APPLYING ENTERPRISE MODELS AS INTERFACE FOR INFORMATION SEARCHING

Auriol Degbelo
Tanguy Matongo

MASTER THESIS 2009
Informatics
APPLYING ENTERPRISE MODELS AS INTERFACE FOR INFORMATION SEARCHING

Auriol Degbelo
Tanguy Matongo

This thesis is performed at Jönköping University, School of Engineering within the subject area information searching. The thesis is part of the university’s master’s degree. The authors are responsible for the given opinions, conclusions and results.

Supervisor: Thomas Albertsen
Examiner: Vladimir Tarasov

Credits point: 30 ECTS (D-level)

Date: 2009-12-11

Archive number
Abstract

Nowadays, more and more companies use Enterprise Models to integrate and coordinate their business processes with the aim of remaining competitive on the market. Consequently, Enterprise Models play a critical role in this integration enabling to improve the objectives of the enterprise, and ways to reach them in a given period of time. Through Enterprise Models, companies are able to improve the management of their operations, actors, processes and also to improve communication within the organisation.

This thesis describes another use of Enterprise Models. In this work, we intend to apply Enterprise Models as interface for information searching. The underlying needs for this project lay in the fact that we would like to show that Enterprise Models can be more than just models but can be used in a more dynamic way which is through a software program for information searching. The software program aims first at extracting the information contained in the Enterprise Models (which are stored into a XML file on the system). Once the information is extracted, it is used to express a query which will be sent into a search engine to retrieve some relevant documents for the query and return them to the user.

The thesis was carried out over an entire academic semester. The results of this work are a report which summarizes all the knowledge gained into the field of the study. A software has been built to serve as a proof of testing the theories.
Acknowledgements

We would like to thank Thomas Albertsen for his support, advices and guidance during this thesis work. We would also like to thank Vladimir Tarasov for his important participation and availability. Finally, we thank Kurt Sandkuhl for his helpful contribution to this thesis work.
Key words

Information Retrieval, Information Extraction, semantic integration, query expansion, ontology mapping, knowledge engineering.
Contents

1 Introduction ........................................................................................................... 1
  1.1 BACKGROUND ............................................................................................... 1
  1.2 PURPOSE/OBJECTIVES ............................................................................... 1
  1.3 LIMITATIONS ............................................................................................... 2
  1.4 THESIS OUTLINE .......................................................................................... 2

2 Theoretical Background ......................................................................................... 4
  2.1 ENTERPRISE MODELS .................................................................................. 4
    2.1.1 Definition ................................................................................................. 4
    2.1.2 Use and advantages ............................................................................... 4
    2.1.3 Tools and Storage .................................................................................... 5
    2.1.4 Definition and characteristics of XML file .............................................. 6
  2.2 INFORMATION EXTRACTION ......................................................................... 9
    2.2.1 Definition ................................................................................................. 9
    2.2.2 Information extraction versus Data extraction ....................................... 9
    2.2.3 Information extraction approaches ....................................................... 10
  2.3 QUERY CONSTRUCTION ............................................................................... 13
    2.3.1 Definition of a query ............................................................................... 13
    2.3.2 Query construction ............................................................................... 13
  2.4 INFORMATION RETRIEVAL .......................................................................... 17
    2.4.1 Definition ................................................................................................. 17
    2.4.2 Information Retrieval versus Information Extraction ............................ 18
    2.4.3 Information Retrieval and Document Preprocessing ............................... 20

2.5 USER INTERFACE ............................................................................................ 22
  2.5.1 Definition ................................................................................................. 22
  2.5.2 A typical User Interface ......................................................................... 23

3 Methods ............................................................................................................... 24

4 Implementation ..................................................................................................... 26
  4.1 DESCRIPTION OF THE CASE .................................................................. 26
    4.1.1 The object type process ........................................................................ 26
    4.1.2 The object type person (as substitute for role) ...................................... 26
    4.1.3 The object type initiative (as substitute for competences) ................. 26
  4.2 MODULE1: ENTERPRISE MODEL ANALYZER ........................................... 27
    4.2.1 Concept building .................................................................................... 27
    4.2.2 System building ..................................................................................... 27
  4.3 MODULE2: QUERY GENERATOR ................................................................. 29
    4.3.1 Concept building .................................................................................... 29
    4.3.2 System building ..................................................................................... 29
  4.4 MODULE3: USER INTERFACE ................................................................. 35
  4.5 SEARCH ENGINES (GOOGLE, DBLP) AND DATABASE ...................... 36
    4.5.1 Search engines ....................................................................................... 36
    4.5.2 Database ................................................................................................. 38

5 Results .................................................................................................................. 40
  5.1 ENTERPRISE MODEL ANALYZER ............................................................... 40
  5.2 QUERY GENERATOR .................................................................................... 43
    5.2.1 Query construction and execution for Google ..................................... 44
    5.2.2 Query construction and execution for DBLP ...................................... 49
    5.2.3 Query construction and execution for a MS Access database ............ 51
List of Figures

FIGURE 1: DIFFERENT STEPS FOLLOWED DURING THE THESIS WORK ........................................... 3
FIGURE 2: EXAMPLE OF XML ELEMENT ..................................................................................... 6
FIGURE 3: EXAMPLE OF NESTED ELEMENTS ............................................................................... 8
FIGURE 4: DIFFERENCE BETWEEN DATA AND INFORMATION .................................................. 10
FIGURE 5: EXAMPLE OF ONTOLOGIES AND THEIR MAPPINGS ............................................. 15
FIGURE 6: IR SYSTEM .................................................................................................................. 18
FIGURE 7: INFORMATION RETRIEVAL VS INFORMATION EXTRACTION ................................. 19
FIGURE 8: A SIMPLIFIED DIAGRAM OF THE STANDARD MODEL OF THE INFORMATION ACCESS PROCESSES .......................................................... 22
FIGURE 9: PROCESS FOR BUILDING AN INFORMATION EXTRACTION SYSTEM ....................... 27
FIGURE 10: AN ALGORITHM FOR INFORMATION EXTRACTION ............................................... 28
FIGURE 11: QUERY CONSTRUCTION FOR GOOGLE AND DBLP(METHOD1) ............................ 31
FIGURE 12: QUERY CONSTRUCTION FOR GOOGLE AND DBLP(METHOD2) ............................ 31
FIGURE 13: GENERIC SQL QUERY WITH A SINGLE COMPETENCE AS INPUT ........................ 32
FIGURE 14: GENERIC SQL QUERY WITH MULTIPLE COMPETENCES AS INPUT ...................... 33
FIGURE 15: QUERY CONSTRUCTION FOR THE MS ACCESS DATABASE (METHOD1) ................ 33
FIGURE 16: QUERY CONSTRUCTION FOR THE MS ACCESS DATABASE (METHOD2) ............... 33
FIGURE 17: AN ALGORITHM FOR QUERY EXPANSION ............................................................ 35
FIGURE 18: STRUCTURE OF THE DATABASE ............................................................................... 39
FIGURE 19: SELECTION OF A TROUX FILE ............................................................................... 40
FIGURE 20: SELECTION OF THE STARTING AND ENDING CONTAINERS .................................. 40
FIGURE 21: LIST OF ROLES FOR THE CASE DESCRIBED ABOVE ........................................... 41
FIGURE 22: COMPETENCES REQUIRED FOR AN ONTOLOGY EXPERT ................................... 41
FIGURE 23: INTERFACE FOR QUERY PERSONALIZATION ......................................................... 43
FIGURE 24: ADDITION OF 'CONTEXTUAL KEYWORDS' (2 QUERY TERMS, BOOLEAN AND) .......... 46
FIGURE 25: ADDITION OF 'CONTEXTUAL KEYWORDS' (2 QUERY TERMS, BOOLEAN OR) ........ 46
FIGURE 26: ADDITION OF 'CONTEXTUAL KEYWORDS' (3 QUERY TERMS, BOOLEAN AND) ....... 47
FIGURE 27: ADDITION OF 'CONTEXTUAL KEYWORDS' (3 QUERY TERMS, BOOLEAN OR) ....... 47
FIGURE 28: ADDITION OF 'CONTEXTUAL KEYWORDS' (4 QUERY TERMS, BOOLEAN AND) ...... 48
FIGURE 29: ADDITION OF 'CONTEXTUAL KEYWORDS' (4 QUERY TERMS, BOOLEAN OR) ...... 48
FIGURE 30: FIRST FIVE RESULTS OF THE QUERY 'ONTOLOGY' IN DBLP ............................... 49
FIGURE 31: AN EXAMPLE OF TUPLE .......................................................................................... 51
FIGURE 32: SQL QUERY WITH 'CONTEXT' AND 'PATTERN' AS INPUT (BOOLEAN AND) ........ 51
FIGURE 33: RESULT OF THE QUERY 'CONTEXT AND PATTERN' IN THE MS ACCESS DATABASE ............................................................................................................ 51
FIGURE 34: AN EXAMPLE OF COMPLETE PROFILE OF A PERSON IN THE MS ACCESS DATABASE ............................................................................................................. 52
FIGURE 35: SQL QUERY WITH 'CONTEXT' AND 'PATTERN' AS INPUT (BOOLEAN OR) ............ 52
FIGURE 36: RESULT OF THE QUERY 'CONTEXT OR PATTERN' IN THE MS ACCESS DATABASE ......................................................................................................................... 53
FIGURE 37: BUTTONS 'BACK' AND 'VIEW HISTORY' .................................................................. 55
FIGURE 38: AN EXAMPLE OF HISTORY OF THE USER'S ACTIONS ........................................... 55
FIGURE 39: AN EXAMPLE OF MULTIPLE SELECTION OF ROLES AND THE RESULT ................ 56
FIGURE 40: FROM AN EM TO CONCRETE OBJECTS ................................................................. 61
FIGURE 41: AEMIS PART OF THE LARGER SYSTEM ................................................................. 71
FIGURE 42: QUERY 'FORMAL SEMANTIC MODEL' IN DBLP ................................................... 89
FIGURE 43: QUERY 'MODEL SEMANTIC FORMAL' IN DBLP ................................................... 89
FIGURE 44: TROUX'S ENTERPRISE MODEL USED TO TEST THE SOFTWARE ......................... 91
List of Abbreviations

DBLP  Digital Bibliography & Library Project
DOS  Disk Operating System
DTD  Data Type Definition
EM  Enterprise Model
EMSE  Enterprise Modeling Software Environment
GML  Generalized Markup Language
GUI  Graphical User Interface
IE  Information Extraction
IR  Information Retrieval
HTML  HyperText Mark-Up Language
UI  User Interface
SDM  System Development Method
SGML  Standard Generalized Markup Language
SQL  Structured Query Language
XML  Extensible Markup Language
W3C  World Wide Web Consortium
1 Introduction

The growing use of Enterprise Modeling in companies has lead researchers to think about different ways to make the use of Enterprise Models more dynamic. The objective of this chapter is to introduce our research problem and how we manage to solve it during our thesis work. It aims at giving an overview of how the thesis was conducted in order to solve the research problem, and showing the limitations or scope of our work. There is also a short outline of the remaining parts of the report.

1.1 Background

As mentioned earlier, the growing use of Enterprise Models has led us to search for more dynamic use of those models by applying it as interface for information searching. Our research questions we could infer from this problem area are:

- How to extract information from Enterprise Models (built using the tool Troux) which are stored in a XML file on the system?
- How to express the extracted information into a query for search engines?

1.2 Purpose/Objectives

The thesis subject aims at exploring the use of Enterprise Models as interface for information searching. This task requires at least 3 steps:

- to extract information from the Enterprise Models specifying the request to be used for information searching
- to generate and execute a query based on the request in the desired information system
- to store and visualize the results of the query

In this thesis work, we will solve our research problem from two perspectives both theoretical and technical.

From the theoretical perspective, our work will be a summary of the existing knowledge in our main field of work which are Enterprise Models, Information Extraction and Information Retrieval. Consequently, different approaches for extracting the request from the models and for expressing the request in a query shall be discussed with their effects on the quality of the retrieved results.

From a technical perspective the work aims at illustrating concrete application of the knowledge gained from the existing literature. For that, a software will be developed which will support the three above steps. The development will include a more detailed requirement analysis.
1.3 Limitations

There are few limitations regarding the task we have to perform in our work in order to make the practical part of the project achievable.

The software developed should be workable fulfilling the three steps mentioned in the section above. Consequently, the software developed aims at finding information in two types of search engines - Google, DBLP (Digital Bibliography & Library Project) - and a database, based on a query built beforehand. No detailed specification has been outlined regarding the type of the search engines and the database. Particularly, for the latter, we did not have to follow a specific database schema, or use specific instruction concerning the fields contained in the tables. We have just built a workable database which would illustrate the most the main goal of the software.

The thesis work aimed at developing a software based on different approaches for extracting the request in order to see their effects on the quality of the retrieved results. Not all the approaches found will be used. We will limit the different approaches to two Boolean operators: the operator AND and the operator OR. Furthermore, we will add some specific terms to our query to make them more accurate for the search in the search engines: those terms are CV and PEOPLE.

1.4 Thesis outline

The thesis is divided into four main parts. Firstly, we will present all the theories related to our research problem. In this chapter, we will first present what Enterprise Models are and how are they used in today’s companies. In our case, as the Enterprise Models are stored in XML file, we will briefly introduce the XML language and how the storage of information is performed in XML document. Then, we will describe the different techniques that allow extracting information from a file for a specific purpose. Once, the information is extracted, it is used to express the request in a query for search engines and database. This first chapter follows a track that will guide us throughout the report. The track is made of three steps mentioned in the previous section which are extracting information from Enterprise Models, generating and executing a query based on the extracted information and storing and visualizing the result of the query. This track is illustrated in the figure 1 below.

Secondly, we will introduce the methods we have been using in order to solve the research problems. This chapter contains the research design and research method we have chosen detailing the different steps of the implementation and the results expected at each steps.

Then, follows the implementation itself. Based on the different steps of our research design and method, we have built the software. This chapter introduces in a more detailed way the different modules of the software, their functionalities and how they interact with each other in order to fulfil the main requirement of the software program. The implementation will result in different prototypes that will illustrate the different approaches we have used. As the implementation is an illustration of the theoretical background, it also follows the track provided in the figure 1 below.

Finally we will present the results of our implementation. The different approaches for retrieving information will be discussed through the prototypes we have built. We will show the effects of each approach on the retrieved document.
Figure 1: Different steps followed during the thesis work
2 Theoretical Background

2.1 Enterprise Models

2.1.1 Definition

‘Modeling’ refers to a systematic set of actions taken in order “to describe a set of abstract or concrete phenomena in a structured and, eventually, in a formal way. [...] Describing, modeling, and drawing is a key technique to support human understanding, reasoning, and communication” (Bubenko, 1992). Modeling can be used in various domains like mathematics, geology, economics, climate, etc. When applied to enterprise - an organization created for business ventures (Wordnet, 2009) - it is called Enterprise Modeling which describes enterprise objectives, activities, information resources, processes, actors, products, requirements… as well as relationships between those entities (Fox & Gruninger, 1998).

The outcomes of Enterprise Modeling activity are Enterprise Models (EM). They are the “representations of the pertinent aspects of an organization’s structure and operation” (Wolverton, 1997). Presley (1997) also defined Enterprise Model as “a symbolic representation of the enterprise and the things that it deals with. It contains representations of individual facts, objects, and relationships that occur within the enterprise” (as cited in L. Whitman).

Enterprise Models can be both descriptive and definitional with the aim of achieving model-driven enterprise design, analysis, and operation. “From a design perspective, an Enterprise Model should provide the language used to explicitly define an enterprise [...]. From an operations perspective, the Enterprise Model must be able to represent what is planned, what might happen, and what has happened. It must supply the information and knowledge necessary to support the operations of the enterprise, whether they are performed by hand or machine. It must be able to provide answers to questions commonly asked in the performance of tasks” (Fox & Gruninger, 1998).

2.1.2 Use and advantages

Once they have been created, Enterprise Models can have a variety of usages as well as lots of benefits for the company.

According to an earlier study made by M. Wolverton (1997), some examples among many of the use of Enterprise Models are:

- Insight: by abstracting away the complexity of the overall organization, Enterprise Models can help to improve the understanding and the organization’s functioning.
- Communication: they can allow all members of the organization to see views of the enterprise based on a common picture.
- Enactment: they can be of great worth in designing and implementing organization’s processes.
Some other examples of the use of Enterprise Models mentioned by L. Madarász, M. Timko and M. Raček (2004) are:

- Changes of organizational structure: to better suit to relevant business activities.
- Help of management: to gain complete view of the business organization.
- Business process reengineering: in the meaning of efficiency, etc.

As mentioned before, Enterprise Modeling can have many advantages for each employee in the company and for the entire organization.

Bubenko, Persson, Stirna (2001) assert that one of the advantages of Enterprise Modeling is the effect on the participants. While modeling, the participants can get better understanding about the organization, its main goals, the different processes, how the processes are performed… The participants can also improve their capability to find solutions to problems in a participative way and by consensus of all the participants. Therefore, Enterprise Modeling enhances communication between the actors and it facilitates the process of organizational learning.

Another advantage of using EM is that it could help to convey semantic unification. It may happen that people in the company use different terms to express the same thing or they can use the same term to express things completely different. Enterprise Modeling will offer a mutually agreed language to the different actors.

### 2.1.3 Tools and Storage

Currently, there are many different types of Enterprise Modeling Software Environment (EMSE). Even if the tendency is towards environment supporting several languages and views, most EMSE implement a set of fixed languages and support a fixed methodology. No tools presently support integration of its own models with models from other tools.

Some examples of EMSE are Mo²Go (used in production companies, service enterprises, government services (e.g. police) and in the health sector), Popkin System Architect (one of the leading tools for Enterprise Architecture technology), and Metis¹ (ranked among the top three providers of products for Enterprise Architecting in the US market) (Petit & Doumeingts, 2002).

In this thesis work, since the model we are starting from is a Troux EM, we will have a closer look on this EMSE.

Troux is a suite of products for enterprise knowledge modeling, knowledge architecting and knowledge management. It offers modeling, navigation, viewing and integration capabilities which allow industry to build and manage its Enterprise Knowledge Architecture. Troux targets enterprise knowledge management independent of particular methods. Analysis, calculations and specific methods are interfaced from other applications when needed.

---

¹ This EMSE was initially provided by Computas Technologies. It was later renamed Troux after the acquisition of Computas Technologies by Troux Technologies in 2005 (Troux Technologies, 2009).
Troux supports holistic modeling and viewing. It enables distributed concurrent model, sub-model, view development and management. This EMSE provides also capabilities for users to develop and extend their modeling languages and to build their own meta-models and methodologies. Finally, it makes the integration of legacy systems easier and helps to use models in support of creative work (Petit & Doumeingts, 2002).

Troux’s languages (for instance ITM, GEM, and EEMM) permit easy definition of data interfaces to databases and other systems using international standards (e.g. XML, URI, HTTP, SOAP, and UNICODE). The integration is supported by standardized data services and APIs.

Once a model has been created with this EMSE, it can be stored as a file in a .xml format. Using XML format can be an advantage because as mentioned by Don Hodge “that means that other XML tools can read Troux models if they have access to Troux’s DTD (Data Type Definition)”

### 2.1.4 Definition and characteristics of XML file

#### 2.1.4.1 Definition

The Extensible Markup Language (XML) is a language that is used to represent data in a way that it should be easily shared among different applications running on different operating systems (Keogh, Davidson and Ken, 2005).

S. Jacob (2009) mentioned that XML by itself does not do anything other than information storage. According to this author “XML is actually a metalanguage, so you can use it to create other markup languages” (Jacob, 2009, p2). That is why XML is defined as an *extensible* language. The term *markup* means that the languages created use tags to surround or mark up text. In effect, “a markup language is a technique for marking, or tagging, content –text, graphics or other elements- with codes to identify it for some secondary purpose or application” (Maivald & Palmer, 2007, p1). What is called tag is in fact “a piece of text contained within left and right angle brackets (<>)” (Maivald & Palmer, 2007, p1).

An example of XML element with the tags is shown below:

![Example of XML element](Maivald & Palmer, 2007, p1)

**Figure 2:** Example of XML element (Maivald & Palmer, 2007, p1)
2.1.4.2 Origins and goals of XML

The concept of markup languages has been introduced by IBM in 1960’s with the creation of Generalized Markup Language (GML). “GML was IBM’s attempt at providing users a means for exchanging data without losing its structure. It was so successful that it germinated the Standard Generalized Markup Language (SGML) which became a standard method for sharing data” (Maivald & Palmer, 2007, p3).

In the early 1990’s Tim Berners-Lee, a researcher at the CERN laboratory in Switzerland, created HTML based on a subset of the SGML language. HTML was originally designed to be a display language; HTML does not provide any means to handle data-intensive applications. Therefore, a group of researchers, the XML Working Group formed under the World Wide Web Consortium (W3C), began the development of an alternative language. The first version of XML made its debut on February 10, 1998.

The design goals for XML according to the W3C (2009) are:

- XML shall be straightforwardly usable over the Internet.
- XML shall support a wide variety of applications.
- XML shall be compatible with SGML.
- It shall be easy to write programs which process XML documents.
- The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
- XML documents should be human-legible and reasonably clear.
- The XML design should be prepared quickly.
- The design of XML shall be formal and concise.
- XML documents shall be easy to create.
- Terseness in XML markup is of minimal importance.

In other words, XML aims at allowing SGML to be served, received, and processed on the web in the same way as HTML. Therefore, XML plays an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere.

2.1.4.3 Storing information in XML document

As we mentioned before, XML is a metalanguage meaning that “XML has the ability to represent the semantics of data in a structured, documented, machine-readable form” (Egnor & Lord, 2000). It is also a means of storing information. According to S. Jacob (2009) XML documents work best with structured information similar to what can be found in a database. The only difference is that “instead of breaking the information into tables and fields, elements and tags describe the data” (Jacob, 2009, p2).

These elements and tags can be nested inside each other in order to create hierarchy of parent and child element (Jacob, 2009, p2). Therefore, a typical XML document contains “nested elements, which implies the relationship one tag has to other tags” (Keogh et al, 2005). An example of nested elements is provided below where the element contact is parent to the elements name, address and phone.
2.1.4.4 Why to use XML

Nowadays, XML has grown and spread through different technologies and into different companies. “XML is the glue that binds these disparate entities together and makes a whole world of data-intensive applications possible” (Maivald & Palmer, 2007, p2). The growing use of XML is due to many advantages this language offers to the users. Some of these advantages are listed below:

- XML is simple and flexible: the rules of creating XML documents are very simple, “without any training in IT, it is possible to create a set of XML markup tags and use them to build an XML document” (Keogh et al., 2005, p32). The latter (Keogh et al, 2005) also argue that what makes XML flexible is the ability to update and structure the XML document without breaking existing processes.

- XML is descriptive: S. Jacob (2009) states that because it is possible for the user to use his own tag, XML document becomes a description of his data. The descriptiveness of XML documents is also underpinned by the use of Document Type Definition (DTD); “before an application can read an xml document, it must learn what XML markup tags the document uses. It does this by retrieving the document type definition. The DTD identifies markup tags that can be used in an XML document and defines the structure of those tags in the XML document” (Keogh et al., 2005, p26)

- XML is precise: XML is a precise standard. If the XML will be read by an XML parser, it should be “well-formed” meaning that the XML document has correct XML syntax (W3C, 2009).

Figure 3: Example of nested elements (Jacob, 2009, p3)
2.2 Information Extraction

In the previous section, we have seen that XML is a language used to exchange electronically stored information. Exchanging this information among different applications and systems implies its extraction from one file (or system) and use it in another file (or system). This process is called Information Extraction.

2.2.1 Definition

As mentioned earlier, the growing use of electronically stored documents has led human beings to find some ideal information distribution within organizations. M. Wolverton (1997) argues that the distribution of information within an organization should be “timely, selective and to some degree automatic”. In other words, human beings should find some automatic system that should be able to “select” the right information for the right purpose. The process of “picking” information from an information source (e.g. a document collection) and using it afterwards for a purpose is called information extraction (IE).

Information extraction is defined by Freitag (1998) as “the problem of identifying the text fragments that answer standard questions defined for a document collection”. Another definition is provided in another study by Engels and Bremdal (2000), who identified information extraction as “the process of extracting information from texts. Information Extraction typically leads to the introduction of a semantic interpretation of meaning based on the narrative under consideration”. Information extraction involves multiple sub-tasks, such as syntactic and semantic pre-processing, slot filling, and anaphora resolution (Cardie, 1997).

One should however be aware that IE is a more limited task than ‘full text understanding’. In full text understanding, the goal is to represent in an explicit fashion all the information in a text. In contrast, in information extraction the semantic range of the output (the relations to represent, and the allowable fillers in each slot of a relation) is defined in advance, as part of the specification of the task (Grishman, 1997).

In short, the “goal of a Information Extraction system [...] is to extract specific types of information from text” and “the main advantage of IE task is that portions of a text that are not relevant to the domain can be ignored”. (Vargas-Vera, Motta, Domingue, Buckingham & Lanzoni, 2000).

2.2.2 Information extraction versus Data extraction

There is a need of bringing forward the difference between the terms “data extraction” and “information extraction” in our report because in the current literature, there are used by some authors as synonyms and by some other authors as distinct terms.

First of all, let us define what information is. Information is defined as “any kind of knowledge that derived from study, experience or instruction. This knowledge is exchangeable amongst people, about things, facts, concepts, etc., in some context (McSweeney, 2009). The Business Dictionary (2009) defines information as “raw
data that is presented within a context that gives it meaning and relevance”. Two concepts (data and information) that are usually interchanged are in perspective in the previous definition.

Data is defined as “an information in numerical form that can be digitally transmitted or processed” (MerriamWebster, 2009). The difference between information and data lays in the fact that “data must be interpreted by a human or machine, to derive meaning” (McSweeney, 2009) and therefore becomes an information. The schema below illustrates the main difference between data and information.

![Figure 4: Difference between data and information](image)

From the definitions above, we can infer that data extraction is a subpart of information extraction. In fact, information extraction is the extraction of data plus interpreting (i.e. giving meaning to) the data extracted. In our work, we do not intend to simply extract data, rather extract data in a way so that they will be meaningful for us. Therefore, the terms ‘data extraction’ and ‘information extraction’ will be used interchangeably to refer to the same concept of identifying the text fragments that satisfy an information need.

### 2.2.3 Information extraction approaches

“Information Extraction approaches [...] represent a group of techniques for extracting information from documents, ultimately delivering a semantic ‘meaning’ representation of it” (Engels & Bremdal, 2000). According to Jackson, Al-Kofahi, Tyrrell and Vachher (2003), “approaches to the problem of information extraction can be differentiated from each other along a number of dimensions, some theoretical, some practical”.

“On the practical side, one can distinguish between those systems that require human intervention at run time, and those that require little or no intervention. Full automation is not always necessary in order to produce a useful tool, and may be undesirable in tasks requiring human judgment” (Jackson, Al-Kofahi, Tyrrell & Vachher, 2003).
On the theoretical side, Appelt and Israel (1999) suggest that there are two basic approaches which are the knowledge engineering approach and the automatic training approach. In the following sections, we present in details the different characteristics of each approach.

2.2.3.1 Knowledge Engineering Approach

The main characteristic of this approach is that a ‘knowledge engineer’ develops the grammars used by a component of the IE system (Appelt & Israel, 1999). The knowledge engineer is someone familiar with the IE system and the formalism for expressing rules for that system. His role is to write rules (for the IE system component) that mark or extract the sought-after information. This task (writing the rules) can be done by the knowledge engineer either on his own or in consultation with an expert in the domain of application. Appelt and Israel (1999) argue that a distinctive trait of this approach is that the knowledge engineer has access to a moderate-size corpus of domain-relevant texts (i.e. all that a person could reasonably be expected to personally examine) and his (or her) own intuitions. Thus, the skill of the knowledge engineer plays a large role in the level of performance that will be achieved by the overall system.

Another important aspect of this approach is that “building a high performance system is usually an iterative process whereby a set of rules is written, the system is run over a training corpus of texts, and the output is examined to see where the rules under- and over generate. The knowledge engineer then makes appropriate modifications to the rules, and iterates the process” (Appelt & Israel, 1999).

Regarding the advantages of this approach, one could mention that “with skill and experience, good performing systems are not conceptually hard to develop” and “the best performing systems have been hand crafted” (Appelt & Israel, 1999). But there are also some drawbacks of using this way of building IE systems. Firstly, it a very laborious development process (it requires a demanding test-and-debug cycle); secondly, some changes to specifications can be hard to accommodate and thirdly the required expertise may not be available.

2.2.3.2 Automatic Training Approach

Unlike the previous one, when using this approach, it is not compulsory “to have someone on hand with detailed knowledge of how the IE system works, or how to write rules for it. It is necessary only to have someone who knows enough about the domain and the task”. According to Appelt and Israel (1999), this person’s role is “to take a corpus of texts, and annotate the texts appropriately for the information being extracted”. The same authors call “training corpus, the result of the annotation of a corpus. Once a suitable training corpus has been annotated, a training algorithm is derived from it, resulting in information that a system can employ in analyzing novel texts. Another way to obtain training data consists in interacting with the user during the processing of a text. The user will be needed to support the system by confirming or invalidating the hypotheses on a text that will be returned back by the system. If the
hypotheses about the given text are not correct, the system modifies its own rules to accommodate the new information.

One of the strengths of this approach is that the automatic training approach focuses on producing training data rather than producing rules. As a result, system expertise is not required for customization, because as said earlier, as long as someone familiar with the domain is available to annotate texts, IE systems can be built for a specific domain. Another advantage of this approach is that ‘data driven’ rule acquisition ensures full coverage of examples. This stems from the fact that the user, first, annotates the text and by doing so, gives all the examples of the sought-after information to the system.

The disadvantages of the automatic training approach also revolve around the fact that it is based on training data. As stated by Appelt and Israel (1999), “Training data may be in short supply, or difficult and expensive to obtain”. Another drawback is that large volume of training data may be required. Finally, it may happen that changes to specification require reannotation of large quantities of training data.

2.2.3.3 When to use what?

As mentioned before, the two approaches for IE are quite different: one is a ‘rule-based’ approach while the other one is a ‘data-driven’ approach. Accordingly, one question that raises naturally is which approach to use to solve a specific problem. Appelt and Israel (1999) have provided some hints that are summarized in the table below:

<table>
<thead>
<tr>
<th>Use Knowledge Engineering approach when</th>
<th>Use Automatic Training Approach when</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Resources (e.g. lexicons, lists) are available</td>
<td>• Resources are unavailable</td>
</tr>
<tr>
<td>• Rule writers are available</td>
<td>• No skilled rule writers are available</td>
</tr>
<tr>
<td>• Training data is scarce or expensive to obtain</td>
<td>• Training data is cheap and plentiful</td>
</tr>
<tr>
<td>• Extraction specifications are likely to change</td>
<td>• Extraction specifications are stable</td>
</tr>
<tr>
<td>• Highest possible performance is critical</td>
<td>• Good performance is adequate for the task</td>
</tr>
</tbody>
</table>

Table 1: Use of Knowledge Engineering Approach vs Automatic Training Approach (Appelt & Israel, 1999)
2.3 Query construction

2.3.1 Definition of a query

A query is defined as “the formulation of a user need” (Baeza-Yates & Ribeiro-Neto, 1999, p100). Webopedia notifies that there are three general methods for presenting a query:

- Choosing parameters from a menu: in this method, the database system presents a list of parameters from which the user can choose according to his needs.
- Query By Example (QBE): here the system presents a blank record (a kind of form) and lets the user specify the fields and values that define the query.
- Query language: is defined as a programming language for formulating queries for a given data format. Query languages exist for both databases and information systems. This is the most complex method because it forces the user to learn a specialized language, but it is also the most powerful.

A query language is usually composed of keywords and the documents (usually stored into a database or information system) containing such keywords are searched for. Thus, the query language can be a simple word or it can be “a more complex combination of operations involving several words” (Baeza-Yates & Ribeiro-Neto, 1999, p100). The same authors classified the different types of keyword queries that exist:

- Single word query: elementary query that can be formulated in a text retrieval system.
- Context query: complement single word queries with the ability to search words in a given context. We can distinguish phrase query (sequence of single-word queries) and proximity query (the sequence of single words is given with a maximum distance between them).
- Boolean query: composed of basic queries that retrieve documents and of Boolean operators (OR, AND, BUT) which work on the sets of documents.
- Natural Language query: where the query becomes a simple list of words and context queries. All the documents matching a portion of the user query are retrieved.

2.3.2 Query construction

Query construction (or query formulation) consists in translating the user’s information need into a query. After the query has been built, it needs to be tested against one or several data sources and give back results to the user. Therefore, query formulation needs to be done according to both the user’s information need and the type of systems where the information are stored in. Van Der Pol (2003) suggests also that formulating a query requires the knowledge of the subject area (or domain knowledge). In fact, the domain knowledge is important because the information need “comprises concepts having several features” and also “several concepts having identical features” (Van Der Pol, 2003). Thus, by having the domain knowledge, one can either enlarge the domain of search (this process is referred to as query expansion)
or delete redundancies in the information need (as the result of the semantic integration between different terms).

One of the ways to represent this domain knowledge is using an ontology. Gladun, Rogushina, and Shtonda (2006) define ontology as “a knowledge represented on the basis of conceptualization that intends a description of object and concept sets and relations between them. Formally, ontology consists of the terms organized in taxonomy, their definitions and attributes, and also connected with them axioms and rules of inference”. According to the same authors, “ontology is a semantic basis in a content description” and “can be applied for communications between the people and software agents”.

Several authors have already suggested and tested the use of ontology in the process of query formulation. After making a survey of approximately 30 important publications on ontology-based search/query systems, Hoang and Tjoa (2006) concluded that ontologies “are very crucial and play a key role” in such systems. In fact, they “appear from the starting (query formulation) until the end (query answering) of querying processes”. They can be used to:

- Provide a pre-defined set of terms for exchanging information between users and systems.
- Provide knowledge for systems to infer information which is relevant to user’s requests.
- Filter and classify information.
- Index information gathered and classified for presentation.

The first two roles presented above are more query formulation oriented whereas the last two have an orientation towards query processing. In this report, we will focus only on how ontology can help in query formulation.

As mentioned earlier, the domain knowledge can be used in query expansion as well as in semantic integration. The next two sections contain a presentation of the contribution of ontology to both concepts.

### 2.3.2.1 Ontology and semantic integration

Tierney and Jackson (2003) have suggested the use of ontology mapping in conceptual semantic integration. These authors argue that “[…] conceptual semantic integration approach allows for two distinct ontologies to be merged/integrated using a set of semantically similar words.”

Ontology mapping is the fact of mapping two ontologies into one. “Given two ontologies O₁ and O₂, mapping one ontology onto another means that for each entity (concept C, relation R, or instance I) in ontology O₁, we try to find a corresponding entity, which has the same intended meaning, in ontology O₂” (Ehrig & Staab, 2004).
Theoretical Background

The following example illustrates a mapping. Two ontologies $O_1$ and $O_2$ describing the domain of car retailing are given (figure 4). A reasonable mapping between the two ontologies is given in table 2 as well as by the dashed lines in the figure.

![Ontologies and mappings](image)

Figure 5: Example of ontologies and their mappings (Ehrig & Staab, 2004)

<table>
<thead>
<tr>
<th><strong>Ontology $O_1$</strong></th>
<th><strong>Ontology $O_2$</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>Thing</td>
</tr>
<tr>
<td>Car</td>
<td>Automobile</td>
</tr>
<tr>
<td>Porsche KA-123</td>
<td>Marc’s Porsche</td>
</tr>
<tr>
<td>Speed</td>
<td>Characteristic</td>
</tr>
<tr>
<td>250 km/h</td>
<td>Fast</td>
</tr>
</tbody>
</table>

Table 2: Mapping Table for the two ontologies $O_1$ and $O_2$ (Ehrig & Staab, 2004)

Noy (2004) has also suggested the use of ontology mapping for semantic integration. She views ontology mapping as having many dimensions. One of them is the mapping discovery between ontologies which is how to find similarities between two given ontologies. Methods to achieve mapping discovery is whether to use a shared ontology (“If two ontologies extend the same reference ontology in a consistent way, then finding correspondences between their concepts is easier” (Noy, 2004)) or to use Heuristics and Machine-learning (which “compares the number and the ratio of shared words in the definitions to find definitions that are similar” (Noy, 2004)).
2.3.2.2 Ontology and query expansion

“Query expansion is needed due to the ambiguity of natural language and also the difficulty in using a single term to represent an information concept. […] The main aim of query expansion (also known as query augmentation) is to add new meaningful terms to the initial query” (Bhogal et al, 2006).

Ontology helps to achieve this goal by providing suggestion of terms that are linked to the initial query of the user. Let us consider again the example of the domain of car retailing used in the previous section. A user might want to search for the term ‘car’. Because “ontologies provide consistent vocabularies and world representations necessary for clear communication within knowledge domains” (Leroy, Tolle & Chen, 1999), this initial query could be enriched by a synonym of ‘car’ which is ‘automobile’. Processing the search with these two terms instead of one may lead to a higher recall, since some documents may not contain ‘car’ but ‘automobile’, both terms referring to the same concept in this domain. As a result, ontologies improve the accuracy of the information search by paraphrasing the query of the user through context identification and disambiguation. (Leger, Lethola & Villagra, 2001).
2.4 Information Retrieval

In the previous section, we discussed about the different concepts that help to build a query. Once the query is built, it is sent into an information system or database in order to find relevant documents (to the query) and give back the results to the user. This is called Information Retrieval. Before going deeper in the concept, let us define what Information Retrieval is.

2.4.1 Definition

Manning, Raghavan and Schütze (2009) state that “information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).” The term ‘unstructured document’ also known as ‘unstructured data’ refers to raw data, which is a data type that “has no identifiable structure. Unstructured data typically includes bitmap images/objects, text and other data types that are not part of a database” (Dorion, 2007). Some other examples of unstructured data are “web pages, office documents, presentation and emails…” (Science for SEO, 2009). Bouton and Hammersley (1996) assert that one should not be misled by the term ‘unstructured’. According to these authors, ‘unstructured’ does not mean that the data lacks all structure (because all data are structured in some way), rather the structure is not helpful for the desired task (or final user).

Manning et al. (2009) argue also that “IR can also cover other kinds of data and information problems beyond that specified in the core definition above”. For instance, IR covers also ‘structured data’. Structured data is “any set of data values conforming to a common schema or type” (Arasu & Garcia-Molina, 2002). This type of data is “organized in a structure so that it is identifiable. The most universal form of structured data is a database like SQL Server or Access” (Dorion, 2007). Besides unstructured and structured data, “IR is also used to facilitate ‘semi structured’ search such as finding a document where the title contains Java and the body contains threading.” (Manning et al., 2009). Semi-structured data is a type of data that is neither completely raw, nor strictly typed (Abiteboul, 1996). It is a type of data which is halfway between the previous two types. Semi-structured data is encountered in several applications. Some examples are on-line documents such as HTML, Latex, BibTex, SGML files (Wang & Liu, 2000).

In order to retrieve information that satisfies user’s need, an IR system is seen as composed of three parts: an input, a processor and an output (Van Rijsbergen, 1979). The input consists of the query of the user (which specifies the information need of the latter) and the document collection (which is the source where this information could be found). The processor is the part of the IR system that is concerned with the retrieval process. It is made of different algorithms that execute the search strategy in response to a query. The output is the result of the search processing. It is “usually a set of citations or document numbers” (Van Rijsbergen, 1979, chap1).

When the retrieval system is on-line, it is possible for the user to change his request during one search session in the light of sample retrieval, thereby, it is hoped, improving the subsequent retrieval run. This procedure is commonly referred to as feedback. All these steps are summarized in the diagram below:
2.4.2 Information Retrieval versus Information Extraction

Information retrieval should not be confused with Information extraction. These two concepts differ in their aims and also in the techniques they use.

According to Pudota, Casoto, Dattolo, Omero and Tasso (2008), “IR aims at retrieving all and only the documents storing information relevant to the user’s information needs, while IE aims at extracting text which matches a template; either it is manual or automatic way, the goal of IE is to search for words, paragraphs, or text snippets contains searched information matched to the specified template and present it in a more organized and structured form. This means that the central notion of IR is relevance, while that of IE is information structure. The former is represented through a query (or, more generally, by some implicit or explicit input from the user), whereas the latter is represented by a template”.

Gaizauksas and Wilks (1998) argue that “the contrast between the aims of IE and IR systems can be summed up as: IR retrieves relevant documents from collections, IE extracts relevant information from documents. The two techniques are therefore complementary, and their use in combination has the potential to create powerful new tools in text processing.”

The table below summarizes in simple words the differences in the aims of Information Retrieval and Information Extraction.

<table>
<thead>
<tr>
<th>Information Retrieval (IR)</th>
<th>Information Extraction (IE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aims to select relevant documents, i.e., it finds documents.</td>
<td>Aims to extract relevant facts from the document, i.e., it extracts information.</td>
</tr>
<tr>
<td>Views text as a bag of unordered words.</td>
<td>Interested in structure and representation of the document.</td>
</tr>
</tbody>
</table>

Table 3: Differences between IR and IE (Khalid, 2008)
Information Extraction and Information Retrieval differ also in the techniques they use. The technical differences arise from their difference in aim, but also for historical reason. According to Gaizauksas and Wilks (1998), most work in IE has emerged from research into rule-based systems in computational linguistics and natural language processing therefore IE must pay attention to the structural or syntagmatic properties of texts by using some information theory, probability theory, and statistics. By contrast, IR systems treat texts as no more than ‘bags’ of unordered words.

Gate Information Extraction (2009) also mentioned the technical differences between IR and IE. It claims that “Information Extraction differs from traditional techniques in that it does not recover from a collection a subset of documents which are hopefully relevant to a query, based on key-word searching (perhaps augmented by a thesaurus). Instead, the goal is to extract from the documents (which may be in a variety of languages) salient facts about prespecified types of events, entities or relationships. These facts are then usually entered automatically into a database, which may then be used to analyze the data for trends, to give a natural language summary, or simply to serve for on-line access”.

From the figure below, we can see that “unlike information retrieval, which concerns how to identify relevant documents from a collection, information extraction produces structured data ready for post-processing.” (Chang, Hsu & Lui, 2003).

![Figure 7: Information Retrieval Vs Information Extraction (Gate Information Extraction)](image)

Figure 7: Information Retrieval Vs Information Extraction (Gate Information Extraction)
2.4.3 Information Retrieval and Document Preprocessing

As underlined by Ziviani (1999), “not all words are equally significant for representing the semantics of a document. [...] Therefore it is usually worthwhile to preprocess the text of the documents in the collection to determine the terms to be used as index terms (or keywords)” An index term is defined as “a pre-selected term which can be used to refer to the content of a document” (Baeza-Yates & Ribeiro-Neto, 1999).

Document preprocessing aims at reducing the number of index terms for a document. In fact, document preprocessing is needed because “using the set of all words in a collection to index its documents generates too much noise for the retrieval task” (Ziviani, 1999). Thus, it is expected that the preprocessing of the documents leads to an improvement of the retrieval performance.

Document preprocessing is a procedure that can be divided mainly into five text transformations:

- Lexical analysis of the text,
- Elimination of stopwords,
- Stemming of words,
- Index terms selection,
- Thesauri.

Here we present the main characteristics of each operation.

2.4.3.1 Lexical analysis of the text

“One of the major objectives of the lexical analysis phase is the identification of the words in the text” (Ziviani, 1999). Ziviani (1999) suggests that in order to perform lexical analysis, one can:

- Reduce multiple space to one space,
- Remove all words containing sequences of numbers unless specified otherwise,
- Remove hyphens (which means that ‘state-of-the-art’ and ‘state of the art’ would be treated as the same),
- Put all letters either in upper case or lower case,
- Remove every punctuation mark.

2.4.3.2 Elimination of stopwords

Stopwords are extremely common words which would appear to be of little value in helping select documents matching a user need (Manning et al., 2009). These words “have little to do with the information being sought by the searcher” (Slawski, 2008). Natural candidates for a list of stopwords are articles, prepositions and conjunctions (Ziviani, 1999).
2.4.3.3 **Stemming of words**

Stemming consists in the substitution of words with their respective stems. “A stem is the portion of a word which is left after the removal of its affixes (i.e., prefixes and suffixes)” (Ziviani, 1999). For instance, the variants ‘connected’, ‘connecting’, ‘connection’, and ‘connections’ have the same stem which is ‘connect’. The most popular stemming (suffix removal) algorithm is the Porter algorithm.

2.4.3.4 **Index terms selection**

Instead of representing a text with all the words it contains, an alternative is to select some words that will be used as index terms for the document. Ziviani (1999) argues that “most of the semantics is carried by the noun words”. Hence, a strategy for selecting index terms is to eliminate systematically all verbs, adjectives, adverbs, connectives and pronouns (Ziviani, 1999). Another strategy for index term selection is to combine two or three nouns in a single component. With this strategy, ‘computer science’ will be treated as one term instead of two.

2.4.3.5 **Thesauri**

Basically, a thesaurus (singular of thesauri) “consists of (1) a precompiled list of important words in a given domain of knowledge and (2) for each word in this list, a set of related words. Related words are, in its most common variation, derived from a synonymity relationship” (Ziviani, 1999). Foskett (as cited in Ziviani, 1999) states that the main purposes of a thesaurus are essentially: (a) to provide a standard vocabulary (or system of references) for indexing and searching; (b) to assist users with locating terms for proper query formulation; and (c) to provide classified hierarchies that allow the broadening and narrowing of the current query request according to the needs of the user.
2.5 User interface

In the previous sections, we have seen how a user could satisfy his information need first by building a query, second sending this query into an information system or database that would retrieve relevant documents and send them back to the user later on. This interaction between the user and the system follows an Information access process mentioned by Baeza-Yates and Ribeiro-Neto (1999) and used by Web search engines.

![Diagram of the standard model of the information access processes](image)

Figure 8: A simplified diagram of the standard model of the information access processes (Baeza-Yates & Ribeiro-Neto, 1999, p263)

A user interface is the program that allows the user to follow the above information access process and therefore to interact with the system (which could be a computer, a database, an information System …).

2.5.1 Definition

User Interface (UI) is defined as “a program that controls a display for the user (usually on a computer monitor) and that allows the user to interact with the system” (The Free On-line Dictionary of Computing (as cited on Die.net, 2003). A user interface can be either graphical or command line (also called text-and-keyboard
oriented). The first interactive user interfaces were text-and-keyboard oriented and usually consisted of commands the user had to remember and the computer responses were infamously brief. DOS is an example of such UI (SearchWinDevelopment, 2006). After the text-and-keyboard oriented UI, the Graphical User Interface (GUI) was created. The GUI “emphasizes the use of pictures for output and a pointing device such as a mouse for input and control” (Die.net, 2003). The elements of a GUI include such things as: windows, pull-down menus, buttons, scroll bars, iconic images, wizards, the mouse … Applications usually use the elements of the GUI that come with the operating system and add their own graphical user interface elements and ideas (SearchWinDevelopment, 2006).

2.5.2 A typical User Interface

Nowadays many object-oriented tools exist that facilitate writing a graphical user interface and forging its design. According to the Usernomics, many technical innovations rely upon User Interface Design to raise their technical complexity to a usable product (Usernomics, 2008). A “good User Interface Design can make a product easy to understand and use, which results in greater user acceptance.” (Usernomics, 2008).

The user interface, in order to be accepted by the user and complete the user’s need should fulfill some requirements. Baeza-Yates and Ribeiro-Neto (1999) argue that:

- The User Interface should allow the user to reassess his goals and adjust his strategy accordingly: it will be useful when the user encounters a ‘trigger’ that causes him to use a different strategy temporarily.
- The UI should support search strategies by making it easy to follow tracks with unanticipated results. This can be achieved in part by supplying ways to record the progress of the current strategy and to store, find, and reload intermediate results.
- An important aspect of the interaction between the user and the UI is that the output of one action should be easily used as the input to the next.
Methods

In this chapter, we present the overall strategy to solve our research problems. We have chosen a flexible research design because our work aims at discussing various ways for extracting the request from the models and for expressing the request in a query. In other words, we intended to try different methods or processes to perform both extraction and retrieval and see the results. Robson (2002) suggests that if the focus “is on processes, a flexible design is probably indicated”.

We have used the System Development Method (SDM). Indeed, our work within the field of IS combines both research and practice. The combination is done in such a way that, on one hand, the research raises a broad understanding of the problem and on the other hand, the development of a program serves as a proof of testing the theories. SDM is the research design methodology which is the most appropriate for this kind of work.

SDM is a disciplined investigation which is specific to the field of IS. In our case, the focus on the SDM was necessary because we intended to illustrate how Enterprise Models can be applied as interface for Information Searching both in theoretical and technical perspectives. Following this approach, we had to use three main steps:

The first step is the concept building where we had to construct our research question. Accordingly, we reviewed the literature to get related knowledge about the research and effort that have been done so far by scholars, in this field. It implied that we searched, found, and synthesized the existing knowledge in order to identify the scope of our work. This step resulted in an overview of the existing literature that was presented above in the section “Theoretical background”.

Based on the knowledge gained gradually, we started the system building step which consisted in the development of a software program based on the different approaches found in the literature. We started defining the main functionality of the system and the way to reach it. For that, we divided the whole system in different modules each of them with a function. We also found a way to make all the modules interact with each other so that it would produce the main requirement of the whole system. Many drafts of algorithms have been created and used to support the building of the modules and some sensitive parts of the software.

At the same time, we also thought about the design of the system. We intended to make the user interface easy to understand (that way we would increase its acceptance by the user) no matter how complex the system could be. We inspired ourselves from the simplicity of many search engines that already exist and try to create a user interface as complete (and simple) as possible.

Then we start building an evolutionary prototype using a development language. Prototyping was an iterative process where we had to develop, go through the literature again and/or ask for confirmation from the expert in the field and go back to the development. The more we got knowledge from the literature and from the expert in the field, the clearer we could draw and follow a consistent framework. Many prototypes following different frameworks have been built.

We formalized the system building step by including in our report a detailed requirement analysis (see appendix1).
The last step consisted in the *system evaluation* where we came up with some prototypes. We discussed the different frameworks or approaches used and their effects on the resulted prototypes.

Concerning the technical aspect of the development of the prototypes, we have chosen to develop the applications in Java programming language. Java was chosen because it is object-oriented and platform-independent. Java has also the advantage of allowing for portability to any machine, provided that this machine possesses a virtual machine. The version of Java used was 1.6.0_13.

Regarding the Integrated Development Environment (IDE), we have preferred to use Netbeans because it is an open source tool with a good support. In addition, this IDE helps to gain a lot of development time through its intuitive interface and its suggestions during the coding process. Finally, Netbeans was chosen because it provides also facilities for the development of the graphical user interface, a feature that was very important for us during our work. The version of Netbeans used was 6.5.1 and the software was developed in Windows XP.
4 Implementation

As mentioned earlier in this report, we divided the system to build in different modules (figure 1), each of them having specific functions. Three (3) modules were mainly defined: the Enterprise Model Analyzer (or EM Analyzer), the Query Generator (query constructor, search engines and database) and the User Interface Module (or UI Module).

The EM Analyzer aims at extracting desired information from the XML file. The desired information are the competencies that are linked to a role in an organization.

The Query Generator contains the query constructor, the search engines and the database (figure 1). Its role is to build queries based on the competencies extracted previously, test these queries using a search engine or a database and give results back. The returned results are people that fit the best to the competencies extracted beforehand.

The last module is the User Interface Module. It was developed to support the previous two modules. Its role is to manage all the interactions with the user from specifying his information need to reading the final results.

A case, provided by the supervisor in order to test the software is shown in appendix6 and described below.

4.1 Description of the case

The model was built using Troux, and is based on the GEM meta-model. It contains different objects: process, person and initiative. All these object types have an attribute "name" which contains the strings to be used in searching. Each object type and its relationship with the others is described below.

4.1.1 The object type process

The object type process contains three processes: ‘Write application’, ‘perform project’ and ‘publish results’. The processes are linked to each other through the relationship 'follows' or 'followed by'. Each process is also linked to the object person (described below) through the relationship 'has responsible' (process has responsible person).

4.1.2 The object type person (as substitute for role)

The object type person contains ‘Ontology expert’, ‘Network expert’, ‘Fractal System Expert’, ‘Context Expert’. The persons are not directly related to each other. However they are related to the object types process and competences through the relationship ‘responsible for’ (person responsible for competence).

4.1.3 The object type initiative (as substitute for competences)

The object type initiative contains the different objects ‘fractalural structure’, ‘rule engine’, ‘context’, ‘Web service’, ‘pattern’, ‘ontology’, ‘awareness’ and ‘network’. Those objects are not related to each other. The objects initiative are related to the
objects person through the relationship ‘has responsible’ (competence has responsible person).

4.2 Module 1: Enterprise Model Analyzer

4.2.1 Concept building

From the literature review, we found that there are two approaches of information extraction: the Knowledge Engineering Approach and the Automatic Training Approach. We have chosen to implement the Knowledge Engineering Approach because first of all, the required expertise was available. This expertise has been provided by our supervisor during the whole duration of our work. We have played the role of the knowledge engineer and were responsible for writing the rules for the information extraction. Finally, we have chosen to build a system with little human intervention since our aim was twofold: automate as much as possible the process of information extraction and at the same time give some flexibility to the user in the expression of his information need.

4.2.2 System building

In order to build this module, we have followed an iterative process which consists of four (4) steps. The first step has consisted in a discussion with the domain expert which has helped to define precisely the information to extract. Using this précised definition of the information, we were able to generate algorithms for the extraction and later on, generate the corresponding code in Java language. Finally, we went back to the domain expert to get his feedback and make refinements when appropriate. These steps are summarized in the diagram below:

![Diagram of the system building process]

Figure 9: Process for building an information extraction system

We proposed an algorithm rather simple to perform the extraction of the information. It is described in the picture below.
Figure 10: An algorithm for information extraction

As mentioned earlier, a XML file is a semi-structured data. Thus, while designing this algorithm, we made sure to query both the syntax and the semantics of this document to take advantage of the richness of XML. In the while loop in particular, both operations are performed (syntax and semantics). In the first place, we extract some information according to the structure (or syntax) of the document by the function FIND_LINKED_OBJECTS. Once these ‘linked objects’ are extracted, we check their meaning (or semantics). This is done by getting the type of the extracted objects (function TYPEOF).

The interpretation (of the extracted data) we used was also very simple and intuitive. We considered that every object contained in a container is of type ‘container’. For instance, an object which is part of the container ‘Roles’ is a ‘role’, and object which is part of the container ‘Process’ is an ‘activity’ and so on. It might happen that a container is composed of several containers (that we name here ‘subcontainers’). In that case, an object of type ‘subcontainer’ is automatically an object of type
‘container’, but an object of type ‘container’ is not considered as an object of type ‘subcontainer’. For example, if a container ‘Organization’ is composed of ‘Professor’ and ‘Student’, a ‘professor’ is automatically a ‘member of organization’ but a ‘member of organization’ might not be a ‘professor’; it might have another role (e.g. ‘student’).

4.3 Module2: Query Generator

4.3.1 Concept building

From the literature review, we found that different types of query exist. In our case, based on the given model and the different search engines and database, we have identified three types of query that could be used: single word query, context query (especially phrase query) and Boolean query.

Many concepts or methods were also provided by scholars for the information retrieval task. Some of them are identification of the language used, lexical analysis, elimination of stop words, use of thesaurus. According to the case we had, all of them were not useful. We needed only the identification of the language and the use of the thesaurus for query expansion.

4.3.2 System building

After the extraction of competences, the intention of the user is to find the ‘best candidate’ for his role based on the extracted competences. Since there could be lots of ways of defining the term ‘best candidate’, our first task was to clarify what we mean by these terms.

We started by defining the ‘best candidate’ as the ‘candidate’ that has the highest ‘relevance’. This definition has introduced two new terms that we need to have a closer look on: ‘candidate’ and ‘relevance’. The relevance and the way it is assigned to a ‘candidate’ depends on the search engine used to perform the information retrieval task. For this reason, this term will be presented in the section 4.5 about the search engines and the database. But regarding ‘candidate’, we have considered four possible ways of defining (from a user point of view) a ‘candidate’ for a role which requires a set of competences.

- A ‘candidate’ for the role is someone that has ALL the competences.
- A ‘candidate’ for the role is someone that has SOME of the competences.

Under the assumption that the extraction of competences has given n competences for a role (n>=2), this consideration could also be made:

- A ‘candidate’ for the role is someone that has at least m competences among n (m<n).
- Finally, another way of defining a ‘candidate’ is to let the user himself express his thought. This means that each competence will be assigned a ‘degree of importance’. We have considered three degrees of importance:
  - Critical: that means that a ‘candidate’ MUST have this competence.
After defining the terms to remove ambiguity in the interpretation of the results, we moved to the second task which is linked to the construction of the query itself. The way the query is built is strongly dependent on whether we use online search engines (DBLP, Google) or a relational database.

4.3.2.1 Search engines (DBLP, Google)

The input of these two search engines is the competences extracted from the XML file. It is possible also that the user chooses the Roles or Persons as input. We did not have any restriction regarding this point because we wanted to make this software adaptable to different Troux Enterprise Models. However to follow the main idea of the work thesis and for the software to be meaningful, it is better to choose Competences as type of results.

From the analysis of the XML file we made, we realized that those competences are expressed in natural language and the language used is English. They can be either single terms (for instance context or ontology) or compounds terms (for instance rule engine or fractal structure). According to the domain expert, the user should be able to use:

- one of the competences as input for the search engines or database: in this case we talk about the single word query (if the competence is expressed using one single term) or phrase query (if the competence is described using a compound term).
- More than one competence as input for the search engines or database: in this case we need to transform the input into Boolean query using the Boolean operators such as AND and OR. The Boolean query will therefore be the concatenation of each of the competences chosen by the user.

If there is only one competence at a time, this competence is just used a keyword for the search engines. But if the user wants to look for people that have several competences at the same time, we have used two different methods to approach the problem. These methods differs in their way of using the Boolean operators during the construction and execution of the query. We will take here a small example because it will help to better illustrate the steps of each of the method.

Example: Let us assume that the user wants to look for people that have the competences ‘context’ AND ‘network’.

The first method consists in linking the different competences with Boolean operators and then use this new collection of terms as input for the search engines. When we apply this method to the example above, we obtain a phrase query which is ‘context AND network’ and this query is, later on, sent to the search engines. Here is a figure to illustrate the different steps of this method for a collection of n competences. In this figure, a competence is referred to as a ‘query term’.
The second method consists in retrieving candidates for each competence independently and then, looking for candidates that appear in both lists of results. This means that we will retrieve all the candidates for ‘context’, store the results in a list (called here list1) and make the same operation for ‘network’ to get a second list (called here list2). We will then make the intersection (because of the Boolean Operator AND) of both lists (list1 and list2) and return the resulting list to the user. The figure below illustrates the different steps of this method, for a collection of n competences(or ‘query term’).

After the query is built it is sent directly as keywords on the webpage of Google and DBLP. The way the query are processed in those search engines is briefly introduced in the section 4.5.
4.3.2.2 Database

Just like the search engines, the input of the database is the set of competences extracted from the XML file using the Enterprise Model Analyzer. We have built a database which contains profiles of people. MS Access 2003 was used because it was easy to manipulate and did not required any additional cost (in fact, as long as one has a Microsoft Office Professional Suite, it is included in it). The structure of the database is briefly introduced in the section 4.5. In our case, a profile of a person is made of three attributes:

- his competences: this attribute describes his different skills and the number of years he has practiced these skills.
- his education: his academic background from primary school to university.
- his work experience: this attribute describes his current position, the companies he has worked in, the position he had in each of the company and the number of years he has occupied this position.

In order to have access to these profiles, SQL was used because it is now a standard language for accessing and manipulating relational databases like MS Access. We were only interested in building ‘SELECT Queries’ because our aim was not to make modifications on the database (update, delete, etc) but simply to get the information that were stored in it beforehand.

Here again, we needed to distinguish between the case where we have only one competence at a time and the case where we want to look for people who have several competences at a time.

If we have only one competence at a time, this competence is used to put a constraint on the attribute competences of the people stored in the database. The generic form of the SQL Query is consequently (the single competence is referred to as ‘keyword’):

```
1 SELECT person_attributes // the desired attributes of the person.
2 FROM database_tables // the tables in the database.
3 WHERE person.competence = 'keyword' // the attribute competence of the person should be
4 equal to the single competence given as input.
```

Legend: // In blue: comments

Figure 13: Generic SQL Query with a single competence as input

But if we have many competences at the same time, two methods were used which are quite similar to the methods presented earlier for the query construction and execution in Google and DBLP. The first method consists in putting the Boolean operators inside the SQL query and run this query later on. The generic form of the SQL query is as displayed below in figure 14; the different steps of this method are shown in figure 15. For the second method, we need to build a set of SQL queries (each with a single competence as input), run these queries and intersect or merge the intermediate results. The steps of this method can be seen in figure 16 below.
Implementation

1. SELECT person_attributes // the desired attributes of the person.
2. FROM database_tables // the tables in the database.
3. WHERE person.competence = 'keyword1' AND/OR person.competence = 'keyword2' AND/OR...
4. ........ AND/OR person.competence = 'keywordn' // n constraints are put on the attribute
5. competence of the person, using the n keywords that are given as input

Legend: // In blue: comments

Figure 14: Generic SQL Query with multiple competences as input

Figure 15: Query construction for the MS Access Database (method1)

Figure 16: Query construction for the MS Access Database (method2)
4.3.2.3 Query expansion

After building the queries (for search engines, we get a phrase query, for the database we get a SQL query), we have considered enlarging the domain of the search to increase the recall of our queries. At this stage we have two possible approaches from the literature: the use of ontology mapping or the use of a thesaurus.

The use of ontology mapping would have required either building a very big ontology to take into account all the concepts (like in figure 5) or building an ontology for each single competence and then mapping the term-ontologies to find similarities between terms. The advantage of this approach is a higher precision in the description of the terms, but its drawback is that it requires lots of resources for implementation and consequently is time-consuming. For this reason, we have preferred the use of a thesaurus which was on the one hand simpler, and on the other hand enough to illustrate the contribution of query expansion for query construction.

We have build a thesaurus using WordNet. It is a large lexical database of English language, which groups words into sets of synonyms called synsets. These sets of synonyms are then interlinked by means of conceptual-semantic and lexical relations. As a result, Wordnet provides a network of meaningfully related words and concepts.

Using this lexical database, we have built a thesaurus which contains the different competences provided in the case and some of their synonyms. The result is shown in the table below:

<table>
<thead>
<tr>
<th>Competences</th>
<th>Synonyms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Awareness</td>
<td>Knowingness, Consciousness</td>
</tr>
<tr>
<td>Context</td>
<td>Circumstance, Setting</td>
</tr>
<tr>
<td>Fractal structure</td>
<td>Geometric pattern, irregular structure</td>
</tr>
<tr>
<td>Network</td>
<td>Web, Connections, protocols, communication</td>
</tr>
<tr>
<td>Ontology</td>
<td>Conceptualization, Description</td>
</tr>
<tr>
<td>Pattern</td>
<td>Design, guide, blueprint</td>
</tr>
<tr>
<td>Rule engine</td>
<td>Software</td>
</tr>
<tr>
<td>Web services</td>
<td>Distributed applications, connected services</td>
</tr>
</tbody>
</table>

Table 4: Thesaurus used in query expansion

---

2 There were no synonym for this term when we used WordNet. We made this assumption because a rule engine is "a piece of software that can execute a business rule that is defined within it" (Chisholm & Ross, 2007)
Implementation

- How the table is used

Let us take the example of a user looking for a person with the competences ‘network’ in the database. He selects ‘network’ and start the database search. If a person is registered in the database as having this competence but literary written as ‘protocols’ the database will not be able to return his profile as a result to the user. However if we use our table, the software will search for the synonyms of network; here there are web, connections, protocols, communication. Then, the software will search for all the persons who have those competences and return them back to the user. Thus, a person with the competence protocols will be returned.

Here is an algorithm we propose to perform query expansion. The role of this algorithm is to generate semantically equivalent queries based on the principle described in the previous paragraph.

```
1 new_query_terms ← ( )  // contains the answer to return.
2 query_terms = {qt1, qt2, ..., qtₙ}  // initial set of query terms.
3 thesaurus = CREATE THESAURUS  // create a thesaurus.
4 for each qt in query_terms  // go through all the query terms.
5   do
6       syn = FIND_SYNONYM (qt, thesaurus)  // find a synonym for the current query term
7       // using the thesaurus created before.
8       new_query_terms ← query_terms − {qt} + {syn}  // replace the current query term
9       // with its synonym.
10       RETURN(new_query_terms)  // return the new derived query to the system for processing
11 endfor
```

Legend: In upper case: functions
In lower case: variables
// In blue: comments

Figure 17: An algorithm for query expansion

4.4 Module 3: User Interface

As said earlier, the user interface (UI) was developed to manage all the interactions with the user, from specifying his information need to reading the final results. An important point to mention here is that we have developed the UI so that it fulfills the requirements laid down by Baeza-Yates and Ribeiro-Neto (1999). As a reminder these authors argue that in order to be accepted by the user and complete his need, a UI should:

- Allow the user to reassess his goals and adjust his strategy accordingly. In our case, this requirement has been translated into two rules that were omnipresent during the development phase. The first rule is that the user should always have the opportunity to go back and to reformulate his request. The second rule is that the user should be able to select as much input as he wants when specifying his request. An implication for this latter point is that
the user should be able to select as much roles as desired as input for the Enterprise Model Analyzer Module for example, rather than being restricted to only one choice possible at a time.

- **Support search strategies by making it easy to follow tracks with unanticipated results.** In order to fulfill this requirement, we have made two important design choices. The first one was to store the results of the Enterprise Model Analyzer in a file before going further to the construction of the query by the Query Generator. The second one is to provide a kind of ‘logs’ that traces all the big steps of the user from the moment when he starts specifying his request to the exit of the application. These ‘logs’ should be written in user friendly language (this language should have nothing to do with the technical aspects of the application) and accessible to the user from a part of the UI.

- The last suggestion of the Baeza-Yates and Ribeiro-Neto (1999) was that *the output of one action should be easily used as the input to the next.* This was done during our work by ensuring that the output of the Enterprise Model Analyzer is the input of the Query Generator and also ensuring that the output of the Query Generator is usable for any further development (Maintainability of the application).

### 4.5 Search Engines (Google, DBLP) and Database

As mentioned earlier, once the query is built, it is used as input in the one of search engine or the database. Then comes the concept of Information Retrieval. Information Retrieval deals with different types of data: structured, unstructured and semi-structured data. To illustrate the fitness of our software program we have decided to use two search engines (Google and DBLP) and a database to retrieve the three types of data (unstructured, semi-structured and structured).

#### 4.5.1 Search engines

##### 4.5.1.1 Google

Google is a search engine used for finding resources on the World Wide Web (or simply the ‘Web’). It uses text-matching techniques to retrieve information (stored on WebPages) that are relevant to the keywords entered in the search box. Once the keywords match, the engine ranks the retrieved WebPages in a list of hits by giving weight to the links that reference a specific page.

The description above means that if we enter our query in Google search box, Google will go through the WebPages retrieving all texts that match our keywords. Therefore the retrieved text does not have any identifiable structure or the structure it has is not helpful for the desired purpose (in our case person’s profiles). For instance, we can look for a person with the marketing competence, Google will retrieve informatics pages, online dictionaries, some articles ... containing “marketing competence” which
have nothing to deal with persons’ profiles. In this sense, we can say that Google retrieves unstructured data.

Google’s default behavior is to interconnect all the keywords with the Boolean operator AND. Regarding the relevance of the results, we have assumed that the hits that are ranked on the top of the result list are the most relevant. This assumption was made because of two reasons:

- In general, the retrieved text does not have any identifiable structure or it is not organized in a way so that we could deduce profiles of people from it. This makes it impossible to assign a relevance to a result.
- Google ranks the pages it considers as most relevant for a query on the top of the result list.

As it was impossible to assign a relevance to a single result, we needed to find another way to get an idea of the quality of the query built. What we did was to count the number of ‘links to people’s profiles’ (CVs or websites describing the profile of people) returned within the first five pages of results because:

- There are usually hundreds of pages of results and it is not possible to go through all of them.
- We made the assumption that an average user goes usually through the first 5 pages to look for the results of his query.

There exist several interfaces of Google (.se for Sweden, .de for Germany, .fr for France, etc) and the hits returned are different depending on the interface used. For this reason and because our competences were written in English, we decided to use www.google.co.uk to test our queries.

4.5.1.2 DBLP

DBLP is an acronym for ‘Digital Bibliography & Library Project’ and is a server that provides bibliographic information on computer science journals and proceedings. DBLP indexes more than 1.2 million of publications and contains more than 10000 links to home pages of computer scientists. Even if a publication is basically a collection of several sequences of written text, it has also different characteristics such as publication year, author of the publication, conference, publication type, etc. The latter attributes convey a little structure to the documents retrieved via DBLP and, in this way, make the whole collection of publications a semi-structured data source.

DBLP provides three interfaces for searching: the author search, the faceted search and the complete search. Each of these interfaces has distinct features. The author search aims at giving all the publications related to an author, given his name. The faceted search allows searching for publications starting from some keywords and shows the result set along with a set of facets (e.g., distinguishing publication years, authors, or conferences). The complete search is a mirror with extended search capabilities (e.g., prefix search, error-tolerant search, etc.).

Since we are looking for people that are the most appropriate for a competence, we have used the faceted search interface of DBLP because it was the only interface that could return authors as results for the keywords provided by the user. Another important feature of this interface is that it provides a ranking of authors according to the number of times they have authored or co-authored publications related to a topic. We have used this feature to define the relevance of our results. Three cases are to be distinguished:
• If one competence C is used as input for searching in DBLP, the relevance $R_{AC}$ of an author A for a topic C is defined as follows:

$$R_{AC} = number\ of\ times\ A\ has\ authored\ or\ co-authored\ publications\ linked\ to\ the\ topic\ C$$

(and this number is provided by DBLP as said earlier)

• If several competences \{C_1, C_2, \ldots, C_n\} are used as input for searching in DBLP, the relevance $R_{A\{C_1, C_2, \ldots, C_n\}}$ of an author A for the topics \{C_1, C_2, \ldots, C_n\} is defined as follows:

$$R_{A\{C_1, C_2, \ldots, C_n\}} = R_{AC_1} + R_{AC_2} + \ldots + R_{AC_n}$$

(It is the sum of relevance of A for each of the topic taken separately)

• If several competences \{C_1, C_2, \ldots, C_n\} are used as input for searching in DBLP and if the user has assigned a degree of importance \{D_1, D_2, \ldots, D_n\} respectively for each of the competences, the relevance $R_{A\{C_1, C_2, \ldots, C_n\}}$ of an author A for the topics \{C_1, C_2, \ldots, C_n\} is defined as follows:

$$R_{A\{C_1, C_2, \ldots, C_n\}} = D_1^* R_{AC_1} + D_2^* R_{AC_2} + \ldots + D_n^* R_{AC_n}$$

(It is the sum of relevance of A for each of the topic taken separately, dampened with the degree of importance of each of the topic)

4.5.2 Database

Database (DB) is known as a structured collection of records which facilitates and speeds the retrieval of information. For our thesis work, we have built a relational database where we have stored the profiles of different persons with different competences. The database has been built with Access and it contains the following tables as presented in the figure below (figure 18). The most important tables are the table person and the table competences. The other tables will be used to give deeper information about the person if his profile matches with the competences needed.
Regarding the definition of the relevance of the results, three cases have been considered here as well. The way the relevance is defined is quite similar to the way it was defined in the case of DBLP. The only difference resides in the way the relevance of an author for a topic is computed.

- If one competence \( C \) is used as input for searching in the DB, the relevance \( R_{AC} \) of an author \( A \) for a topic \( C \) is defined as follows:

\[
R_{AC} = \text{number of years } A \text{ has practiced the competence(or skill) } C
\]

- If several competences \( \{C_1, C_2, \ldots, C_n\} \) are used as input for searching in the DB, the relevance \( R_{A\{C_1, C_2, \ldots, C_n\}} \) of an author \( A \) for the topics \( \{C_1, C_2, \ldots, C_n\} \) is defined as follows:

\[
R_{A\{C_1, C_2, \ldots, C_n\}} = R_{AC_1} + R_{AC_2} + \ldots + R_{AC_n}
\]

- If several competences \( \{C_1, C_2, \ldots, C_n\} \) are used as input for searching in the DB and if the user has assigned a degree of importance \( \{D_1, D_2, \ldots, D_n\} \) respectively for each of the competences, the relevance \( R_{A\{C_1, C_2, \ldots, C_n\}} \) of an author \( A \) for the topics \( \{C_1, C_2, \ldots, C_n\} \) is defined as follows:

\[
R_{A\{C_1, C_2, \ldots, C_n\}} = D_1 \times R_{AC_1} + D_2 \times R_{AC_2} + \ldots + D_n \times R_{AC_n}
\]
5 Results

As a result of this project, a software has been developed with respect to the different modules described above (EM analyzer, Query Generator, User Interface). The different modules put together fulfill the main goal of this thesis work which is to apply Enterprise Models as Interface for Information Searching.

The results of the different modules are presented below knowing that the user interface is in fact the support which allows the interaction between the user and the different modules of the software.

5.1 Enterprise Model Analyzer

The implementation of the architecture built earlier was done in Java. As mentioned previously, there is a little intervention of the user during the information extraction process. The first operation the user has to do is to select a Troux file. This proposition is done by the graphical user interface below:

![Figure 19: Selection of a Troux file](image)

After selecting the Troux file, the user is asked to choose a starting container and an ending one, as displayed in the following figure:

![Figure 20: Selection of the starting and ending containers](image)
Then the list of all the objects belonging to the starting container is displayed. In this example, the starting container selected was the container ‘Roles’ and therefore all the objects of type ‘Roles’ (or simply all the roles) are displayed. The result is as presented here:

![Figure 21: List of roles for the case described above](image1)

The ending container selected (this operation cannot be seen in the figure above) was ‘Competences’, because our intention is to get all the competences required for a specific role. The next step for the user is to select the object of his choice by double-clicking on it. This selection of the object of his choice could also be done by highlighting the object followed by clicking on the search button on the right. The user has also the possibility of making a choice of multiple objects. In that case, the result is the union of the results of each single object. Redundancies are automatically removed.

![Figure 22: Competences required for an ontology expert](image2)
Results

The figure above (figure 22) is the ending point of the information extraction process. This process has started with the user specifying the Enterprise Model file from which the information should be retrieved. Then the user specified that he was interested in competences that are linked to roles. He selected the role ‘Ontology Expert’ and got the final results in figure 22 (the competences context, ontology, pattern and rule engineer).
5.2 Query Generator

As said earlier, we have identified four possible ways of defining a ‘candidate for a role’ when this role requires a set of competences. One of the ways to give this definition was to let the user himself define what he means by assigning a degree of importance to each of the competence found. This is done by the means of a checkbox ‘PersonalizeSearch’ provided to the user once the competences have been extracted. The use of this definition has resulted in a construction of a new window to manage this functionality of the software. This window is shown in the figure below (figure 23):

![Figure 23: Interface for query personalization](image)

Furthermore, we found that this definition of candidate can meet up with one of the three other definitions depending on the way the degree of importance is assigned.

If for instance all the competences are assigned a degree of competence ‘critical’, then it means ALL the competences MUST be there and it is equivalent to the definition ‘a candidate for a role has all the competences’.

If instead, all the competences are assigned a degree of competence ‘Important’, then it means that ALL the competences DO NOT NEED to be there and it is equivalent to the definition ‘a candidate for a role has some of the competences’.

Finally, if we have a mix of degree of relevance (critical, very important, important) then it means that SOME of the competences MUST be there and it is equivalent to the definition ‘a candidate for a role has at least m competences among n(m<n)’.
Results

In conclusion, the value of this interface resides in the fact that if helps to get a better expression of the need of the user because it is not possible to predict all the thoughts of the user beforehand. Once this need is expressed, the query construction and execution depends on both the definition of ‘candidate’ used and the data source used to get our final results. The specificities are presented in the following sections.

5.2.1 Query construction and execution for Google

The second method of query building for search engines presented earlier (Section 4.3.2.1, figure 12) could not be implemented. The main reason is that it is almost impossible to interpret the intermediate results (which are texts with no identifiable structure or not organized in a way so that we could deduce profiles of people from it). Hence, we have implemented only the first method proposed (Section 4.3.2.1, figure 11)\(^3\). All the tests were performed using Google.co.uk on November 17, 2009. As said earlier, we’ve just gone through the first five pages of results to get links to people’s profiles. That means that all the results presented below are out of 50 results because each page in Google.co.uk displays 10 hits.

5.2.1.1 A ‘candidate’ for a role has all the competences

We have interconnected all the query terms with the Boolean AND and sent the query as such in Google. We found no links to profiles of people within the first five pages of results for 2 query terms. We have an average one link to a person’s profile when we used 3 or 4 query terms.

5.2.1.2 A ‘candidate’ for a role has some of the competences

The query terms were interconnected with the Boolean OR and the phrase query obtained was sent to Google. We found no links to profiles of people within the first five pages of results (for 2, 3 and 4 query terms).

5.2.1.3 A ‘candidate’ for a role has at least n competences among n (m<n)

We didn’t make these tests but their results can be easily deduced from the previous two tests above. Using the case provided, the maximum value of m is 3 and from the previous tests (sections 5.2.1.1 and 5.2.1.2), we know how much we get for each type of query. Thus, using the previous results, we can say that if m = 3, then the number of results is one (see section 5.2.1.1); but if m<3, there is no links to profiles of people when we use this interpretation of ‘candidate’.

\(^3\) Note: all the tests presented here (and in the following sections) were performed using competences required for an Ontology Expert to be in line with the results presented earlier (Section 5.1) for the Enterprise Model Analyzer.
5.2.1.4 Query expansion

Each execution of a query in Google involves opening one web browser. Applying the concept of query expansion to the execution of query in Google would have involved opening several Google web pages for the user. If we take for example the request ‘context AND ontology’ and we use the synonyms found before (see table 4), we get at least four derived web requests when we use query expansion: ‘context AND conceptualization’, ‘context AND description’, ‘circumstance AND ontology’, ‘setting AND ontology’. This number of derived requests can even be much bigger if we consider associating the synonyms of the terms between them. Our intention is not to overload the user with results, rather to give him a reasonable amount of results both in terms of relevance and volume. Furthermore, Google itself is already equipped with query expansion algorithms. For these two reasons, we have considered that applying query expansion to the query build for Google Search Engine was of little interest.

5.2.1.5 Addition of ‘contextual’ keywords to the query

The results we got from the previous tests (sections 5.2.1.1 to 5.2.1.4) have lead us to consider adding ‘contextual’ keywords to the query in order to improve the precision (which was 1/30 for ‘AND queries’ and 0 for ‘OR queries’). These ‘contextual’ keywords are words that will help to give a more specific context to the query. We have built and tested new queries by incorporating two different ‘contextual’ keywords.

The first keyword we used was ‘CV’, which was added at the end of the query. ‘CV’ was added because it is a keyword for people’s profile. This means for example that a query that was ‘context OR ontology’ becomes ‘context OR ontology CV’. But ‘CV’ is an abbreviation that means different things depending on the context of use. For instance, it can mean ‘Consonant Vowel’ in linguistic or be used to refer to the term ‘cardiovascular’ in medicine. For this reason, we have considered the use of a second set of keywords for a better delimitation of the context of our query.

The second keyword (or set of keywords) we used was ‘PEOPLE....CV’. Our intention when adding ‘people’ was to make the context more precise and say that we are looking only for people’s CV. By this way, ‘context AND ontology’ becomes ‘PEOPLE context AND ontology CV’.

We have identified different subsets of 2, 3 and 4 query terms. These different subsets were used to build queries that were later on sent into Google. Here are the summary of the results we obtained from the tests which were performed using Google.co.uk on November 14, 2009. More details on the results are provided in appendix2.

---

4 Further details on this topic are given in the article from the Search Engine Journal ‘What is Google Query Expansion? Cases and Examples’.

5 “The two most frequent and basic measures for information retrieval effectiveness are precision and recall”. Precision is the fraction of retrieved documents that are relevant whereas recall is the fraction of relevant documents that are retrieved (Manning et al., 2009).
Results

Using the competences extracted previously for an ontology expert, we can identify 6 different subsets of 2 query terms which are:

- \{\text{context, ontology}\} which is referred to as 2TQuery1 on the histograms
- \{\text{context, pattern}\} which is referred to as 2TQuery2 on the histograms
- \{\text{context, ‘rule engine’}\} which is referred to as 2TQuery3 on the histograms
- \{\text{ontology, pattern}\} which is referred to as 2TQuery4 on the histograms
- \{\text{ontology, ‘rule engine’}\} which is referred to as 2TQuery5 on the histograms
- \{\text{pattern, ‘rule engine’}\} which is referred to as 2TQuery6 on the histograms

![Figure 24: Addition of ‘contextual keywords’ (2 query terms, Boolean AND)](image)

![Figure 25: Addition of ‘contextual keywords’ (2 query terms, Boolean OR)](image)
Results

We can also identify 4 subsets of 3 query terms which are:

{context, ontology, pattern} which is referred to as 3TQuery1 on the histograms
{context, ontology, ‘rule engine’} which is referred to as 3TQuery2 on the histograms
{context, pattern, ‘rule engine’} which is referred to as 3TQuery3 on the histograms
{ontology, pattern, ‘rule engine’} which is referred to as 3TQuery4 on the histograms

Figure 26: Addition of ‘contextual keywords’ (3 query terms, Boolean AND)

Figure 27: Addition of ‘contextual keywords’ (3 query terms, Boolean OR)

Finally, we can identified 1 subset of 4 query terms, which is
{context, ontology, pattern, ‘rule engine’} which is referred to as 4TQuery on the histograms.

Figure 28: Addition of 'contextual keywords' (4 query terms, Boolean AND)

Figure 29: Addition of 'contextual keywords' (4 query terms, Boolean OR)
5.2.2 Query construction and execution for DBLP

After the execution of a query using its interface, DBLP returns a list containing 16 authors, ranked according to the number of publications they have authored or co-authored in a given domain. Here (figure 30) is an example of a list of results (first five results) for when the query term ‘ontology’ is selected and sent in DBLP.

![Figure 30: First Five results of the query 'ontology' in DBLP](image)

The result of the tests against DBLP are summarized in the following sections (5.2.2.1 to 5.2.2.4). These tests were performed on November 20, 2009. More details on the results are given in appendix3.

5.2.2.1 A ‘candidate’ for a role has all the competences

We have interconnected all the query terms with the Boolean AND. The resulting query was sent into DBLP.

**Method1:** Using this method, we were able to get people but only when we’ve used 2 query terms. We got no result when 3 and 4 query terms were used.

**Method2:** No people were found for 2, 3 and 4 query terms.

5.2.2.2 A ‘candidate’ for a role has some of the competences

We have interconnected all the query terms with the Boolean OR. The resulting query was sent into DBLP.

**Method1:** Some of the queries gave us some people, others return no results, but mostly, we didn’t get any results when we used this method.
Results

Method2: This method has given results for all queries. The particularity of the results obtained here is that one gets lists of different sizes depending on whether we use 2, 3 or 4 query terms. The lists’ size are respectively 16, 32, 48.

5.2.2.3 A ‘candidate’ for a role has at least m competences among n (m<n)

The results here are deduced from the results of the section 5.2.2.1.

Method1: Depending on the value given to m, we may have a variation in the final result at the end. If m=3, it implies that the ‘candidate’ for a role has at least 3 competences among 4. But as we didn’t get any results when we used 3 query terms previously (see section 5.2.2.1) we will get no people when we use this definition. If m <3 (that is to say m=2) instead, then we will get some people to answer our query as we have seen in section 5.2.2.1 above.

Method2: Using this definition will lead us to no people returned by DBLP because no results were found for 2,3 and 4 query terms in section 5.2.2.1.

5.2.2.4 Query expansion

From the previous tests, it appears that no results are found when we use subsets with more than 2 query terms (interconnected with AND as well as OR). For that reason, we decided to test the query expansion only on subsets of 3 and 4 query terms because the differences would be easier to observe. The query expansion algorithm has been applied using both method1 and method2. Using the given thesaurus above (section 4.3.2.3, table 4) and one synonym for each term, we can derive these new sets of 3 query terms. The tests were performed on November 23, 2009. The details of the results are provided in appendix4.

Method1:

• Boolean AND
We obtained no results when using either subsets of 3 or 4 query terms.

• Boolean OR
For 3 query terms, we had in overall 12 derived queries; 6 of them have returned results whereas we didn’t got any results for the other 6 remaining.
For 4 query terms, only one (1) derived query(out of 4) has returned a result when we’ve used DBLP.

Method2:

• Boolean AND
Just like in method1, we obtained no results when using either subsets of 3 or 4 query terms.

• Boolean OR
We didn’t perform the tests using this method because the previous tests were already satisfactory(see section 5.2.2.2).
5.2.3 Query construction and execution for a MS Access database

A query on a database returns tuples. A tuple (or record) is a collection of column values. An example of tuple is provided here:

Figure 31: An example of tuple

5.2.3.1 A ‘candidate’ for a role has all the competences

This definition of ‘candidate’ asks for the interconnection of all query terms with the Boolean AND.

**Method 1:** If we reuse the generic SQL query with multiple competences as input (see figure 14 in section 4.3.2.2) and assume that the user is looking for two competences that are ‘context’ and ‘pattern’, this query becomes:

```
1 SELECT person_attributes // the desired attributes of the person.
2 FROM database_tables // the tables in the database.
3 WHERE person.competence= ‘context’ AND person.competence= ‘pattern’ // 2 constraints are put on the competence of the person
```

Legend: // In blue: comments

Figure 32: SQL query with ‘context’ and ‘pattern’ as input (Boolean AND)

A result for this query is a tuple for which the column ‘competences’ (see figure 31) has both the values ‘context’ and ‘pattern’ at the same moment. And this is not possible because in a database, each column of a tuple has one and only one value at a time. Because of this technical limitation of the Ms Access database, we could not implement this method.

**Method 2:** This method was used to get results from the database. An example of result when we use the query ‘context AND pattern’ presented before is displayed below (figure 33)

Figure 33: Result of the query ‘context AND pattern’ in the Ms Access database

Here the result displayed is ‘Antony Makoumba’ with a relevance 38. If the user wants to get more details about the profile of this person, he can click on his name and then he gets the following result (figure 34)
Figure 34: An example of complete profile of a person in the Ms Access database

This figure represents the CV of Anthony Makoumba. We can see that he has experience in the competence ‘context’ for 32 years and in the competence ‘pattern’ for 6 years and that is the reason why we got the relevance 38 (= 32+6) in figure 33 above. We can also have other information like his education and his previous experiences which are quite useful to get a complete idea of his professional profile.

5.2.3.2 A ‘candidate’ for a role has some of the competences

This interpretation of the term ‘candidate’ has lead us to interconnect all the terms with the Boolean OR instead. Let us look at the results obtained for both methods proposed earlier.

**Method1:** We will assume here also that the user is looking for two competences that are ‘context’ and ‘pattern’. The generic SQL query with multiple competences as input (see figure 14 in section 4.3.2.2) becomes:

```
1 SELECT person_attributes // the desired attributes of the person.
2 FROM database_tables // the tables in the database.
3 WHERE person.competence=’context’ OR person.competence=’pattern’ // 2 constraints are put on the competence of the person
```

Figure 35: SQL query with ‘context’ and ‘pattern’ as input (Boolean OR)

In a database context, this query means that we are looking for tuples for which the column ‘competences’ has either the value ‘context’ or ‘pattern’. The result is given on the figure below (figure 36):
Results

Figure 36: Result of the query 'context OR pattern' in the Ms Access database

**Method2**: there is no difference between the results obtained with this method and the results obtained with method1. Here are the different steps of this method

- Firstly, look for the tuples for which the column ‘competences’ has the value ‘context’
- Secondly, look for the tuples for which the column ‘competences’ has the value ‘pattern’
- Thirdly, merge these two results

These three steps taken altogether are exactly equivalent as saying ‘look for tuples for which the column ‘competences’ has either the value ‘context’ or ‘pattern’.

5.2.3.3 A ‘candidate’ for a role has at least m competences among n(m<n)

This interpretation of ‘candidate’ leads us to a query which contains both the Booleans AND and OR. For instance, if we assume that a candidate for a role has at least 2 competences among 4, one query (among many) that could be build to look for the role ‘ontology expert’ is:

'context AND (ontology OR pattern OR rule engine)'

Thus, we can deduce the results of the use of this definition from the previous tests (sections 5.2.3.1 and 5.2.3.2). As seen above, method1 was not applicable when the Boolean term is AND, and method2 and method1 are equivalent when we use the Boolean term OR. Consequently, method2 is the only one that is applicable for this type of queries. Furthermore, this query could be also written as ‘context AND \(\sigma\)’ where \(\sigma = \{\text{ontology OR pattern OR rule engine}\}\). Therefore, the results obtained here will be quite similar (in terms of retrieval principles) to the results obtained in section 5.2.3.1.

5.2.3.4 Query expansion

The results of query expansion are deduced also from the results of the previous tests (sections 5.2.3.1 and 5.2.3.2). We make here the same assumption that the user is looking for ‘context’ and ‘pattern’. We can perform query expansion both for the case when we interconnect the terms with the Boolean AND and the case when we interconnect the terms with the Boolean OR. For that reason, the method suitable for query expansion is method2 because the previous tests have shown that this is the only method that is applicable for both types of Boolean queries. The derived subsets using the thesaurus build earlier (section 4.3.2.3, table 4) are: \{circumstance, pattern\} and \{context, design\}. The final result is the merging of the results obtained separately for each of the subset \{context, pattern\}, \{circumstance, pattern\} and
Results

{context, design}. In other words, we can say that the final results is the result of a query which is:

' (context X ontology) OR (circumstance X pattern) OR (context X design)' with

\[ X = \{\text{AND, OR}\} \] (X is one of the two Booleans).

This query could also be seen as a query \( \alpha \) OR \( \beta \) OR \( \sigma \) where \( \alpha = \text{‘context X ontology’} \), \( \beta = \text{‘circumstance X pattern’} \) and \( \sigma = \text{‘context X design’} \).

Therefore, the results obtained with query expansion are also quite similar (in terms of retrieval principles) to the results obtained in section 5.2.3.2.
5.3 User Interface

A button “Back” (highlighted with the red rectangle in figure 37 below) has been added for every window except the first one which is for the initialization and has no previous step. The user has also the possibility to find competences linked to many roles at the same time. Figure 39 displays an example where the user has selected three roles at the same time as input and gets results. The button “View History” (highlighted with red round dots in figure 37) is for the ‘logs’ of the user and displays the traces of his operations on the application. Every time competences are extracted from the Troux file, a .txt file called ‘tempresults’ is created on the machine of the user to store the results given by the Enterprise Model Analyzer. The location of this file could be found when the user consults the history of his operations (see figure 38).

Finally, we can easily see in figure 39 that the output from the initialization phase (list of all the roles in the current Enterprise Model) is used as input for the Enterprise Model Analyzer module to give the results (the required competences for the selected roles) that will be used for further processing (searching for people).

Figure 37: Buttons 'Back' and 'View History'

Once the user clicks on the button ‘View history’, he gets the history of his operations. An example of history is shown below:

Figure 38: An example of history of the user’s actions
Figure 39: An example of multiple selection of roles and the result
6 Conclusion and recommendations

We start this section with a brief summary of the results found. We present afterwards the different implications of these results and their possible application.

6.1 Summary of the results

Three modules were developed during our work. The User Interface Module is not presented here because it has only supported all the tests made.

6.1.1 Module1: Enterprise Model Analyzer

The construction of this module was done through an iterative process which involves mainly the knowledge engineer (for rules generation) and the expert domain (for validation). During the development of the IE system we realized that it was not possible to completely automate the steps of the information extraction. For example, the user was needed to specify the starting container (‘Role’) and the type of results he wants to get (‘Competences’). The main reason why we needed the intervention of the user is that it is not possible to know beforehand all the specifications that characterise his need. For instance, the name of the Troux file he wants to extract information from will surely change; he may want to find ‘Processes’ that are linked with ‘Roles’, etc.

6.1.2 Module2: Query Generator

6.1.2.1 Query construction and execution in Google

As seen previously, we have built phrase queries by interconnecting the competences with either ‘AND’ or ‘OR’. The queries built were afterwards sent to Google.co.uk and we have counted the number of links to people’s profile found in the first 50 results.

First of all, we realized that it was not possible to assign a relevance to a single hit returned because the retrieved text does not have any identifiable structure or it is not organized in a way so that we could deduce profiles of people from it.

In addition, we can notice that ‘AND queries’ have returned more results than ‘OR queries’. We got 4 results for ‘AND queries’ and 0 result for ‘OR queries’ when we used 3 query terms; we found 1 result for ‘AND queries’ and 0 result for ‘OR queries’ when we used queries with 4 terms. It’s only when we have used 2 query terms that we didn’t found any results in both cases.

This result is surprising in the sense that ‘AND queries’ are more restrictive and consequently are expected to give less results than ‘OR queries’. A possible explanation to that is that we’ve only taken the first five pages of results. We claim that this tendency will be inversed if the whole list of returned hits is scanned.
Conclusion and recommendations

For instance, the query \{context AND ontology AND pattern\} returns about 1,800,000 results\(^6\) whereas the query \{context OR ontology or pattern\} returns about 1,350,000,000 results\(^6\), which is much bigger than the 1,800,000 found for the ‘AND query’. But as said earlier in our report it is not possible to go through the whole list, one needs to define a threshold in order to assign a quality criteria to the returned results.

In our attempt to improve the precision for the results returned by Google, we have considered the addition of ‘contextual’ keywords to the query. We added ‘CV’ and ‘PEOPLE ... CV’. The number of hits returned were much higher when these keywords were added (The differences can be easily seen in figures 26-29, section 5.2.1.5). What is more, the addition of ‘PEOPLE ... CV’ has lead also to more results(in overall) than the addition of ‘CV’ only.

Finally, the results we found suggest that if query construction and execution in Google is to be completely automated, the definition of candidate which is the best is ‘a candidate has some of the competences’, because this is the definition which has given the highest number of results during our tests.

6.1.2.2 Query construction and execution in DBLP

Here both methods of query construction and execution were possible to apply. As a reminder, the first method consisted in building a phrase query (by adding Booleans operators to the terms) and executing this query. The second method consisted in executing a query for each keyword and then intersecting/merging the list of the results obtained respectively.

Applying the first method has lead us to some results both for ‘AND queries’ and ‘OR queries’. The second method gave us no results for ‘AND queries’ but some hits (more than for method1) were found for ‘OR queries’. Thus, our tests imply that building a phrase query for DBLP is more appropriate when we use the Boolean operator AND. On the contrary, if we intend to build ‘OR queries’, then is better to use the merging of the lists because the number of results returned is higher. We should also mention here that these two methods of query generation differ also in their time performance. Building a phrase query and sending it later on in DBLP is much faster (1s to 5s) than executing several queries and then intersecting/merging the list of results (the time can vary from 15s to 80s). This criteria should be also taken into consideration when one is deciding the most appropriate method for a specific situation.

Finally, we have performed query expansion in order to improve the precision for the results. For ‘OR queries’, our goal was achieved but for ‘AND queries’, we didn’t get any results at the end. We suggest two possible explanations to this: either,

- There is effectively no publication that contains the different terms we were looking for, or
- The reason may be that even if DBLP uses keywords matching techniques like Google, it is a bit more restrictive. For instance, DBLP takes into consideration the order of the terms in the query. For example, the query ‘formal semantic model’ returns\(^7\) six authors as results whereas the query ‘model semantic formal’ returns\(^7\) nothing.

\(^6\) The tests were performed using Google.co.uk on November 27, 2009.
\(^7\) The tests were performed using DBLP on November 27, 2009. Details on the results are provided in appendix5
We can conclude this section saying that the definition of candidate which is best for a complete automation of the query generation process is ‘a candidate has some of the competences’, because here as well, this definition has given the highest number of results during our tests.

6.1.2.3 Query construction and execution in a MS Access database

It would have been of little interest to look at the number of hits retrieved because we created the Ms Access database ourselves and consequently we knew what was inside. For this reason, we focused on the methods proposed earlier and their applicability. Instead of phrase queries, SQL queries were build here because SQL is the standard language for querying relational databases.

Regarding the results, it was not possible to interconnect multiple competences with ‘AND’ in a single SQL query because of the technical limitations of the database. We needed to execute various SQL queries and then intersect the results at the end. When the Boolean is OR instead, we found that both approaches for query construction and execution were equivalent.

Finally, we would say that the best definition of candidate would be ‘a candidate has some of the competences’ because it gives more flexibility on the choice of the approach to use to build and execute the query.

6.2 Generalization of the results

6.2.1 The use of Enterprise Model as interface for information searching

The results found previously show the use of Enterprise Model as interface for information searching for the three different types of data sources (unstructured, structured and semi-structured). An Enterprise Model can also be used as interface for information searching in both internal data sources (like a local database) and external data sources (like a remote collection of documents).

An important precision to make here is that such use is possible if and only if the Enterprise Model is stored in a way so that information extraction is possible from it. In other words, the file storage format should allow for interpretation of the data inside the stored EM. The extension .XML was used during our work, but our results are valid for any other extension of Enterprise Model files provided that this extension gives a structure in a way or another to the content of the file.

Finally, when dealing with external data sources, an additional requirement is an access point(or mechanism) to the remote data source(e.g. an interface).

6.2.2 Building an Information Extraction System

Using the results found previously, we can suggest that there are at least three types of actors involved in the process of building an IE system (for a given domain) using a knowledge engineering approach. These are:
• **Knowledge provider**: his role is to provide knowledge about the domain. It can be either a domain expert like it has been during our work or it can also be a documentation about the current domain of work. The knowledge provider is responsible for the accuracy of the given information about the domain of work.

• **Knowledge engineer**: his role is to translate the domain knowledge into rules for the IE system. He is responsible for the accuracy of the rules written in the system.

• **Information consumer**: his role is to specify the current information need and question the IE system accordingly. He is responsible for the accuracy of the specification of his information need.

These different roles are cumulative. In our case for instance, the knowledge provider was at the same time the final user of the IE system. The knowledge provider and the knowledge engineer are important because if there is no one to give knowledge about the current domain or if there is no one to translate this knowledge into rules, it will surely not be possible to build the IE system. But we consider the information consumer as equally important because he is the one that is going to use the system at the end and consequently, he should have his word to say (mainly regarding the way the interactions with the IE system are handled).

Finally, we suggest that these three roles are required during the whole life of the IE system built because after the launching of the application, one needs maintenance and evolution. These actors are required at these moments also, in order to keep the IE system both consistent with the domain of work and acceptable to the final user.

### 6.2.3 Approaches for query construction and execution

From the different results we have seen, we can suggest that there are two approaches (at the technical level) for query construction and execution. The first approach consists in building one query which contains all the constraints and execute this query. It is more adequate to use this approach when phrase queries are involved in the query generation process.

The second approach consists in building as many queries as terms at disposal and then, intersect or merge the intermediate results. The use of this approach is more adequate when structured queries are involved in the query generation process. Regarding the second approach, one needs also to be aware that the less the data source is structured, the less this approach is applicable. The reason is that if the data source is much unstructured, then the intermediate results cannot be interpreted and therefore intersected/merged to give a final result.

### 6.2.4 Query generation and domain of the final result

We refer to the domain of the final result (or simply domain) as what we get at the end. An example of domain is people’s profile like it was during our work. Another example can be ‘car’(if one is looking for the best car for a race) and so on. We find out that the query and the type of data source can complement each other in the determination this domain of the final results.
When the data source is structured or semi-structured, the deduction of the domain is quite straightforward. By the database structure for example, one can easily conclude if we are talking of people, articles sold in a shop, etc. But when the data source is unstructured, the use of additional keywords that are domain-specific in the query is of worth value to improve the quality of the final results.

6.2.5 Query generation: which definition of ‘candidate’ to use?

This section concerns only the case when we are looking for a ‘candidate’ for a set of requirements. Based on the results found previously, our conclusion is that if the user is involved in the query generation process, the best to do is to let him define what a ‘candidate’ is through assigning a degree of importance to each of the requirements. If the query generation process is to be completely automated, then we suggest that ‘candidate’ should have some of the requirements because it will give a larger set of results to the user at the end.

6.2.6 Summary

We intend to conclude on the results found with this figure (figure 40):

![Diagram](image)

**EM = \{role, competence, etc.\}**

**\{people, car, goods, etc.\}**

Figure 40: From an EM to concrete objects

The figure above shows ‘two worlds’. On the one hand, we have an Enterprise Model which is about abstract concepts and links between these concepts. These concepts exist only in the mind of individuals and cannot be perceived by the senses. An example of an abstract concept is a competence. On the other hand, we have concrete objects. These are objects that belong to immediate experience or actual things or events; they are real, tangible. An example of a concrete object is a person (with a certain age, profession, marital status, etc.). Our work has shown that it is possible to build a bridge between these two worlds, through the use of an Enterprise Model Analyzer followed by a Query Generator.
6.3 Applicability of the results
These results can be used by any institution or companies that have already Enterprise Models for their activity. These Enterprise Models can be used as starting point for the construction of tools to aid decision-making process. Because the EM is the starting point, the more it is accurate in capturing the reality, the more the final tool will be helpful for the final user.

In companies for example, one could think of establishing a link between an EM and human resources databases with software modules similar to the EM Analyzer and the Query Generator. These software modules will help to find people that are the most appropriate for a role and can help a manager to create a completely new team or reorganize existing ones. By building only an Enterprise Model Analyzer, companies can have a tool that will help to support the recruitment process in the sense that information extracted using this module can give an idea of the profile of the people to look for. However it is important to be aware that the scope of applicability is broader than the sole domain of human resources. One could think of any problem where the objective is to find the best object for a requirement or a set of requirements. Examples could be the best car for this race (Formula 1), the best medicine for this diagnosis (healthcare), etc.

6.4 Recommendation for further studies

6.4.1 Challenges to address in Enterprise Modeling
The implication of our results are that storage formats that allows for the interpretation of the content should be favored for EM Tools in order to improve the interoperability with tools for information extraction. Thus, future research could orientate towards:

- ‘How to store the content of a EM file in a structured or semi-structured way?’ or
- ‘How to make conversion of EM tools’ storage formats to a standard format like XML?’.

This will help to make the enterprise knowledge contained in the file sharable and herewith, available for reuse by other tools.

6.4.2 Challenges to address in Information Extraction
As we have seen earlier, the user is required in the information extraction process. Therefore, a question which naturally arises is ‘where should the person stop and the information extraction interface start?’.

Further studies should be conducted in order to clarify

- The borders of the user involvement and system involvement in the information extraction process.
- Both how much activity and which type of activity the user should be able to direct the system to do at once.
6.4.3 Challenges to address in Information Retrieval

6.4.3.1 Information retrieval from several sources
During this work, we have seen the search of information in several sources. But these searches were conducted separately and were independent from one another. A field of study that could be addressed also is the integration of results of searches from various data sources. If it was possible, for instance, to make the use of these results in a synergistic way, that will be of great benefit for the final user.

6.4.3.2 Information retrieval from unstructured data sources
Our results suggest also that studies are conducted towards finding ‘how to improve the quality of information retrieval from unstructured data source?’ because the outcomes were not satisfactory enough in our case.

6.4.3.3 Define the relevance of a returned result
We have seen earlier that it was not possible to assign a relevance for a result returned by Google. Regarding DBLP, we have used the number of articles authored or co-authored by an author to define his relevance for a domain. But this way of considering relevance was vehemently criticised by Parnas (2007). He argues that researchers (or authors) should not be measured by the number of papers they publish, rather by the correctness, importance, real novelty, or relevance of their contributions. These measurements, even if they are more accurate and can be easily understood in human language, are very challenging to express in a software system. For the MS Access database, we used the number of years a person has practiced a competence to define the relevance but this was a very simplistic definition. Other parameters could have been taken into account like his education and work experience to make it more accurate.

Thus, the best way to define a relevance of a returned result remains an open question.
7 References


65


W3C. Retrieved October 7, 2009, from [http://www.w3schools.com/Xml/xml_dtd.asp](http://www.w3schools.com/Xml/xml_dtd.asp)


8 Appendix

Appendix1: Requirements Specification For the Software AEMIS.........................70

Appendix2: Addition of ‘contextual’ keywords to the queries and search in Google.co.uk.................................................................76

Appendix3: Results of the search in DBLP.................................................................82

Appendix4: Query expansion and search in DBLP.....................................................85

Appendix5: Example of execution of query in DBLP................................................88

Appendix6: Troux’s Enterprise Model used to test the software..............................90
8.1 Requirements Specification For the Software AEMIS

(Applying Enterprise Models as Interface for Information Searching)
8.1.1 Introduction

8.1.1.1 Purpose

This document is the requirements specification for the software we have to develop for our thesis work. The document aims at providing all the requirements needed to develop such software for the research group Information Engineering at Jonkoping University. These requirements would help to well understand the needs of the Research group and establish the services the software should provide in a structured way that will facilitate the process of development, validation and test of the software. Therefore, the document will be useful for the Research group, for the students whose task is to develop the software and maybe for other students (who will have to maintain or improve the software in the future).

8.1.1.2 Scope

The growing use of Enterprise Modeling in companies has lead researchers to think about other ways to make Enterprise Models more advantageous. In fact, Enterprise Models can be more than just models but they can be used in a more dynamic way which is through a software program. The Software which has to be built is called AEMIS (Applying Enterprise Models as Interface for Information Searching). It aims at exploring the use of Enterprise Models as interface for information searching. For that the software should be able to extract information for the Enterprise Models and use them as input for information searching in search engines and database.

8.1.1.3 Definitions and acronyms and abbreviations

In this part, the user will find some useful definition of abbreviations or acronyms used throughout the document:

AEMIS: Applying Enterprise Models for Information Searching
SRS: Software Requirement Specification
JTH: Jönköping Tekniska högskolan

8.1.1.4 Overview

In the remainder of the report, we present first of all an overall description of the general factors that affect AEMIS and its requirements. Then, we describe further the different services provided by AEMIS. Furthermore, we present the different constraints imposed on the software.
8.1.2 Overall description

8.1.2.1 Product perspective

For this software to fulfill its main requirement which is to apply Enterprise Models as interface for information searching, it needs to be linked to other systems or other components. Therefore, the software is not self-contained. In fact, on one hand, AEMIS is dependent on the type of file (containing the Enterprise Models) from which the information will be extracted. On the other hand, AEMIS is dependent on the types of search engine and/or database used to retrieve the information. The figure below illustrates briefly the major components of the larger system and their interconnections to AEMIS:

![Diagram of AEMIS part of the larger system]

Figure 41: AEMIS part of the larger system

8.1.2.2 User interface

The user interacts with the software through a user interface which should make AEMIS easy to understand and use, resulting in greater user acceptance. The user interface is also a means of AEMIS to communicate with the other parts of the larger system which are the file (containing the Enterprise Models), the search engines and the Database. Therefore it should include some buttons for:

- Selecting the Troux File on the disk.
- Undertaking researches in search engines and database.

8.1.2.3 Product functions

AEMIS is designed to provide at least three functions which are:

- the extraction of information from the Enterprise Model specifying the request to be used for information searching.
- the generation and execution of a query based on the request in the desired information system.
- the storage and visualization of the results of the query.
8.1.2.4 User characteristics

The different users of AEMIS should have some knowledge about Enterprise Models. In other words, they should be able to create, read through and understand Enterprise Models. The users should also have some basic knowledge about the software Troux. Knowledge about XML format is not mandatory.

8.1.2.5 Constraints

The software should be workable fulfilling all the requirements needed. The software should be as much automatic as possible. In other words the software should not require lot of human intervention. Herewith, it should be directly connected to the different search engines and database as long as the user has selected the kind of competences he is interested in and the kind of search he wants to undertake (into a search engine or a database).

8.1.2.6 Assumptions and dependencies

An assumption for AEMIS is that this software should be stored on a JTH server (if there is no place available on the existing ones, a new one should be bought). Therefore, all the users (people of the Research group) of the software should have a direct access to the software on the server. For the user to use the software they should have an Operating System which could be for example, Microsoft Windows 2000, Microsoft Windows NT 4.0 with Service Pack 6a or later, Microsoft Windows XP or Microsoft Vista.

8.1.3 Specific requirement

8.1.3.1 User requirement

- Functional requirement

1. Functional requirement 1
   The software should be available for all the persons who would like to use it within the research group Information Engineering.

2. Functional requirement 2
   The software should need less human intervention. It should be as automatic as possible.

3. Functional requirement 3
   AEMIS must be able to extract information from the Enterprise Model specifying the request to be used for information searching.

4. Functional requirement 4
AEMIS must be able to generate and execute a query based on the request in the desired information system.

5. **Functional requirement 5**
AEMIS must be able to provide a way for the storage and visualization of the results of the query.

6. **Functional requirement 6**
The interface of the software should be easy to use and easily understandable.

### 8.1.3.2 Software requirement

1. **Functional requirement 1**
The software should be available for all the persons who would like to use it within the research group Information Engineering.
   - **Software requirement specifications**
   1.1 The software should be stored on the server which belongs to JTH.
   1.2 The research group should have a shared space on the server where the software will be stored.
   1.3 The software does not need to have an interface for identification as it will be stored internally in the database.

2. **Functional requirement 2**
The software should need less human intervention. It should be as automatic as possible.
   - **Software requirement specifications**
   1.1 The human intervention needed to access the software are more related to the expression of the user’s needs in a clear way.
   1.2 Each intervention of the user is followed by some automatic actions performed by the software.
   1.3 When the user select the Troux file with the extension .kmv, the system should automatically:
   - Change the format of the file into a (.XML) file.
   - Go through the XML file and extract the names of all the containers which will be suggested as a list to the user.
   1.4 When the user selects the starting and ending containers, the system should automatically:
   - Extract from the XML file, all the objects (the roles) belonging to the starting container and suggest them as a list to the user.
   1.5 When the user selects the object(s) he is interested in, then the system should automatically:
   - Extract all the objects of the ending container (competences) type related to the previously selected objects.
1.6 When the user selects the competence(s) he is interested in, and selects the type of 
search engine or database

- The search engine or database returns the results (or profile of persons) 
matching with the competences selected.

3. **Functional requirement 3**

AEMIS must be able to extract information from the Enterprise Model specifying the request to be used for information searching.

- **Software requirement specifications**

1.1 Change the format of the file containing the models into a (.XML) format.
1.2 Go through the XML file and extract the names of all the containers which will be 
suggested as a list to the user.
1.3 The list of containers will be contained into two identical lists the starting and the 
ending container.
1.4 The user has to select his starting (roles) and ending containers (competences).
1.5 The system should list all the objects of the starting container (roles).
1.6 The user will select the role(s) he is interested in.
1.7 The system should go look for all the objects linked to this role (syntax).
1.8 For each object linked to this role, it should check the type of object to see if it is 
of type competence (semantic).
1.9 All the objects linked to the role(s) selected and of type Competences should be 
displayed as result in a list to the user.
1.10 Those objects will be used later on for information retrieval in search 
engine or database.

4. **Functional requirement 4**

AEMIS must be able to generate and execute a query based on the request in the 
desired information system.

5. **Functional requirement 5**

AEMIS must be able to provide a way for the storage and visualization of the results 
of the query.

- **Software requirement specifications**

1.1 The visualization of the results should be done through the user interface
1.2 A (.txt) file is automatically created and stored in the system. This file contains the 
information retrieved by the software.

6. **Functional requirement 6**

The software interface should be easy to use and easily understandable.

- **Software requirement specifications**

1.3 The interface of the software should contain just the minimum required to make 
the software work at each step of the process.
1.4 The interface should contain some graphical facilities such as buttons, scroll bars, 
iconic images which are more illustrative of the different functionalities of the 
software.
1.5 The interface should contain the logo of the university at each step of the process.

1.6 The first page of the software should contain
- Firstly, the interface should provide button ‘BROWSE’ to browse and select the Troux file on the system.
- Then, combo boxes will be used to provide the lists of containers to the user.
- To validate his choice of containers one button ‘OK’ will be close to the combo boxes.
- A button ‘CLOSE’ should also be provided in case the user would like to close the software.

1.7 On the second page the user is supposed to select the list of roles he is interested in.
- For that a list will be used with one button ‘Search’.
- Buttons ‘BACK’, ‘CLOSE’ and ‘VIEW HISTORY’ should be there in case the user would like to go back to the previous step, close the software or see the history of his searches.

1.8 Once the user has selected the role(s) he is interested in, the competences related should be displayed in a list
- At the bottom of the page three buttons ‘Google’, ‘DBLP’ and Database representing the different types of search.
Remarks
The tests have been performed on November 17, 2009. The results were counted using the first five pages.

A = Page number in Google. An example is given here

B = Query terms used.

In each column is given the number of links to people’s profiles found on a page of results in Google.co.uk. The column ‘Sum’ represent the sum of all number of the links found from page1 to page5.
### Appendix2

#### 2 TERMS

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Sum=</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>context AND ontology</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>context AND pattern</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>context AND ‘rule engine’</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ontology AND pattern</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ontology AND ‘rule engine’</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>pattern AND ‘rule engine’</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

#### 2 TERMS

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Sum=</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>context AND ontology</td>
<td>6</td>
<td>5</td>
<td>7</td>
<td>6</td>
<td>7</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td>context AND pattern</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>context AND ‘rule engine’</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>ontology AND pattern</td>
<td>4</td>
<td>8</td>
<td>6</td>
<td>6</td>
<td>4</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>ontology AND ‘rule engine’</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>pattern AND ‘rule engine’</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

#### 2 TERMS

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Sum=</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>context AND ontology</td>
<td>4</td>
<td>8</td>
<td>5</td>
<td>9</td>
<td>6</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>context AND pattern</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>context AND ‘rule engine’</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>ontology AND pattern</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>ontology AND ‘rule engine’</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>pattern AND ‘rule engine’</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>A</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>Sum=</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>------</td>
</tr>
<tr>
<td>context OR ontology</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>context OR pattern</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>context OR ‘rule engine’</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ontology OR pattern</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ontology OR ‘rule engine’</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>pattern OR ‘rule engine’</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B</th>
<th>A</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Sum=</th>
</tr>
</thead>
<tbody>
<tr>
<td>context OR ontology</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>7</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>context OR pattern</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>context OR ‘rule engine’</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>3</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>ontology OR pattern</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>ontology OR ‘rule engine’</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>pattern OR ‘rule engine’</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B</th>
<th>A</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Sum=</th>
</tr>
</thead>
<tbody>
<tr>
<td>context OR ontology</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>4</td>
<td>7</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>context OR pattern</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>context OR ‘rule engine’</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>6</td>
<td>5</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>ontology OR pattern</td>
<td>3</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>6</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>ontology OR ‘rule engine’</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>pattern OR ‘rule engine’</td>
<td>2</td>
<td>7</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>22</td>
<td></td>
</tr>
</tbody>
</table>
### 3 TERMS

#### Boolean AND

<table>
<thead>
<tr>
<th>B</th>
<th>A</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Sum=</th>
</tr>
</thead>
<tbody>
<tr>
<td>context AND ontology AND pattern</td>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>context AND ontology AND ‘rule engine’</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>context AND pattern AND ‘rule engine’</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>ontology AND pattern AND ‘rule engine’</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

#### Boolean AND + ‘CV’

<table>
<thead>
<tr>
<th>B</th>
<th>A</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Sum=</th>
</tr>
</thead>
<tbody>
<tr>
<td>context AND ontology AND pattern</td>
<td></td>
<td>4</td>
<td>8</td>
<td>3</td>
<td>6</td>
<td>7</td>
<td>28</td>
</tr>
<tr>
<td>context AND ontology AND ‘rule engine’</td>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>context AND pattern AND ‘rule engine’</td>
<td></td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>ontology AND pattern AND ‘rule engine’</td>
<td></td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
</tbody>
</table>

#### Boolean AND + ‘PEOPLE…CV’

<table>
<thead>
<tr>
<th>B</th>
<th>A</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Sum=</th>
</tr>
</thead>
<tbody>
<tr>
<td>context AND ontology AND pattern</td>
<td></td>
<td>2</td>
<td>9</td>
<td>7</td>
<td>6</td>
<td>10</td>
<td>34</td>
</tr>
<tr>
<td>context AND ontology AND ‘rule engine’</td>
<td></td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>context AND pattern AND ‘rule engine’</td>
<td></td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>ontology AND pattern AND ‘rule engine’</td>
<td></td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>3 TERMS Boolean OR</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>Sum=</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>context OR ontology OR pattern</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>context OR ontology OR ‘rule engine’</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>context OR pattern OR ‘rule engine’</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ontology OR pattern OR ‘rule engine’</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3 TERMS Boolean OR ‘CV’</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Sum=</th>
</tr>
</thead>
<tbody>
<tr>
<td>context OR ontology OR pattern</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>context OR ontology OR ‘rule engine’</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>context OR pattern OR ‘rule engine’</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>ontology OR pattern OR ‘rule engine’</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3 TERMS Boolean OR ‘PEOPLE…CV’</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Sum=</th>
</tr>
</thead>
<tbody>
<tr>
<td>context OR ontology OR pattern</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>19</td>
</tr>
<tr>
<td>context OR ontology OR ‘rule engine’</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>4</td>
<td>6</td>
<td>19</td>
</tr>
<tr>
<td>context OR pattern OR ‘rule engine’</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>ontology OR pattern OR ‘rule engine’</td>
<td>3</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>6</td>
<td>28</td>
</tr>
<tr>
<td>4 TERMS Boolean AND</td>
<td>B</td>
<td>A</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>---------------------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>context AND ontology AND pattern AND ‘rule engine’</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4 TERMS Boolean AND + ‘CV’</th>
<th>B</th>
<th>A</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Sum=</th>
</tr>
</thead>
<tbody>
<tr>
<td>context AND ontology AND pattern AND ‘rule engine’</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4 TERMS Boolean AND + ‘PEOPLE…CV’</th>
<th>B</th>
<th>A</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Sum=</th>
</tr>
</thead>
<tbody>
<tr>
<td>context AND ontology AND pattern AND ‘rule engine’</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4 TERMS Boolean OR</th>
<th>B</th>
<th>A</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Sum=</th>
</tr>
</thead>
<tbody>
<tr>
<td>context OR ontology OR pattern OR ‘rule engine’</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4 TERMS Boolean OR + ‘CV’</th>
<th>B</th>
<th>A</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Sum=</th>
</tr>
</thead>
<tbody>
<tr>
<td>context OR ontology OR pattern OR ‘rule engine’</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4 TERMS Boolean OR + ‘PEOPLE…CV’</th>
<th>B</th>
<th>A</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Sum=</th>
</tr>
</thead>
<tbody>
<tr>
<td>context OR ontology OR pattern OR ‘rule engine’</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>18</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
8.3 Results of the search in DBLP

Remarks
The tests were performed on November 20, 2009.

A = Method used to perform the searching

B = Query terms used.

Meaning of 1 and 0 in the table:
1: some people(authors) were found for this query using DBLP.
0: no results were returned by DBLP for this query
## Appendix 3

<table>
<thead>
<tr>
<th>Terms</th>
<th>Boolean AND</th>
<th>Method 1</th>
<th>Method 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>2 TERMS</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>Method 1</td>
<td>Method 2</td>
</tr>
<tr>
<td>context AND ontology</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>context AND pattern</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>context AND ‘rule engine’</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ontology AND pattern</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ontology AND ‘rule engine’</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>pattern AND ‘rule engine’</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td><strong>3 TERMS</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>Method 1</td>
<td>Method 2</td>
</tr>
<tr>
<td>context AND ontology AND pattern</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>context AND ontology AND ‘rule engine’</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>context AND pattern AND ‘rule engine’</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ontology AND pattern AND ‘rule engine’</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td><strong>4 TERMS</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>Method 1</td>
<td>Method 2</td>
</tr>
<tr>
<td>context AND ontology AND pattern AND ‘rule engine’</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
## 2 TERMS
### Boolean OR

<table>
<thead>
<tr>
<th>B</th>
<th>A</th>
<th>Method1</th>
<th>Method 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>context OR ontology</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>context OR pattern</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>context OR ‘rule engine’</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ontology OR pattern</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ontology OR ‘rule engine’</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>pattern OR ‘rule engine’</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

## 3 TERMS
### Boolean OR

<table>
<thead>
<tr>
<th>B</th>
<th>A</th>
<th>Method1</th>
<th>Method 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>context OR ontology OR pattern</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>context OR ontology OR ‘rule engine’</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>context OR pattern OR ‘rule engine’</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ontology OR pattern OR ‘rule engine’</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

## 4 TERMS
### Boolean OR

<table>
<thead>
<tr>
<th>B</th>
<th>A</th>
<th>Method1</th>
<th>Method 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>context OR ontology OR pattern OR ‘rule engine’</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
8.4 Query expansion and search in DBLP

Remarks
The tests were performed on November 23, 2009.

A = Method used to perform the searching

B = Query terms used.

Meaning of 1 and 0 in the table:
1: some people(authors) were found for this query using DBLP.
0: no results were returned by DBLP for this query
### 3 TERMS
**Boolean AND**

<table>
<thead>
<tr>
<th>B</th>
<th>Method1</th>
<th>Method2</th>
</tr>
</thead>
<tbody>
<tr>
<td>circumstance AND ontology AND pattern</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>context AND conceptualization AND pattern</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>context AND ontology AND design</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>circumstance AND ontology AND <code>rule engine</code></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>context AND conceptualization AND <code>rule engine</code></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>context AND ontology AND software</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>circumstance AND pattern AND <code>rule engine</code></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>context AND design AND <code>rule engine</code></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>context AND pattern AND software</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>conceptualization AND pattern AND <code>rule engine</code></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ontology AND design AND <code>rule engine</code></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ontology AND pattern AND software</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

---

### 4TERMS
**Boolean AND**

<table>
<thead>
<tr>
<th>B</th>
<th>Method1</th>
<th>Method2</th>
</tr>
</thead>
<tbody>
<tr>
<td>circumstance AND ontology AND pattern AND <code>rule engine</code></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>context AND conceptualization AND pattern AND <code>rule engine</code></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>context AND ontology AND design AND <code>rule engine</code></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>context AND ontology AND pattern AND <code>software</code></td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
### Appendix 4

#### 3 TERMS

<table>
<thead>
<tr>
<th></th>
<th>Method1</th>
</tr>
</thead>
<tbody>
<tr>
<td>circumstance OR ontology OR pattern</td>
<td>1</td>
</tr>
<tr>
<td>context OR conceptualization OR pattern</td>
<td>1</td>
</tr>
<tr>
<td>context OR ontology OR design</td>
<td>1</td>
</tr>
<tr>
<td>circumstance OR ontology OR ‘rule engine’</td>
<td>0</td>
</tr>
<tr>
<td>context OR conceptualization OR ‘rule engine’</td>
<td>0</td>
</tr>
<tr>
<td>context OR ontology OR software</td>
<td>1</td>
</tr>
<tr>
<td>circumstance OR pattern OR ‘rule engine’</td>
<td>0</td>
</tr>
<tr>
<td>context OR design OR ‘rule engine’</td>
<td>0</td>
</tr>
<tr>
<td>context OR pattern OR software</td>
<td>1</td>
</tr>
<tr>
<td>conceptualization OR pattern OR ‘rule engine’</td>
<td>0</td>
</tr>
<tr>
<td>ontology OR design OR ‘rule engine’</td>
<td>0</td>
</tr>
<tr>
<td>ontology OR pattern OR software</td>
<td>1</td>
</tr>
</tbody>
</table>

#### 4 TERMS

<table>
<thead>
<tr>
<th></th>
<th>Method1</th>
</tr>
</thead>
<tbody>
<tr>
<td>circumstance OR ontology OR pattern OR ‘rule engine’</td>
<td>0</td>
</tr>
<tr>
<td>context OR conceptualization OR pattern AND ‘rule engine’</td>
<td>0</td>
</tr>
<tr>
<td>context OR ontology OR design OR ‘rule engine’</td>
<td>0</td>
</tr>
<tr>
<td>context OR ontology OR pattern OR ‘software’</td>
<td>1</td>
</tr>
</tbody>
</table>
8.5 Example of execution of query in DBLP

Remarks
The tests were performed on November 27, 2009.
Figure 42: query ‘formal semantic model’ in DBLP

Figure 43: query ‘model semantic formal’ in DBLP
8.6 Troux’s Enterprise Model used to test the software
Figure 44: Troux's Enterprise Model used to test the software