] Poole, John (2001), Model-Driven Architecture: Vision, Standards And Emerging Technologies, Position Paper Submitted to ECOOP 2001, Workshop on Metamodeling and Adaptive Object Models

[45] OMG (1997), UML Resource Page, viewed 6 May 2010, http://www.uml.org/

[46] SearchSoftwareQuality.com (2000), Unified Modeling Language Definition, viewed 6 May 2010, http://searchsoftwarequality.techtarget.com/definition/Unified-Modeling-Language

[47] OMG (2008), Meta Object Facility (MOF) 2.0 Query/View/Transformation, v1.0 Specifications, viewed 8 May 2010, http://www.omg.org/spec/QVT/1.0/PDF/

[48] Haase, A., Volter, M., Efftinge, S., and Kolb, B. (2007), Introduction to openArchitectureWare 4.1. 2, in MDD Tool Implementers Forum.

[49] ikv++ technologies (2007), medini QVT homepage, viewed 20 June 2009, http://projects.ikv.de/qvt

[50] France Telecom (2007), Smart QVT homepage, viewed 22 June 2009, http://smartqvt.elibel.tm.fr/ (no longer available)

[51] Reyes, Eva, Rodriguez, Daniel and Dolado, Javier (2007), Overview of XBRL technologies for decision making in Accounting Information System, Paper presented at ADIS - Decision Support in Software Engineering 2007 Workshop, http://www.sc.ehu.es/jiwdocoj/remis/cfpadis2007.htm

[52] W3C, SPARQL Query Language for RDF (2008), viewed 5 October 2010, http://www.w3.org/TR/rdf-sparql-query/