Francisco António Gonçalves Cavaco

Licenciado em Ciências de Engenharia

Ontologies Learn by Searching

Dissertation to obtain the Master degree in Electrical Engineering
and Computer Science

Orientador: Ricardo Luís Rosa Jardim Gonçalves

Professor Auxiliar, Departamento de Engenharia Electrotécnica

Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Co-orientador: João Filipe dos Santos Sarraipa, Investigador, UNINOVA

Júri:

Presidente: Prof. Doutor José António Barata Oliveira
Arguente: Prof. Doutor João Pedro Mendonça da Silva
Vogais: Prof. Doutor Ricardo Luís Rosa Jardim Gonçalves
Prof. Doutor Hervé Panetto
MSc. João Filipe dos Santos Sarraipa

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro 2011

Copyright

Aos meus pais,

ABSTRACT

Due to the worldwide diversity of communities, a high number of ontologies representing the same segment of reality which are not semantically coincident have appeared. To solve this problem, a possible solution is to use a reference ontology to be the intermediary in the communications between the community enterprises and to outside. Since semantic mappings between enterprise's ontologies are established, this solution allows each of the enterprises to keep internally its own ontology and semantics unchanged. However information systems are not static, thus established mappings become obsoletes with time. This dissertation's objective is to identify a suitable method that combines semantic mappings with user's feedback, providing an automatic learning to ontologies & enabling auto-adaptability and dynamism to the information systems.

KEYWORDS

The keywords of this dissertation are: Ontology, Semantic mapping, Semantic Interoperability, Dynamic Information Systems, Complex Systems.

RESUMO

Devido à existência duma diversidade de comunidades em todo o mundo, um número elevado de ontologias que representam o mesmo segmento da realidade não sendo semanticamente coincidentes têm surgido. Para resolver o problema, uma solução possível é usar uma ontologia de referência como intermediária nas comunicações entre as empresas de uma comunidade e para com o exterior. Desde que se estabelecem mapeamentos semanticos entre as ontologias das empresas, esta solução permite que cada uma das empresas mantenha internamente a sua própria ontologia e semântica inalterada. Contudo os sistemas de informação não são estáticos, logo mapeamentos estabelecidos tornam-se obsoletos com o tempo. O objetivo desta dissertação é então identificar método que combine os mapeamentos semanticos com o feedback de utlizadores, proporcionando uma aprendizagem automática às ontologias & obtendo assim alguma capacidade de auto-adaptabilidade e dinamismo aos sistemas de informação.

PALAVRAS-CHAVE

As palavras-chave desta dissertação são: Ontologias, Mapeamentos semânticos, Interoperabilidade semântica, Sistemas de Informação Dinamicos, Sistemas complexos.

ACKNOWLEDGEMENTS

The author wishes to express his sincere thanks and appreciation to all people that helped him during his studies and during the realisation of this dissertation.

To his advisor Professor Ricardo Gonçalves for giving him the honour to work together, for all the comments and suggestions during the development of this work which were essential to be succeeded.

To João Sarraipa for being there every day, for his attention, guidance and support during this research and the preparation of this dissertation.

And finally to his parents and his family who always believed him and supported him in everything they could. You were always there and the author will never forget it.

LIST OF ACRONYMS

AI - Artificial Intelligence

ANN - Artificial Neural Networks

ATL - Atlas Transformation Language

BN - Bayesian Network

CPT - Conditional Probability Table

DAG - Directed Acyclic Graph

EM - Expectation-Maximization

FL - Fuzzy Logic

GUI - Graphical User Interface

HTML - HyperText Markup Language

IDE - Integrated Development Environment

IPFP - Iterative Proportional Fitting Procedure

JSP - JavaServer Pages

KNN - Nearest Neighbour Algorithm

KRE - Knowledge Representation Element

KRRM - Knowledge Representation Requirements Model

LCIM - Levels of Conceptual Interoperability Model

LO - Learning Ontology

MLN - Markov Logic Network

MO - Mediator Ontology

OWL - Web Ontology Language

PC - Personnel Computer

RDF - Resource Description Framework

SME - Small and Medium Enterprises

SVM - Support Vector Machines

URL - Uniform Resource Locator

XML - Extensible Markup Language

TABLE OF CONTENTS

TABLE OF FIGURES

1. INTRODUCTION

The formation of cooperation and collaboration alliances between several small organizations is proving, in multiple cases, to be more efficient and competitive by comparison with big companies. Manufacturing processes span nowadays across borders as result of the globalization markets pressure, thus research on manufacturing management has turned from an intra-enterprise to an inter-enterprise focus.

Consequently there is a growing interest in electronic business (e-business) solutions to facilitate information sharing between organizations in the supply chain. However, due to the worldwide diversity of communities, a high number of knowledge representation elements, such as ontologies which are not semantically coincident, have appeared representing the same segment of reality. Thus, even operating in the same domain, enterprises do not understand each other, making the seamless communication among various systems and applications more difficult and sometimes impracticable.

As a result, there is a demand for intelligent solutions capable of reinforcing partnerships and collaborations between enterprises, which needs to be able to maintain and manage its information systems dynamically. Thus, author proposes the increase of Interoperability of industrial information systems domain by the use of a knowledge–based methodology able to build a reference ontology and to provide its maintenance through the system users feedback.

1.1. Motivation

Nowadays the world is in a constant change, which requires, dynamic information systems. To deal with these systems there is an important need to implement solutions able to intelligently evolve and adapt themselves to external feedback as from user's interactions.

With a large amount of information that can exceed the user capability to find what he really searches for, there is a need of a tool to find what the user is looking for and to learn from the user searching patterns. Thus, this dissertation intends to contribute to solve semantic interoperability problems by proposing a methodology & platform able to interact with humans, learning from them and helping them.

1.2. Research Method

The used research method on this dissertation was reasoned on the classical research method [1] which consists on seven steps. This method starts with the study of a problem and ends with the analysis of the obtained results. If these results are not the desired ones, this method provides the possibility to go back to the first steps and try again a new approach. In Figure 1.1 it is possible to see these referred seven steps.

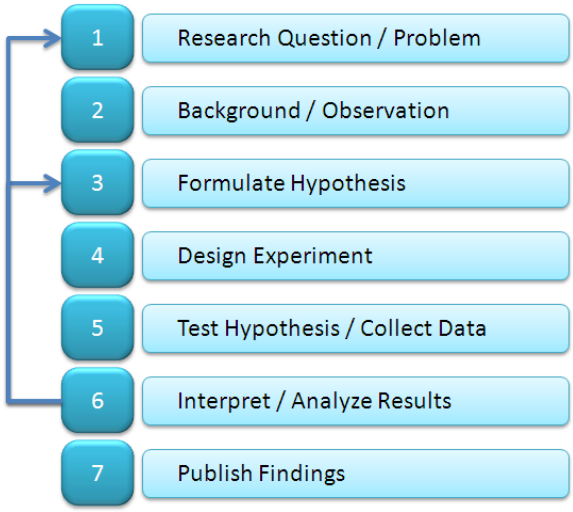

Figure 1.1: Phases of the Classical Research Method [1]

A short description of each step of Figure 1.1 followed by the author is described below:

1. **Research Questions / Problem:** This is the most important step in a research. In this step the area of interest is defined by the research question, which may have several secondary questions to better define its the main idea. The research question must be clearly defined in order to make it able to be confirmed or refuted. The author's research questions that conducted to this dissertation are:

   - How to enhance the knowledge acquisition from information system's users?
   - How to improve ontology based systems to facilitate its intelligence increase?

2. **Background / Observation:** The state of the art research is done in this step studying previously similar works, presenting literature review and previously projects in order to have a start point for the dissertation. Showing existing ideas

of other authors will identify the most suitable solutions, prototypes to be developed for this dissertation.

All humans when thinking about some concept can create its mental description, which can be different from person to person. This mental description depends on the importance that description has in our mind and that can change as we want. Is around these ideas to which the author wants to explore on the specification of the intelligent information system able to adopt its concepts and relations.

There are already some techniques able to improve intelligent systems, such as fuzzy logic, artificial neural networks, machine learning, etc. To represent the concepts, a knowledge base, ontologies may be used. They are well suit for information sharing when their information domains are related to a particular area of knowledge [2].

In this dissertation it is intended to build an ontology learn by searching from user's usability, maintaining the knowledge base updated with the corresponding semantic mappings. More details about this are described in section 2 - Literature Review.

3. **Formulate Hypothesis:** This step is where the predictions for the results of the research work are worked out. A hypothesis must be simple to understand, specific, conceptually clear and measured. The author's hypothesis for this dissertation are:

    - An ontology based framework integrated with proper operational research methods would facilitate the knowledge acquisition from user's feedback and would increase the intelligence of information systems management.
    - Maintaining updated the mappings established between ontologies would enhance knowledge re-use and adaptation.

4. **Design Experiment:** It is in this step where it is possible to find the detailed plan of the experimental phase steps. It first gives an approach to the used technologies in chapter 4 and then in chapter 5 it presents the architecture that represents the proof-of-concept for the proposed dissertation.

5. **Test Hypothesis:** To test the hypothesis and get the results, the developed prototype must be tested running under different scenarios. In chapter 6 there are tests, whose results are analysed against the hypothesis.

6. **Interpret / Analyse Results:** This step is where the achieved results are analysed against the hypothesis. If the results are not the expected ones, conclusions can be discussed. After the results discussion, it is also needed to consider next steps, giving some recommendations for further research. This step is represented by the chapter 7 “Conclusions and Future Work”.

7. **Publish Findings:** If the research results end up in a valuable contribution to the research community, it should be presented in order to share author’s ideas. This can be made as scientific papers, in conferences, Journals and so on. The author published one scientific paper that represents his conceptual contribution to this dissertation issue.

1.3. Dissertation Outline

The first section of this dissertation, the *Introduction*, states the main ideas that conducted to the study for this research project. Then, the *Literature Review* chapter intends to present the state of the art research conducted, containing a resume of the inspirational literature used to support the implemented system functionalities, aligned with the background observation. Chapter *Knowledge based methodology for semantic interoperability* gives an overview where this dissertation proposed system will be embedded. This also presents how such proposed system can work with MENTOR which is a methodology to build a reference ontology of a community of enterprises that consequently together can work to improve interoperability in manufacture domain of a set of enterprises. Then all the architecture of both systems together is presented. In chapter Reference tools & models for ontology management theoretical implementation is analysed, presenting the used technologies in oLEARCH implementation. In chapter 5 it is presented oLEARCH architecture including a description of the implemented oLEARCH services. Afterwards, in chapter 6, the implemented architecture is tested and compared with formulated hypothesis to check its validation. At the end, in *Conclusions and Future Work* chapter is where this dissertation conclusions and future work topics are presented.

2. LITERATURE REVIEW

The Literature Review chapter intends to be a synthesis of the state of the art related to the dissertation areas and concepts.

2.1. Artificial Intelligence and Neuroscience

Artificial intelligence was formally initiated in 1956. Its conception's goal was to understand and emulate a biological brain in a computer simulation. Artificial intelligence attempts to build intelligent systems as well as understand them. There are several definitions for artificial intelligence which can be organized in four categories [3]: Systems that think like humans; Systems that act like humans; Systems that think rationally; Systems that act rationally. A system is rational if it does the right thing. The aim of computational neuroscience is to explain how electrical and chemical signals are used in the brain to represent and process information [4].

Neuroscience area is extremely complex, composed by many subfields [5]. Knowing how neural systems processes all the gathered information is essential to understand human brain, giving the opportunity for AI to develop not only in artificial neural networks but also in other areas where there are a need to implement learning systems.

Research in neuroscience may provide to AI some opportunities to expand. Computational neuroscience is the study of brain function in terms of the information processing properties of the structures that make up the nervous system. It is an interdisciplinary science that links the diverse fields of neuroscience, cognitive science and psychology with electrical engineering, computer science, mathematics and physics [6].

Semantic memory (also called conceptual knowledge) is one of the aspects of human memory that corresponds to general knowledge representation of objects, word meanings, facts and people, without connection to any particular time or place [7]. This provides to humans the ability to, given a concept, be able to create its mental visualization and also to see the most important associations that the individual has to the referred concept. Although this kind of memory may depend on individual's experience it is mostly shared in a given culture.

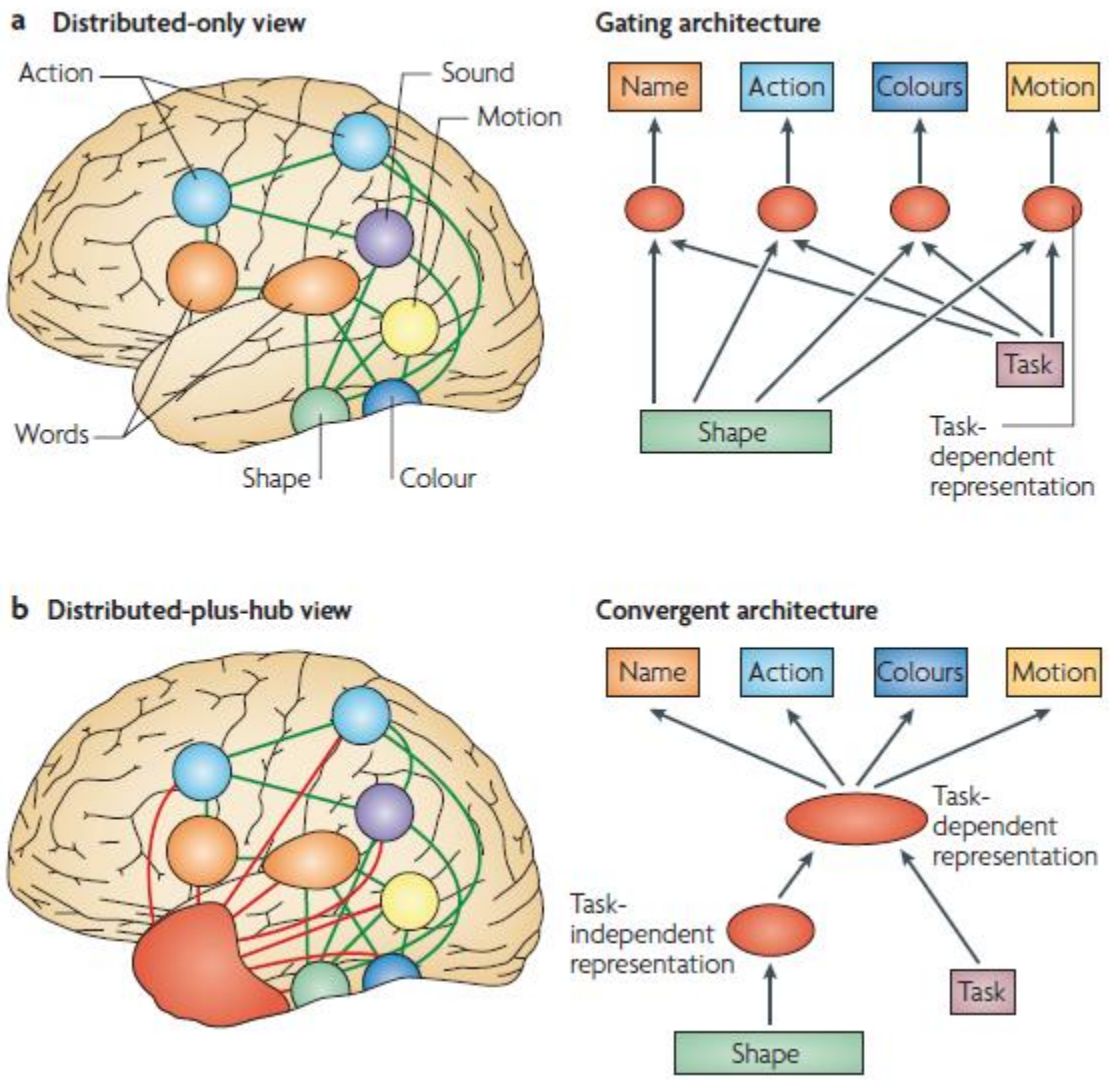

Figure 2.1: Two theoretical positions regarding the neuroanatomical distribution of the cortical semantic network and schematic models based on these views [8].

There are many theories supporting neural basis of semantic memory (Figure 2.1). This distributed-only view proposes that a semantic network is composed by these regions (Shape, Action, Sound, Colour, Words, and Motion) along with all the diverse connections between them, represented by the green lines. However, there are several cases of individuals whom suffered brain damages and could recover, probably because the entire neural basis of semantic memory might have a different approach. Figure 2.1b suggests that, in addition to distributed-only view proposed representation, there are connections (shown as red lines) between the various shapes representations and a modal hub. It's in this hub where the associations between different pairs of attributes (such as shape and name or shape and action, etc.) are processed. Figure's right-hand side shows the corresponding convergent architecture of these views [8].

Nowadays neuroscience has already enough knowledge regarding how parts of the

brain work. Some of this knowledge, like connections, internal connectivity and what is represented by neuronal activity of each brain region, biophysical properties of single neurons [9] and the effect of lesions, all might provide a solid base for computational understanding of brain function regarding neuronal network operations of each region [10]. Learning mechanisms are, in addition, important to understand how brain processes the information. Is by modifying the synaptic connection strengths (or weights) between neurons that useful neuronal information processors for most brain functions, including perception, emotion, motivation, and motor function, are built [11].

Accordingly to this, Hebbian learning [12] stated that simultaneous activation of cells leads to increases in synaptic strength between those cells. These weights are adjusted in order to better represent the relationship between two cells. Transporting it into artificial neural networks, it can be a method of determining how to increase/decrease weights between two neurons.

2.1.1. Conclusions

This research lead to better understand how human brain works and how can it be represented in artificial intelligence. Using Figure 2.1b as inspiration for semantic representation, the author proposes in this dissertation MEDIATOR as a modal hub to create the associations between possible new concepts or possible new patterns and a reference ontology for the other connections. For this dissertation both, MEDIATOR and reference ontologies, are representing the knowledge base. To simulate strength's connections between neurons the author will associate weights into each concept and into its conceptual relations, therefore stronger relations will have a higher weight.

2.2. Human Based Learning Techniques

Although computers are getting faster, they are still not fast enough to perform all the parallel operations of human brain at the same time. Discover brain's algorithm is one of the major research objectives in computational neuroscience. Next are presented two human based learning techniques, Fuzzy Logic and Neural Networks.

2.2.1. Fuzzy Logic

The concept of Fuzzy Logic (FL) was conceived by Lotfi Zadeh, a professor at the University of California at Berkley. He reasoned that people do not require precise, numerical

information input, and yet they are capable of highly adaptive control. If feedback controllers could be programmed to accept noisy, imprecise input, they would be much more effective and perhaps easier to implement. The increasing level of information that can be obtained on a system is essential for the improvement and the control of this system as well as the processes linked with it. Human expert knowledge and acquisition by sensors directly on the system, allow to access to this part of Knowledge. Linking these two sources which are as different as complementary must contribute to a more complete and coherent knowledge modelling, allowing a stronger integration of the system processes [13].

Unfortunately, U.S. manufacturers have not been so quick to embrace this technology while the Europeans and Japanese have been aggressively building real products around it. FL is a problem-solving control system methodology that lends itself to implementation in systems ranging from simple, small, embedded micro-controllers to large, networked, multi-channel PC or workstation-based data acquisition and control systems. It can be implemented in hardware, software, or a combination of both. FL provides a simple way to arrive at a definite conclusion based upon vague, ambiguous, imprecise, noisy, or missing input information. It incorporates a simple, rule-based IF X AND Y THEN Z approach to a solving control problem rather than attempting to model a system mathematically. Ex: terms like "IF (process is too cool) AND (process is getting colder) THEN (add heat to the process)".

FL requires some numerical parameters in order to operate such as what is considered significant error and significant rate-of-change-of-error, but exact values of these numbers are usually not critical unless very responsive performance is required in which case empirical tuning would determine them. There are several unique features that make FL a particularly good choice for many control problems. It is inherently robust since it does not require precise, noise-free inputs and can be programmed to fail safely if a feedback sensor quits or is destroyed. The output control is a smooth control function despite a wide range of input variations. Since the FL controller processes user-defined rules governing the target control system, it can be modified and tweaked easily to improve or drastically alter system performance. New sensors can easily be incorporated into the system simply by generating appropriate governing rules. Also, any sensor data that provides some indication of a system's actions and reactions is sufficient. This allows the sensors to be inexpensive and imprecise thus keeping the overall system cost and complexity low.

FL can control nonlinear systems that would be difficult or impossible to model mathematically. This opens doors for control systems that would normally be deemed unfeasible for automation. When there are a reasonable number of inputs and outputs for a single implementation, defining the rule base can become complex since rules defining their

interrelations must also be defined. In these cases it is better to break the control system into smaller chunks and use several smaller FL controllers distributed on the system, each with more limited responsibilities.

An important property of FL is the membership function, which is a graphical representation of the magnitude of participation of each input. It associates a weighting with each of the inputs that are processed, define functional overlap between inputs, and ultimately determines an output response. The rules use the input membership values as weighting factors to determine their influence on the fuzzy output sets of the final output conclusion. Once the functions are inferred, scaled, and combined, they are defuzzified into a crisp output which drives the system. There are several applications of FL. In [13] it is presented the improvement of a defect recognition system for wooden boards by using knowledge integration from two expert fields, wood expertise and industrial vision expertise. In this article, a numeric model of wood defect recognition was created according to a tree structure where each inference engine is a fuzzy rule based. The aim of this work was to obtain a generic model to recognize the defects, reusable and reproducible independently of the user's experience.

Fuzzy Logic provides a completely different, unorthodox way to approach a control problem. This method focuses on what the system should do rather than trying to understand how it works. One can concentrate on solving the problem rather than trying to model the system mathematically, if that is even possible. This almost invariably leads to quicker, cheaper solutions. Once understood, this technology is not difficult to apply and the results are usually quite surprising and pleasing.

2.2.2. Artificial Neural Networks

An Artificial Neural Network (ANN) [14] is an information processing paradigm that is inspired by the way biological nervous system, such as brain, process information. An ANN is like people, learns by example. It is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to synaptic connections that exist between the neurones. This is also valid for ANNs.

Neural Networks, with their remarkable ability to drive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by humans or other computer techniques. A trained neural network can be thought of as an expert in the category of information to what it was given to analyse. Other advantages that ANNs have are:

Adaptative learning (an ability to learn how to do tasked based on the data given); Self Organization (can create its own organisation or representation of the information it receives during leaning time); Real time operation (ANN computations may be carried out in parallel); Fault tolerance via Redundant Information Coding (Partial destruction of a neural network leads to the corresponding degradation of performance. However, some networks capabilities may be retained even with major network damage. While conventional computers use an algorithmic approach, i.e. the computer follows a set of instructions in order to solve a problem. They cannot solve a problem that the specific steps are unknown.

ANN and conventional algorithmic computers are not in competition but complement each other. There are tasks more suited to an algorithmic approach like arithmetic operations and tasks more suited to neural networks. Even more, there are a large number of tasks that required the combination of the two approaches.

A neural network is a directed graph consisting of nodes with interconnecting synaptic and activation links [15]. It is characterized by four properties:

- Each neuron is represented by a set of linear synaptic links, an externally applied bias, and a possibly nonlinear activation link. The bias is represented by a synaptic link connected to an input fixed at +1.

The synaptic links of a neuron weight their respective input signals.

- The weighted sum of the input signals defines the inducted local field of the neuron in question.
- The activation link squashes the induced local field of the neuron to produce an output.

The issue of knowledge representation in a neural network is directly related to that of network architecture. Unfortunately, there is no well-developed theory for optimizing the architecture of a neural network required to interact with an environment of interest, or for evaluating the way in which changes in the network architecture affect the representation of knowledge inside the network. Indeed, satisfactory answers to these issues are usually found through an exhaustive experimental study for a specific application of interest, with the designer of the neural network becoming an essential part of the structural learning loop.

The learning process can be categorized by two simple characteristics: learning with a teacher (also referred to as supervised learning); and learning without a teacher. These different forms of learning are also performed by ANN.

From the various types of ANN the author gives some special attention to feed forward neural networks (which can return back the output to the input, thereby giving rise to an iteration process) because of its similarity (in this case) to the system to which this technology could be applied.

To finalise, ANN are one of the best machine learning technologies to identify patterns or trends in data, they are well suited for prediction or forecasting needs including sales forecasting, industrial process control, customer research, data validation, risk management, etc.

2.2.3. Conclusions

The presented learning techniques fit in different applications. Fuzzy Logic is widely used in systems control due to its human mimic in decision make. In contrast, ANN is based on the thinking process of human brain. Its learning process involves learning algorithms and training data.

There are no fuzzy elements in the dissertation proposed architecture, thus it is not applied. By other side, due to ANN learning properties, it could have been used in the architecture as algorithm to find the highest semantic similarities between concepts.

2.3. Ontologies

The human reasoning organizes his knowledge of the world in a tree structure. This is the base's principle of an ontology. They support the interoperability between systems, improving the organisation of information, its control and understanding [16][17].

"An ontology is a formal, explicit specification of a shared conceptualisation." [18]. It provides a vocabulary that describes a domain of interest and a specification of the meaning of terms used in the vocabulary [19]. In other words, they represents a knowledge representation used to capture information and knowledge about a subject, generally within the structure of a semantic network, consisting of a diagram composed of nodes and arcs [20]. We can define a class hierarchical ontology representing concepts, objects or entities characterized by their properties [21].

By defining shared and common domain theories, ontologies help both people and machines to communicate concisely, supporting the exchange of semantics and not only syntax [22]. Unfortunately, there is no universal solution to build an ontology and designers of ontologies themselves apply different views of the same domain during ontology

development. This yields semantic heterogeneity at ontology level, which is one of main obstacles to semantic interoperability [23].

Ontology matching is a promising solution to the semantic heterogeneity problem [24]. Its main purpose is to find correspondences among entities from different ontologies. Ontology matching has emerged as a crucial step when information sources are being integrated, such as when companies are being merged and their corresponding knowledge bases are to be united [25].

Ontology maintenance pertains to how to organise, search, and update existing ontologies [27]. The enormous number of semantic Web pages and ontologies makes manual ontology maintenance a daunting task. As a result, automated solutions should be explored for ontology integrations and mapping. Ontology learning appears to be an attractive approach to this goal [27].

2.3.1. Ontology Learning

Ontology learning seeks to discover ontological knowledge from various forms of data automatically or semi-automatically [27]. It provides the ability to not only discover ontological knowledge at a larger scale and a faster pace, but also mitigate human-introduced biases and inconsistencies. Moreover, ontology learning can support refining and expanding existing ontologies by incorporate new knowledge [27].

To facilitate ontology construction and discover ontological knowledge, machine learning is commonly used [27][28]. The majority of ontology learning techniques are unsupervised because training data annotated with ontological knowledge are commonly not available, resulting in a higher employment of statistical techniques [27].

2.3.2. Conclusions

To structure data in a way that machines can handle, ontologies can be used. Its management can be assisted by knowledge acquisition together with machine-learning techniques [27]. This association leads to the appearance of Ontology Learning which, although it is not yet a full automatic machine knowledge acquisition, it still can be a powerful tool to assist in the management of the ontologies.

2.4. Operations research methods

Operations research methods aim to aid in making decisions by providing the needed

quantitative information based on a scientific method of analysis [26][29].

Operations research is an interdisciplinary mathematical science, as it employs techniques from other mathematical sciences, such as mathematical modelling, statistical analysis, and mathematical optimization, or focuses on the effective use of technology by organizations, providing solutions to complex decision-making problems [30]. A problem in the real world is modelled, usually in mathematical terms, then mathematical techniques, together with data analysis and computational algorithms are applied, looking for optimization [31].

Born before World War II, its origin comes from military efforts. Nowadays, operational research encompasses a wide range of problem-solving techniques and methods applied in the pursuit of improved decision-making and efficiency [32]. Operations Research is well adapted to such decision making in business. Applications into business management have been discussed in different disciplines such as Management Science, Operations Management, Logistics Management, Supply Chain Management, and Decision Sciences. It set up and uses mathematical models, usually related to questions of planning in business, industry, or management [33].

2.4.1. Markov Chains

Markov logic is a novel language that provides the capability of joining in the same representation probabilistic graphic models (Markov networks) to handle uncertainty, and first-order logic to handle complexity. By attaching weights to first-order logic formulas (the higher the weight, the bigger is the difference between a world that satisfies the formula and one that does not), a Markov Logic Network (MLN) can be built. This network can be used as a template to construct Markov networks, providing the full expressiveness of probabilistic graphical models and first-order logic [34].

Given a set of constants (i.e., individuals) of the domain and an interpretation, the groundings of the formulas in an MLN can generate a Markov network by adding a variable for each ground atom, an edge if two ground atoms appear in the same formula, and a feature for each grounded formula. The probability distribution of the network is defined as

$$P(X = x) = \frac{1}{Z} exp\left(\sum_{i=1}^{F} w_i n_i(x)\right),$$

where F is the number of formulas in the MLN, ni(x) is the (binary) number of true groundings of Fi in the world x, Wi is the weight of Fi, and Z is a normalizing constant.

Formulas' weights can be learned generatively from example data by maximizing the pseudo-log-likelihood [35] of that data, while efficient inference can be done using approximate inference algorithms, such as the MC-SAT [35].

There are several areas where Markov Process has an important rule, such as in statistic, marketing, genetic, computer vision, diagnostic and troubleshooting, software debugging, speech recognition and understanding algorithms, internet, musical composition, etc [36].

Even not knowing, everyday millions of internet users use Markov in some navigation patterns and especially in Google "RankPage". The PageRank of a webpage is the probability to be at page i in the stationary distribution on the following Markov chain on all (known) webpages [37]. If N is the number of known webpages, and a page i has ki links then it has transition probability $\frac{\alpha}{k_i} + \frac{1-\alpha}{N}$ for all pages that are linked to and $\frac{1-\alpha}{N}$ for all pages that are not linked to. The parameter α is taken to be about 0.85. Markov models have also been used to analyze web navigation behaviour of users. A user's web link transition on a particular website can be modelled using first- or second-order Markov models and can be used to make predictions regarding future navigation and to personalize the web page for an individual user.

Another interesting application of Markov networks is about to build a probabilistic scheme for ontology matching, called iMatch [38]. First, it uses undirected networks, which better supports the non-causal nature of the dependencies. Second, it handles the high computational complexity by doing approximate reasoning, rather than by ad-hoc pruning. Third, the probabilities that it uses are learned from matched data. Finally, iMatch naturally supports interactive semiautomatic matches. iMatch uses Markov Networks rather than Bayesian Networks since there is no inherent causality in ontology matching .

2.4.2. Bayesian Networks

A Bayesian network (BN) is a directed acyclic graph with attached local probability distributions [39]. Nodes in the graph represent random variables (corresponding to attributes, features etc.). Each random variable has a mutually exclusive and exhaustive set of values (states). Edges in the graph represent direct interdependences between two random variables. Bayesian networks consist of two sort of knowledge:

- qualitative knowledge that describes interdependencies by means of directed graph;

- quantitative knowledge that captures relations among random variables by means of Conditional Probability Tables (CPTs). An advantage of BNs, compared to other uncertainty representation formalisms, is the possibility to model complicated mutually related phenomena in quite a tractable way.

Thus, BNs provide a means of capturing existing knowledge about a domain, learning the stochastic properties of that domain and thereby adjusting its model. This adjust can be related to the ontologies mapping establishment.

BNs are currently being exploited also for estimating effects of different types of behaviour and as support for human or automated decision tasks. Some sample applications include using BNs to reduce power consumption of machines with reference to user behaviour [40], to diagnose faults in industrial processes [41] or to monitoring and manipulating cause and effects for modelled systems as disparate as the weather, disease and mobile telecommunications networks [42].

2.4.3. BayesOWL

BayesOWL is a framework which augments and supplements OWL for representing and reasoning with uncertainty based on Bayesian networks [43][44]. This framework consists of three key components: 1) a representation of probabilistic constraints as OWL statements; 2) a set of structural translation rules and procedures that converts an OWL taxonomy ontology into a BN directed acyclic graph (DAG); and 3) a method SD-IPFP based on 'iterative proportional fitting procedure' (IPFP) that incorporates available probability constraints into the conditional probability tables (CPTs) of the translated BN. The translated BN, which preserves the semantics of the original ontology and is consistent with all the given probability constraints, can support ontology reasoning, both within and cross ontologies, as Bayesian inferences, with more accurate and more plausible results.

OWL is first augmented to allow additional probabilistic mark-ups so that probability values can be attached to individual concepts in an ontology. Secondly, a set of structural translation rules is defined to convert this probabilistically annotated OWL ontology taxonomy into a directed acyclic graph (DAG) of a BN. Finally, the BN is completed by constructing conditional probability tables (CPTs) for each node in the DAG.

For instance, in the Fenz et al. [45] research work, the objective of the Bayesian network is to determine asset-specific threat probabilities by taking asset-specific influence factors into account. The advantage of the proposed Bayesian threat probability

determination is that it gives the risk manager a methodology to determine the threat probability in a structured and, by incorporating the security ontology, comprehensible way. However, the high dependence on realistic input values requires further research on sound methods to gather, store, and provide these crucial threat probability calculation components.

Although useful, the task of building the structure and assigning the probability distributions of a Bayesian Network is complex and knowledge-intensive. Bayesian Networks are notoriously difficult to build accurately and efficiently which has somewhat limited their application to real world problems. It requires the identification of relevant statistical variables in the application domain, the specification of dependency relations between these variables and assignment of initial probability distributions.

2.4.4. Conclusions

After this research it is possible to see how and where these methods can be applied. Bayesian Networks are a possible solution when it is needed to apply probabilistic to get a conclusion especially when the cause and effects of a system are modelled. One example is if there are no clouds in the sky then it can't rain.

For Markov chains nothing is invalid, just less probable. This is measured using weights, as explained before. Another characteristic is that the next state depends not on the previous states but only on the current one.

In the proposed architecture system knowledge is not represented as cause and effect, because sometimes there is no modelled relation between some concepts and in additional it could generate a cycle graph. Consequently, BN could not have been applied but Markov. Like ANN, Markov could have been used in the architecture as algorithm to find the highest semantic similarities between concepts. However, it was applied a simpler solution to facilitate the prototype implementation.

2.5. Machine Learning

Machine learning is a scientific discipline concerned with the design and development of algorithms that allow computers to evolve behaviours based on empirical data, such as from sensor data or databases [46]. It studies computer learning algorithms to do stuff. This learning must be automatically without human intervention or assistance. Machine learning paradigm can be viewed as "programming by example", learning to do better in the future, based on what was experienced in the past [47].

Recent advances in machine learning make possible to design efficient prediction algorithms for data sets with huge number of parameters. Machine learning procedure was used to produce a small set of classification rules that made two-thirds of correct predictions. Not only did these rules improve the success rate of the loan decisions, but the company also found them attractive because they could be used to explain to applicants the reasons behind the decision [48].

Machine learning market applications are endless. They can be applied to optical character recognition, face detections, spam filtering, topic spotting, spoken language understanding, medical diagnosis, fraud detection, weather prediction, etc.

There are some areas where machine learning techniques reached a level of performance equal or even greater than human experts. For example in astronomy, machine learning has been used to develop a fully automatic cataloguing system for celestial objects that are too faint to be seen by visual inspection. In chemistry, it has been used to predict the structure of certain organic compounds from magnetic resonance spectra [49].

Machine learning can be involved in ontology knowledge maintenance by applying its learning techniques. There are four basically different styles of learning techniques in use: Classification; Association; Clustering; and Numeric Prediction.

Classification

Classification learning, sometimes also called supervised, because, in a sense, the method operates under supervision by being provided with the actual outcome for each of the training examples [49]. Training data has to be specified what are trying to learn (the classes) [50].

Association learning

Data can be mined to identify associations. Introduced in 1993 [51] the task association rule mining has received a great deal of attention till nowadays where the meaning of such rules is still one of the most popular pattern discovery methods in knowledge discovery data [52]. Association rules differ from classification rules as they can predict not just the class but any attribute and more than one attribute's value at a time [49]. This is the reason for the higher existing number of association rules than classification rules. To avoid association rules swamp, they are often limited to those that apply to a certain minimum number of examples (say 80% of the dataset) and have greater than a certain minimum accuracy level (say 95% accurate) [49]. Association rules usually involve only nonnumeric attributes [49].

Ontology learning focuses on association learning [27]. The generalized association-rule-learning algorithm extends its baseline by aiming at descriptions at the appropriate

taxonomy level. For example, “snacks are purchased together with drinks” [28].

Clustering

Clustering technique is an unsupervised learning task to learn a classification from the data [50], usually applied to group items that seem to fall naturally together [49] instead of requiring a predefined classification [50]. The challenge is to find these clusters and assign the instances to them and to be able to assign new instances to the clusters as well [49]. Data items are grouped according to logical relationships or consumer preferences. For example, data can be mined to identify market segments or consumer affinities. It is a common technique for statistical data analysis [46].

Numeric Prediction

Numeric Prediction [49] is a variant of classification learning in which the outcome is a numeric value rather than a category. Linear regression is a natural technique to be considered. Its idea is to express the class (x) as a linear combination of the attributes (a_1, a_2,…, a_k), with predetermined weights (w_0, w_1, …, w_k):

$$x = w_0 + w_1 a_1 + w_2 a_2 + \cdots + w_k a_k$$

The weights are calculated from the training data.

Any regression technique, whether linear or non-linear can be used for classification. The trick is to perform a regression for each class, setting the output equal to one for training instances that belong to the class and zero for those that not. The result is a linear expression for the class. Then, given a test example of unknown class, calculate the value of each linear expression and choose the one that is largest [49].

Logistic regression attempts to produce accurate probability estimates by maximizing the probability of the training data. If the data can be separated perfectly into two groups using a hyperplane, it is said to be linearly separable and so there is a very simple algorithm for finding a separating hyperplane [49]. The algorithm is called perceptron learning rule, which is the grandfather of neural networks [49].

2.5.1. Instance-based learning

Instance-based learning is a kind of classification learning. In instance-based learning training examples are stored verbatim, and a distance function is used to determine which member of the training set is closest to an unknown test instance [49]. Once the nearest training instance has been located, its class is predicted for the test instance. Some varieties of instance-based learning deal only with ratio scales because they calculate the “distance”

between two instances based on the values of their attributes [49]. If the actual scale is ordinal, a numeric distance function must be defined [49].

Deriving suitable attributes weights from the training set is a key problem in instance-based learning [49].

Distance function

Although there are other possible choices, most instance-based learners use Euclidean distance. The distance between an instance with attribute values $a_1^{(1)}$, $a_2^{(1)}$,. . . , $a_k^{(1)}$ (where k is the number of attributes) and one with values $a_1^{(2)}$, $a_2^{(2)}$, . . . , $a_k^{(2)}$ is defined as

$$\sqrt{\left(a_1^{(1)} - a_1^{(2)}\right)^2 + \left(a_2^{(1)} - a_2^{(2)}\right)^2 + \ldots + \left(a_k^{(1)} - a_k^{(2)}\right)^2}.$$

When comparing distances it is not necessary to perform the square root operation; the sums of squares can be compared directly [49].

Nearest-Neighbour

In instance-based learning, each new instance is compared with existing ones using a distance metric, and the closest existing instance is used to assign the class to the new one. This is called the nearest-neighbour classification method. Sometimes more than one nearest neighbour is used, and the majority class of the closest k neighbour's (or the distance-weighted average, if the class is numeric) is assigned to the new instance. This is termed the k-nearest-neighbour method [49].The nearest neighbour algorithm (KNN) belongs to the class of pattern recognition statistical methods. The method does not impose a priori any assumptions about the distribution from which the modelling sample is drawn. It involves a training set with both positive and negative values. A new sample is classified by calculating the distance to the nearest neighbouring training case. The sign of that point will determine the classification of the sample. The performance of the KNN algorithm is influenced by three main factors: (1) the distance measure used to locate the nearest neighbours; (2) the decision rule used to derive a classification from the k-nearest neighbours; and (3) the number of neighbours used to classify the new sample [54].

KD-Tree

Nearest neighbours classification method can be found more efficiently by representing the training set as a tree, although it is not quite obvious how [49].There are many real-life problems which requires fast analyses and fast search in multidimensional data. It has the advantage that is easy to build and has a simple algorithm for closest points and ranged search. KD-tree works quite well for small dimensions however for N>10 KD-tree may become too slow. Another drawback is that the basic KD-tree tree do not allows

balancing. The entire tree must be reorder periodically to improve it balancing, which is not convenient for changing data and large datasets [55].

Dijkstra

Dijkstra's algorithm, published in 1959 [59], is a graph search algorithm to produce the shortest path in a graph with nonnegative edge path costs.

For the first iteration the current intersection will be the starting point and the distance to it (the intersection's label) will be zero. For subsequent iterations (after the first) the current intersection will be the closest unvisited intersection to the starting point—this will be easy to find. From the current intersection, update the distance to every unvisited intersection that is directly connected to it. This is done by determining the sum of the distance between an unvisited intersection and the value of the current intersection, and relabeling the unvisited intersection with this value if it is less than its current value. Nodes marked as visited are labelled with the shortest path from the starting point to it and will not be revisited or returned to. Once you have marked the destination as visited you have determined the shortest path to it, from the starting point. This algorithm consideration in determining the next "current" intersection is its distance from the starting point. In some sense, this algorithm "expands outward" from the starting point, iteratively considering every node that is closer in terms of shortest path distance until it reaches the destination. When understood in this way, it is clear how the algorithm necessarily finds the shortest path, however it may also reveal one of the algorithm's weaknesses: its relative slowness in some topologies [60].

2.5.2. Data Mining

Data mining is the process of discovering meaningful correlations, patterns and trends by sifting through large amounts of data stored in repositories. Data mining employs pattern recognition technologies, as well as statistical and mathematical techniques [56]. Such patterns are called structural because they capture the decision structure in an explicit way. In other words, they help to explain something about the data [57].

While large-scale information technology has been evolving separate transaction and analytical systems, data mining provides the link between the two. Data mining software analyzes relationships and patterns in stored transaction data based on open-ended user queries. Several types of analytical software are available: statistical, machine learning, and neural networks [58].

In [49] data mining is separated in five major elements: 1) Extract, transform, and load transaction data onto the data warehouse system; 2) Store and manage the data in a

multidimensional database system; 3) Provide data access to business analysts and information technology professionals; 4) Analyze the data by application software; 5) Present the data in a useful format, such as a graph or table.

Data mining is frequently used to gain knowledge, not just predictions [49]. Its applications are extremely vast, in economics, statistics, forecasting, to find, identify, validate, to use patterns, are some of the example areas.

Some of the data mining algorithms able to be used on the knowledge maintenance process are described in the next section.

K-means

The classic clustering technique is called k-means. All instances are assigned to their closest cluster centre according to the ordinary Euclidean distance metric. The k-means clustering algorithm usually requires several iterations, each involving finding the distance of k cluster centres from every instance to determine its cluster [49]. It remains the most widely used partitional clustering algorithm in practice. The algorithm is simple, easily understandable and reasonably scalable, and can be easily modified to deal with streaming data. Note that each iteration needs N × k comparisons, which determines the time complexity of one iteration. The number of iterations required for convergence varies and may depend on N, but as a first cut, this algorithm can be considered linear in the dataset size. One issue to resolve is how to quantify closest in the assignment step. The default measure of closeness is the Euclidean distance, in which case one can readily show that the non-negative cost function will decrease whenever there is a change in the assignment or the relocation steps, and hence convergence is guaranteed in a finite number of iterations [53].

Support vector machines

It offers one of the most robust and accurate methods among all well-known algorithms. It has a sound theoretical foundation, requires only a dozen examples for training, and is insensitive to the number of dimensions. In addition, efficient methods for training SVM are also being developed at a fast pace. In a two-class learning task, the aim of SVM is to find the best classification function to distinguish between members of the two classes in the training data. The metric for the concept of the best classification function can be realized geometrically. For a linearly separable dataset, a linear classification function corresponds to a separating hyperplane f(x) that passes through the middle of the two classes, x_n can be classified by simply testing the sign of the function f(x): x_n belongs to the positive class if f(x)>0. Because there are many such linear hyperplanes, what SVM

additionally guarantee is that the best such function is found by maximizing the margin between the two classes. Intuitively, the margin is defined as the amount of space, or separation between the two classes as defined by the hyperplane. Geometrically, the margin corresponds to the shortest distance between the closest data points to a point on the hyperplane. Having this geometric definition allows us to explore how to maximize the margin, so that even though there are an infinite number of hyperplanes, only a few qualify as the solution to SVM. The reason why SVM insists on finding the maximum margin hyperplanes is that it offers the best generalization ability. It allows not only the best classification performance (e.g., accuracy) on the training data, but also leaves much room for the correct classification of the future data [53].

Apriori

Apriori is a seminal algorithm for finding frequent itemsets using candidate generation []. It is characterized as a level-wise complete search algorithm using anti-monotonicity of itemsets, if an itemset is not frequent, any of its superset is never frequent. By convention, Apriori assumes that items within a transaction or itemset are sorted in lexicographic order. The introduction of this technique boosted data mining research and its impact is tremendous. The algorithm is quite simple and easy to implement [53].

EM

Finite mixture distributions provide a flexible and mathematical-based approach to the modelling and clustering of data observed on random phenomena. We focus here on the use of normal mixture models, which can be used to cluster continuous data and to estimate the underlying density function. These mixture models can be fitted by maximum likelihood via the EM (Expectation–Maximization) algorithm [53].

PageRank

It is a search ranking algorithm using hyperlinks on the Web. PageRank produces a static ranking of Web pages in the sense that a PageRank value is computed for each page off-line and it does not depend on search queries. The algorithm relies on the democratic nature of the Web by using its vast link structure as an indicator of an individual page‘s quality. In essence, PageRank interprets a hyperlink from page x to page y as a vote, by page x, for page y. However, PageRank looks at more than just the sheer number of votes, or links that a page receives. It also analyzes the page that casts the vote. Votes casted by

pages that are themselves important weigh more heavily and help to make other pages more important [53].

AdaBoost

Is one of the most important ensemble methods, since it has solid theoretical foundation, very accurate prediction, great simplicity (Schapire said it needs only just 10 lines of code), and wide and successful applications. Ensemble learning deals with methods which employ multiple learners to solve a problem. The generalization ability of an ensemble is usually significantly better than that of a single learner, so ensemble methods are very attractive. First it assigns equal weights to all the training examples. Denote the distribution of the weights at the t-th learning round as Dt. From the training set and Dt the algorithm generates a weak or base learner ht. Then, it uses the training examples to test ht, and the weights of the incorrectly classified examples will be increased. Thus, an updated weight distribution Dt+1 is obtained. Such a process is repeated for T rounds, and the final model is derived by weighted majority voting of the T weak learners, where the weights of the learners are determined during the training process. It is evident that AdaBoost was born with theoretical significance. AdaBoost has given rise to abundant research on theoretical aspects of ensemble methods, which can be easily found in machine learning and statistics literature [53].

Bayes

The Bayesian algorithm is a set of rules for using evidence (data) to change your beliefs. Bayesian econometrics is the systematic use of a result from elementary probability, Bayes‘theorem. It is very easy to construct, not needing any complicated iterative parameter estimation schemes. This means it may be readily applied to huge data sets. It is easy to interpret, so users unskilled in classifier technology can understand why it is making the classification it makes. And finally, it often does surprisingly well: it may not be the best possible classifier in any particular application, but it can usually be relied on to be robust and to do quite well [53].

2.5.3. Conclusions

This research will help the author to identify existing works which can be some how linked together in order to respond to the author’s research questions.

Brain analyses intend to understand and represent human’s conceptual knowledge into

ontologies, providing them the ability for knowledge maintenance and learning. The key idea is to associate machine learning techniques together with operational research methods in a way to facilitate the enrichment of ontologies with capabilities of learn.

The use of machine learning systems in certain tasks may improve or even exceed human's performance in their daily tasks. Diagnosis area is one of the main applications of expert systems [49]. Machine learning can be useful in situations in which producing rules manually is too labour intensive [49]. Also, it can be applied to preventative maintenance of electromechanical devices, as it can measure unusual changes in the devices which a human hardly could. There are other areas of machine learning application returning successful results. For example in a loan application, where it not only improved the success rate but also the company could use it to explain to applicants the reasons behind the decision [49]. Domains in which companies possesses massive volumes of precisely data may use machine learning to get profit of it. For example in marketing and sales areas, an automated analysis of checkout data may uncover the fact that customers who buy beer also buy chips, a discovery that can be significant from the supermarket operator's point of view [49].

Data mining can be applied not only for predictions but also to gain knowledge, applying patterns recognition, statistical and mathematics technologies discovering meaningful relationships. The goal of the presented data mining algorithms was to find an algorithm able to somehow measure products importance (for example calculating the weight of its semantic similarity).

In the architecture the applied machine learning technique follows a classification learning type, more specifically the instance-based learning. The instance-based learning deals with ratio scales because it calculates the "distance" between two instances based on the values of their attributes. In the proposed architecture this distance represents the semantic similarities between two concepts (weights). Dijkstra was the "distance" method chosen to retrieve the closest path between two concepts, this means the path with highest semantic similarities. Dijkstra was chosen because of its simplicity and web abundance implementations in JAVA language. To prepare this algorithm to deal with weights, it was only a matter of transforming weights in distances. This is done by inverting its value in order to have the highest weight with the smallest value.

3. KNOWLEDGE BASED METHODOLOGY FOR SEMANTIC INTEROPERABILITY

Nowadays, enterprises are demanded to collaborate and establish partnerships to reach global business and markets [61]. Moreover, several examples of agile and virtual enterprises are proven to be efficient and competitive enough to equal big companies response. The electronic business (e-business) also approached clients and suppliers, with computational systems greatly aiding manufacturing companies in such task.

However, enterprises are facing some difficulties concerning the lack of interoperability of systems and software applications to manage and to increase their collaborative business [62][63]. Multiple organizations operating in the same business domain may have different views of the same “subject”. When they want to describe their knowledge in an electronic way it will probably lead to different conceptual/computational models. Consequently it might conduct to interoperability problems when systems intend to share information between each other. The need of having complete and fully integrated systems, that seamlessly communicate and understand each other, requires meaning support of the data within probable multiple domains involved [64].

Interoperability can be described as the ability of two or more systems or components to exchange information and to use the information that has been exchanged [65]. Also it can be described using the Levels of Conceptual Interoperability Model (LCIM), i.e. a model that shows the different levels of interoperability that may exist between systems, from technical interoperability through conceptual interoperability [66][67] (Figure 3.2).

3.1. Knowledge Representation Model for Systems Interoperability

Knowledge may be defined as facts, information, and skills acquired by a person, through experience or education; the theoretical or practical understanding of a subject. It is the awareness or familiarity gained by experience of a fact or situation [68]. Knowledge is also used to mean the confident understanding of a subject with the ability to use it for a specific purpose if appropriate.

Thus, knowledge is the appropriate collection of information, gained through a deterministic process and which intent is to be useful. When someone memorizes information, then they have accumulated knowledge. This knowledge has useful meaning to them, but it does not provide for, in and of itself, an integration such as would infer further knowledge [69]. Knowledge pursues the gathering of new knowledge in a kind of never

ending cycle. Knowledge acquisition is the action beyond such process. Its main objective is to transform tacit in explicit knowledge, and effectively to improve the approach to elicit knowledge from domain experts, towards interoperable intelligent systems [70]. Tacit knowledge is knowledge that people carry in their minds, which provides context for people, places, ideas, and experiences [71]. Explicit knowledge is knowledge that has been or can be articulated, codified, and stored in certain media [71].

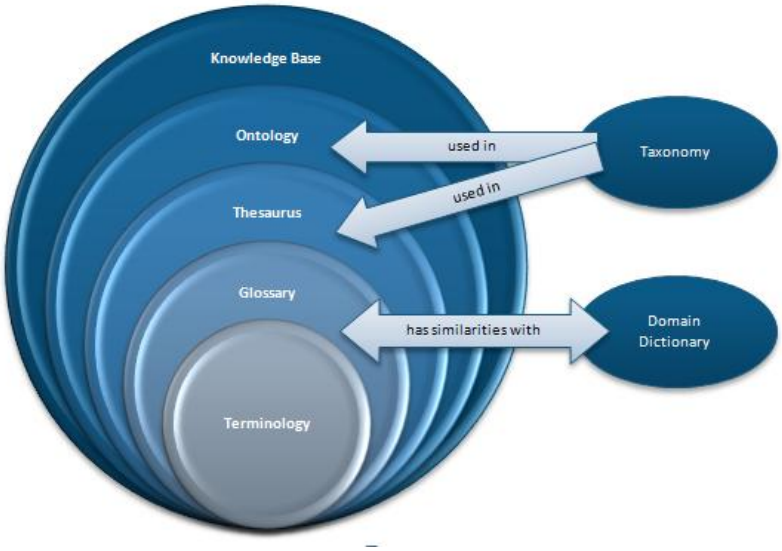

Figure 3.1: Knowledge Representation Elements

Knowledge representation studies the formalisation of knowledge and its processing within machines. Techniques of automated reasoning allow a computer system to draw conclusions from knowledge represented in a machine-interpretable form. A Knowledge Representation Element (KRE) can make the formal representation of knowledge in a specific domain become easier. Figure 3.1 illustrates the KRE's that should be defined in the path to build a domain's knowledge base. It represents the distinct level of conceptualization that each one has, showing an increase of its presence from Terminology to the Knowledge Base [72].

Glossary is a specialized vocabulary with corresponding annotations/definitions as a domain dictionary. This vocabulary includes a terminology that is unique and/or has special meaning in the field of interest. Its purpose is to unify the knowledge sharing in information systems communications. Thesaurus represents a structure (taxonomy) of associated meanings supplied by the glossary, establishing the lexicon in the domain. Ontology represents the knowledge related to the domain. It aggregates the lexicon represented by the thesaurus accomplished with rules to represent some specific knowledge (e.g. products). The knowledge Base represents the knowledge in a domain combined with explicit representations of real individuals (e.g. instances of products).

The Knowledge Representation Requirements Model (KRRM) is a model proposed by Turnitsa and Tolk that shows the needs for greater ability to represent knowledge, and also gives the levels of conceptual interoperability that may be reached if the requirements are met [67].

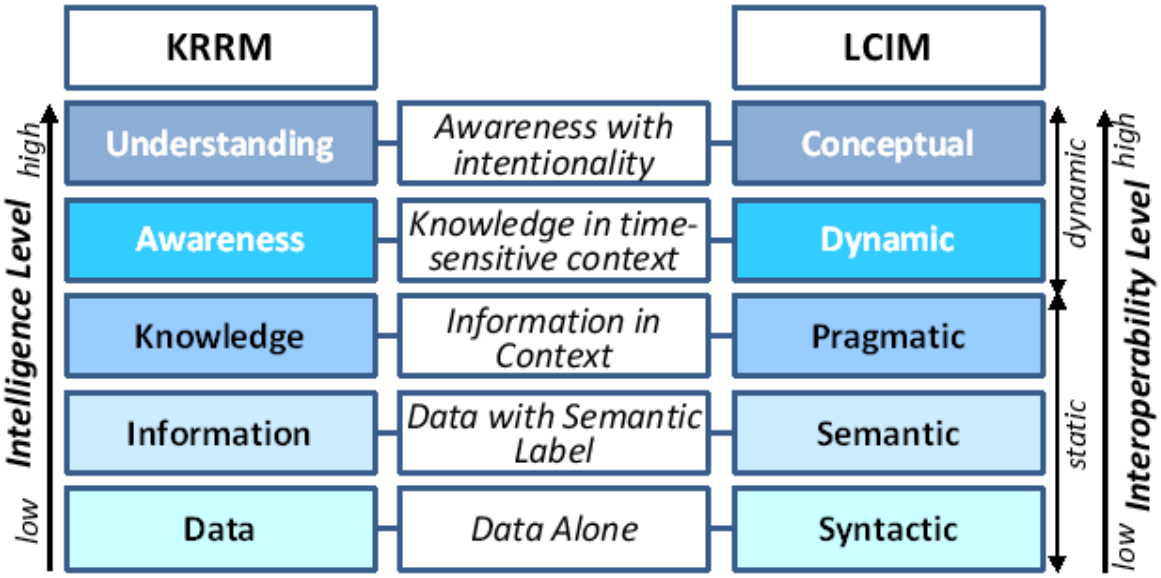

Figure 3.2: Knowledge Representation Requirements Model [67]

Nowadays, the increase of obstacles to systems' interoperability (both in number and complexity) is especially due to the rapid enterprises business developments in network technology and services. A whole continuous stream of newly shared information has instigated the improvement of the information systems interoperability solutions and alternatives. A consequence of the dynamic nature of these new distributed information environments is the introduction of uncertainty about the available information [71].

A low level of intelligence is required to deal only with data as well as a low level of interoperability. This is represented in Figure 3.2, in the lowest layer. Thus, the level of intelligence is directly proportional to the interoperability level, and to solve uncertainties on knowledge representations, it is needed to jump from the static levels of interoperability to the dynamic ones (see interoperability levels at right part Figure 3.2).

For this it is needed, as identified by the Turnitsa and Tolk's model, to reach the awareness intelligence level of systems by having its knowledge in time-sensitive context. Such objective could be reached by knowledge maintenance abilities from external knowledge system actors' feedback. Such knowledge maintenance should be able to lively track the knowledge evolution and update itself accordingly to the systems outputs interactions.

3.2. Extending mentor methodology to a dynamic level

Systems that possess knowledge and are capable of decision making and reasoning are regarded as 'intelligent' [72][73]. Some recognised techniques, such as fuzzy logic, artificial neural networks, machine learning and evolutionary algorithms can contribute to the increase of the system's 'machine intelligence quotient' [74]. The rationale behind the intelligent label of those techniques is their ability to represent and deal with knowledge [75]. Consequently, the feedback given by the users' interaction with an ontology-based system could be used to get some patterns. These patterns through appropriate analysis could be used to improve the knowledge/ontologies. Such objective can be reached through the use of the most appropriate technologies (e.g. machine learning algorithms), which associated to ontologies could provide a statistical result, facilitating to reach a "Learning Ontology (LO)".

As mentioned by Brewster [76], the evaluation of the output of ontology learning systems remains a major challenge. The usual approaches to ontology evaluation have largely been based on quality control of the ontology building process and ensuring the ontology abides by certain principles [77][78]. But, on this case it is intended to provide another kind of LO, which intends to learn from its usability from users (e.g. customers) in order to constantly improve the semantics interoperability between systems and to maintain its represented knowledge. Knowledge maintenance can be a facilitator to the dynamic information systems interoperability and consequently as a main actor to the increase of systems interoperability.

In [79], authors proposed the use of MENTOR methodology, to organize the knowledge and then as a facilitator to the knowledge maintenance processes implementation. MENTOR – Methodology for Enterprise Reference Ontology Development [79], is a collaborative methodology developed with the idea of helping a group of people, or enterprises, sharing their knowledge with the other in the network, and provides several steps as semantic comparisons, basic lexicon establishment, ontology mappings and some other operations to build a domain's reference ontology. It aims to combine the knowledge described by different formalisms in a semantic interoperable way [80]. This methodology is composed by two phases and each phase has three steps, which can be seen on Figure 3.3.

The Lexicon Settlement, or Phase 1, represents the knowledge acquisition by getting a collection of terms and related definitions from all participants. This phase is divided into three steps: Terminology Gathering, Glossary Building, and Thesaurus Building. The first step is a very simple one, and it represents the knowledge gathering from all actors in the collaborative network in a form of a list of terms. In the Glossary Building step, a glossary is built after serial discussions about the terms that every participant contributed to the network

on the previous step. These discussions are followed by a voting process, with all participants deciding which corresponding terms and definitions compose the glossary. Beyond the glossary, the semantic mismatches record is another output that results from this step. The last step of this phase is composed by a cycle where the knowledge engineers define a taxonomic structure with the glossary terms. If there is an agreement in both structure and classified terms, the thesaurus is defined. If not, the cycle starts again for another iteration. In this first phase, it could be valuable to have a multi-language dictionary for situations where a common language is not shared by all participants.

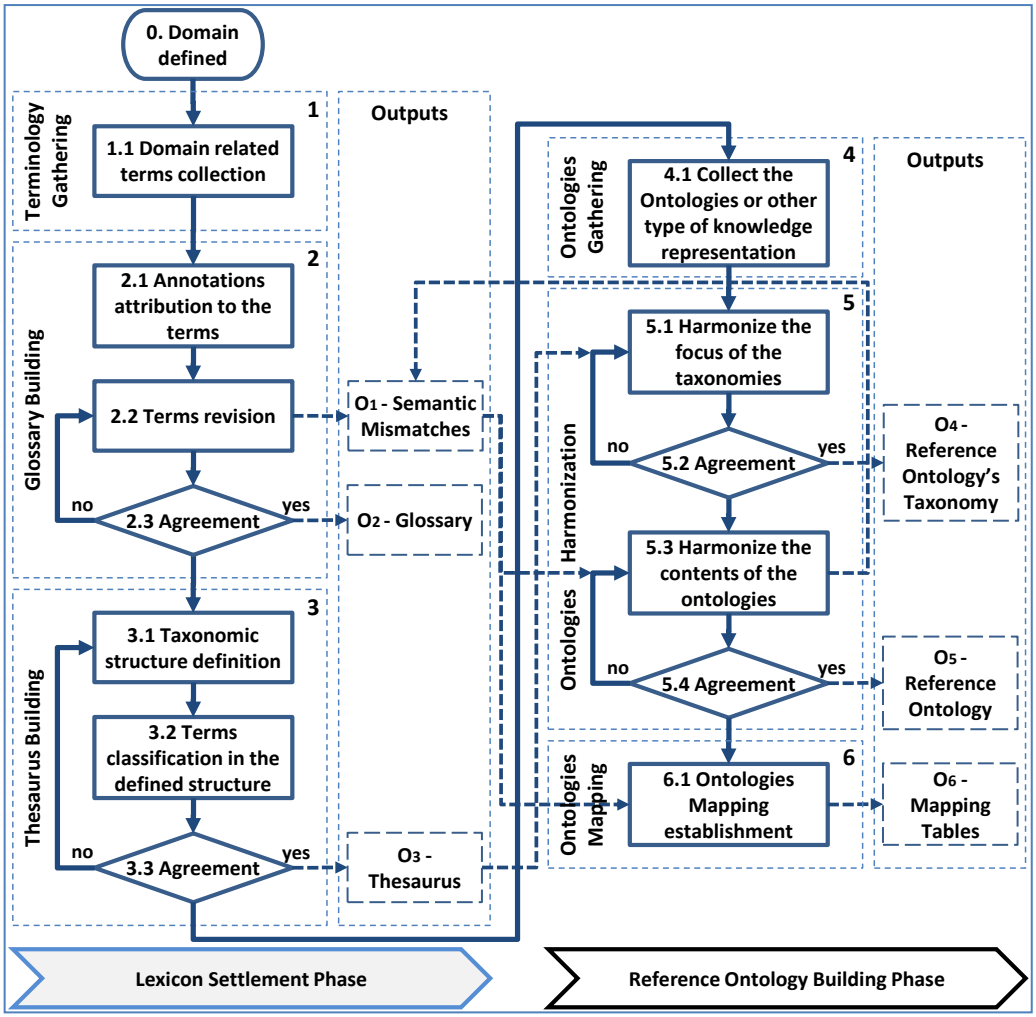

Figure 3.3: MENTOR Methodology

The Reference Ontology Building, or Phase 2, is the phase where the reference ontology is built, and the semantic mappings between participant's ontologies and the reference ontology are established. This phase, just like the first phase, is divided into three steps: Ontologies Gathering, Ontologies Harmonization, and Ontologies mapping. The first step comprehends the acquisition of ontologies in the defined domain. In Ontologies Harmonization step, it is needed to proceed to two harmonization types: taxonomic

harmonization and contents harmonization. First, a discussion and voting process about the reference ontology structure takes place where the common classes are defined by unanimity. This process of discussing and voting is then repeated for the contents harmonization. The final step of this phase, the Ontology Mapping, attempts to relate the vocabulary of two ontologies that share the same domain. In this case, the idea is to establish mappings between each participant's ontology and the reference ontology defined on the previous step [79].

The extension of MENTOR methodology capabilities to a dynamic level is obtained as a third phase of the methodology. This phase implements the feedback mechanisms for a sustainable evolutional learning of the dynamic ontological system. Such phase represents the encircling of the lexicon settlement and reference ontology building phases of MENTOR's methodology. It will provide a continuous LO (Figure 3.4).

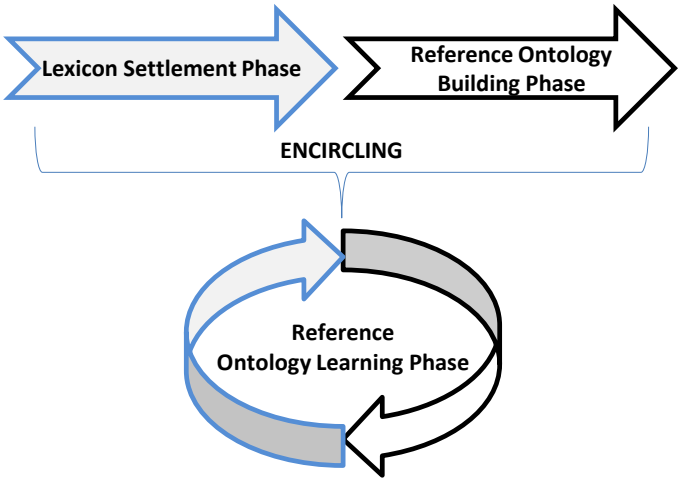

Figure 3.4: Changes to MENTOR methodology [80].

3.3. Knowledge based methodology.

Manufacturing industry has been widely populated with internet services geared to support parts and components evaluation such as e-procurement of mechanical parts. This was the scenario chosen to explore knowledge maintenance techniques in a reference ontology created in a collaborative environment using MENTOR methodology.

In the following it is presented an overview of the MENTOR tool architecture accomplished with its maintenance plug-in, called oLEARCH. The concept was inspired from the concept LEARCH defined by Ratliff et al. [81] that means "LEArning to seaRCH". Such concept represents algorithms for imitation learning in robotics with the main purpose to search something. Thus, author found appropriated to define "oLEARCH - Ontologies LEArn

by seaRCHing", as a new concept related to ontologies able to change/adapt their knowledge (to learn) through their users' patterns of searching/reasoning [83].

The proposed knowledge-based methodology architecture (Figure 3.5) is composed by three main components: 1) information models able to represent knowledge; 2) java libraries acting as ontology handlers, machine learning functions, and web services; and 3) user interfaces able to provide MENTOR and its maintenance plug-in functions. All of these components are supported by the Protégé tools namely: Jena API; Protégé server and its MetaProject Ontology plus the Collaborative Protégé plugin.

Protégé is a free, open source ontology editor and knowledge-base framework that supports two main ways of modeling ontologies via the Protégé-Frames and Protégé-OWL [82]. Protégé also provides a set of libraries that enable a user to work with ontologies in Java language [84].

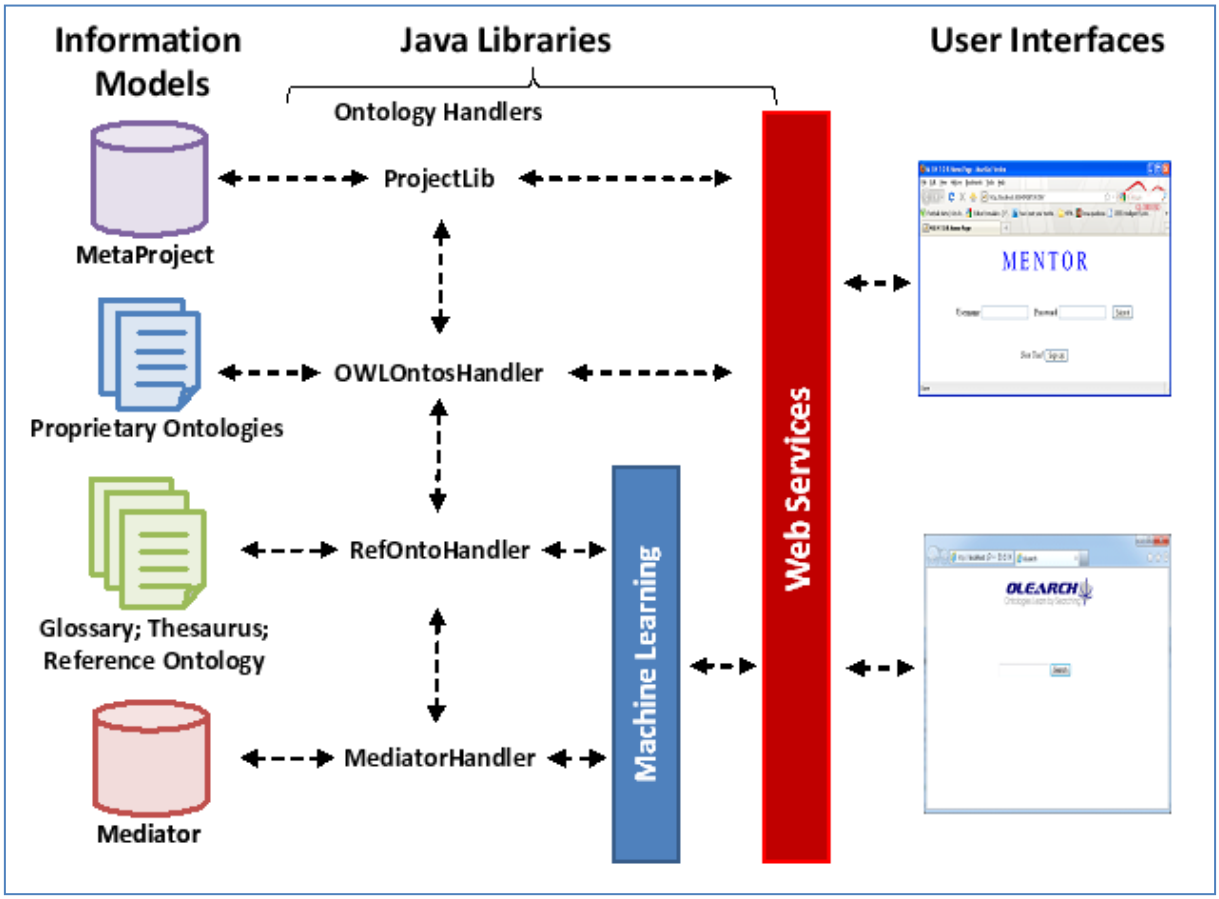

Figure 3.5: Proposed architecture for knowledge-based methodology implementation.

Collaborative Protégé is an extension of the existing Protégé tool to support collaborative ontology building [85]. With Collaborative Protégé users are able to join a project through a server, and if they have permission, they can edit ontology classes and properties through desktop or web Protégé clients. All changes made are synchronically

shared with all the participants through associated annotations and in additional users are able to track the changes and notes.

With this architecture users through the MENTOR part, are able to create knowledge-based projects or connect to existing ones, and run the entire six MENTOR's collaborative ontology building steps to the definition of glossary, thesaurus and reference ontology respectively of a specific domain. Other operations have been also developed to provide management operations to the process. The Metaproject is a frame-based ontology that comes with Protégé. Its key role is to represent the information about hosted knowledge projects with their users and access permissions [85]. The original Metaproject was accomplished with specific MENTOR requirements to keep track of the MENTOR steps and their status, like for instance, to know the name of users that finished the operations of a current step.

oLEARCH is responsible to provide knowledge maintenance abilities to the proposed architecture. It uses a machine learning technique, which main purpose is to find and describe structural patterns in data [49]. This is made through a product search tool in mechanical area. When users search a specific product they introduce concepts that will provide an increase of the lexicon associated to that specific product. Such process is reached through reasoning, using introduced concepts over the reference ontology to find the desired products classified on that ontology. The machine learning process starts by clustering the introduced concepts with the reference ones (from reference ontology) and, how much times a concept is used, more it gains importance (weight) on decision for the output results. The output is ruled by an instance-based learning approach, which distance is inversely related to the "weight" that the concepts have. This means that the result product is found through a distance function that determines which product is closest in (semantics) relation to the concepts introduced.

After several utilizations, patterns are obtained. One example is related to the proposition of new concepts to a specific object. Thus, with proper patterns knowledge is able to be adapted & maintained to new nomenclatures used by the community. This will be able to be accomplished by a right connection between oLEARCH and MENTOR.

As a final remark, Web Services are useful in this architecture as they are responsible for giving to worldwide users the opportunity to use these services, facilitating them the building of their own user interfaces to use MENTOR and oLEARCH functionalities.

3.4. Conclusions

The methodology proposed has been implemented and tested in an industrial furniture environment. The methodology for collaborative enterprise reference ontology building was tested using an example about bolt suppliers to validate it. The validation addressed the process of choosing a bolt supplier by a mechanical engineer or designer, which quite often brings interoperability issues. This interoperability, in this case, is related to semantics, to the nomenclature and definition of a bolt. Suppliers usually define proprietary nomenclatures for their products and its associated knowledge representation. Problems persist although standardization bodies developed and proposed several standards focused in bolt specifications. Thus, the need to align product data and knowledge emerged as a priority to solve the dilemma. By using the proposed extended MENTOR methodology these interoperability problems were worked out.

After reference ontology agreement, the second stage addresses the upgrade and enrichment that reference ontology may suffer. The dynamic evolution of systems and manufacturing environments contribute to decrease the interoperability level over time, thus, they need to be updated and maintained. This is covered with oLEARCH, which validation is currently carried out in an e-procurement scenario between manufacturing companies. On-going validations focused on economic aspects will contribute to completeness of current work.

4. REFERENCE TOOLS & MODELS FOR ONTOLOGY MANAGEMENT

This chapter presents reference tools and models for ontology management. It presents the oLEARCH architecture used technologies and ontologies. These ontologies were used to represent a domain business knowledge and semantic mappings.

4.1. Used Technologies

The oLEARCH application was implemented in NetBeans 6.9.1, which is an open-source **I**ntegrated **D**evelopment **E**nvironment (IDE) supporting development of Java applications, JavaScript, WebServices, C++, etc, owned by Sun Microsystems Company.

4.1.1. Java

In oLEARCH there are sixteen java classes inside five packages. The main used classes and functions can be seen in Appendix 1. Java is a programming language that allows developers to write programs, such as utilities, games, and business applications. This software is platform-independent and can be developed to run in almost every system just using a Web browser. One of the biggest advantages of Java language is being object oriented which permit to reuse implemented code by other developers [86].

The implemented code of some main functions is presented in Appendix 2. An example of the implemented code is shown below. The presented function receives a graph with respective weights and returns a graph with all weights inverted. The smallest weight will be one as it uses the formula (biggest weight +1 – current graph weight).

```
public List transformWeightInDistance(List graph){
    int maxWeight = 0;
    List graphList = new ArrayList();

    for (Iterator i = graph.iterator(); i.hasNext();){
        int auxWeight = 0;
        Graph myGraph = (Graph) i.next();
        auxWeight = myGraph.getWeight();
        if (auxWeight > maxWeight){
            maxWeight = auxWeight;
        }
    }
    for (Iterator j = graph.iterator(); j.hasNext();){
        int graphWeight = 0;
        Graph finalGraph = (Graph) j.next();
        graphWeight = finalGraph.getWeight();
        finalGraph.setWeight(maxWeight + 1 - graphWeight);
        graphList.add(finalGraph);
    }
    return graphList;
  }
```

Figure 4.1: Java code implemented to transform a concept weight in distance.

4.1.2. Web Services

Web Services big advantage is that it provides the possibility to share the application functions on the internet. Basically it converts an application into Web applications, simply publishing the services. Using this technology it is possible to access the oLEARCH application anywhere in the world without need to install any software, only using an internet browser [87]. Another relevant Web Services property is that they are a good solution for interoperability as they provide the communication between different operating systems, different programming languages, or even different network platforms. Their functional logic is calling a library using received parameters and returning values.

```
try {
            olearchws.OLearchServiceService service = new olearchws.OLearchServiceService();
            olearchws.OLearchService port = service.getOLearchServicePort();
             // TODO initialize WS operation arguments here
            // TODO process result here
      } catch (Exception ex) {
    }
```

Figure 4.2: Implemented code to invoke oLEARCH web services.

In oLEARCH there are two web services available. One is responsible to search and return the desired products and the other one provides the ability to increment a concept weight in the reference ontology.

In the Figure 4.2 it is displayed the oLEARCH web services invocation code.

AJAX (Asynchronous JavaScript and XML)

Ajax is a set of tools used to build a fast and dynamic web site. Using Ajax it's possible to save resources as it can selectively modify a part of a page without reloading the all document. Ajax provides Web application's functions sharing using asynchronous XML messaging [88]. In oLEARCH, it is used to invoke SOAP Web services, enabling data exchange in order to access oLEARCH application through internet.

The example shown in Figure 4.3 is the used one by oLEARCH when increase weight service is called, which sends to the server the product name to have its weight increased. This is parameterized using type "POST". Parameter 'url' is the URL to which the request is sent, 'dataType' is the type of data expected from the server, 'data' has the map that is sent to the server with the request and 'contentType' the type of content.

```
 var jqxhr = $.ajax({
        type: "POST",
        url: webMethod,
        dataType:  "xml",
        data: soap,
        contentType:"text/xml;
charset=utf-8",
```

Figure 4.3: Ajax implemented invocation.

jQuery

jQuery is a library with javascripts functions able to work with AJAX technology. It permits to assign events, to define effects and animations, to create and/or change HTML elements, etc [89]. In oLEARCH GUI (web page) such library was used. The following code show how such library was inserted to oLEARCH web page to facilitate soap messages handling using ajax.

```
<script language="javascript"  type="text/javascript" src="jquery-1.5.js"></script>
```

Figure 4.4: Code to insert jQuery library into a web page.

Soap

Soap is a protocol XML-based for accessing a WebService. It provides the ability to communicate between applications running on different operating systems, with different technologies and programming languages through HTTP which is supported by all Internet browsers and servers [90].

Figure 4.5 shows the used SOAP request to call the responsible function to increase a product weight. It just sends to oLEARCH server the user selected product name.

```
//Create a new SOAP Request
var soap = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>"+
   "<S:Envelope xmlns:S=\"http://schemas.xmlsoap.org/soap/envelope/\">"+
   "<S:Header/>"+
   "<S:Body>"+
      "<ns2:updateWeight xmlns:ns2=\"http://oLearchWS/\">"+
         "<prodName>"+valor+"</prodName>"+
      "</ns2:updateWeight>"+
   "</S:Body>"+
"</S:Envelope>"
```

Figure 4.5: Soap request.

JavaServer Pages

JavaServer Pages programming language (JSP) was used to design oLEARCH web application. This technology provides a simplified, fast way to create dynamic web content

platform-independent. JSP is stored in its textual form in the web application. When a JSP page is first called the file is compiled into Java Servlets[1] class and stored in the server memory providing fast responses when called again as it don't need to be recompiled [91].

In JSP it is possible to use JAVA code, it just needs to be between two symbols like <% %>. Figure 4.6 show the code of the first oLEARCH JSP page and its result.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">

<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>oLearch</title>
   </head>
   <body>
      <form action="oLearch.jsp" target="_self">
      <table border="0" cellpadding="2" align="center">
<tr>
      <td>
            <input type="text" name="keyword">
            <input type="submit" value="Search">
      </td>
</tr>
</table>
      </form>
   </body>
</html>
```

Figure 4.6: oLEARCH JSP code and web page resulting.

4.1.3. Protégé

Protégé is a free, open source ontology editor and knowledge-base framework that supports two main ways of modelling ontologies via the Protégé-Frames and Protégé-OWL [82].

Protégé-OWL api - Jena

One of the most used Java APIs for RDF and OWL is JENA, which provides functions to help programmers dealing with OWL data models.

Figure 4.7 presents a code example in JAVA about how to read an ontology

[1] A servlet is a Java programming language class used to extend Web services applications **Error! Reference ource not found.**.

(refOntology) from a URI an ontology into a Jena OWL model.

```
public static JenaOWLModel owlModel;

String uri = "file:///C:/";
fileName = “refOntology.owl";
owlModel = ProtegeOWL.createJenaOWLModelFromURI(uri + fileName);
```

Figure 4.7: Read an ontology into variable owlModel.

This code is used in oLEARCH to access the reference ontology shown in Figure 5.4. All the instances, classes and properties information is gathered in ‘owlModel’ variable.

Protégé-OWL to Java classes

Protégé also provides an open source Java library to work with ontologies in Java language [84]. This technology provides classes and methods to manipulate OWL data models in a simple and faster way.

OLearch interacts with the MEDIATOR ontology by using these java classes. Figure 4.8 shows an example of one of these referred Java classes, which is a method to get a model element.

```
public ModelElement getModelElement(String name) {
      RDFResource res =
owlModel.getRDFResource(OWLUtil.getInternalFullName(owlModel, name));
      if (res == null) {
         return null;
      }
      if (res instanceof ModelElement) {
         return (ModelElement) res;
     } else if (res.hasProtegeType(getModelElementClass())) {
         return new DefaultModelElement(owlModel, res.getFrameID());
      }
      return null;
   }
```

Figure 4.8: Example of reading a model element using protégé-OWL to java classes.

SPARQL Query

SPARQL is a standard query language for the semantic web which can be used to

query an RDF schema or an OWL model [92]. It is very useful to filter out individuals (instances) with some specific characteristics, however it is not that simple and easy to work as other protégé technologies mentioned before.

Figure 4.9 shows a SPARQL function implemented in oLEARCH to get the product with the entered name.

```
String prefixMO = "PREFIX MO:<" + OWLUtil.getActiveOntology(owlModel).getURI() +
">";

   String queryString = prefixMO  +
      "SELECT ?Name ?Product_Name " +
      "WHERE { " +
      "?Name " +
           ":Product_Name \"" + product_name + "\"; " +
   "}";
           QueryResults results = owlModel.executeSPARQLQuery(queryString);
```

Figure 4.9: SPARQL java code exemple.

The elected SPARQL editor was “TopBraid Composer”. This is a graphical development environment for modelling data, connecting data sources, and designing queries, rules and semantic data processing chains [93].

4.1.4. Conclusions

All the presented technologies were used for the oLEARCH implementation. Java is a powerful programming language that easily interacts with other technologies, as the used ones in oLEARCH. Adding web services to oLEARCH was the way to provide its access to users at any part of the world. For a better interaction with ontologies, protégé is a good solution, not only because of all existing documentation but also because of all the available tools. However protégé to java classes must be improved because it still has some bugs, and SPARQL technology is not that simple to work with, but still be useful to filter individuals, as commented before.

4.2. Reference Ontology

Ontologies represent knowledge through the use of elements with its properties, rules, facts and relations. OLearch uses an ontology to represent the knowledge of a domain, thus it has instance products classified on it. This ontology is shown in Figure 4.10.

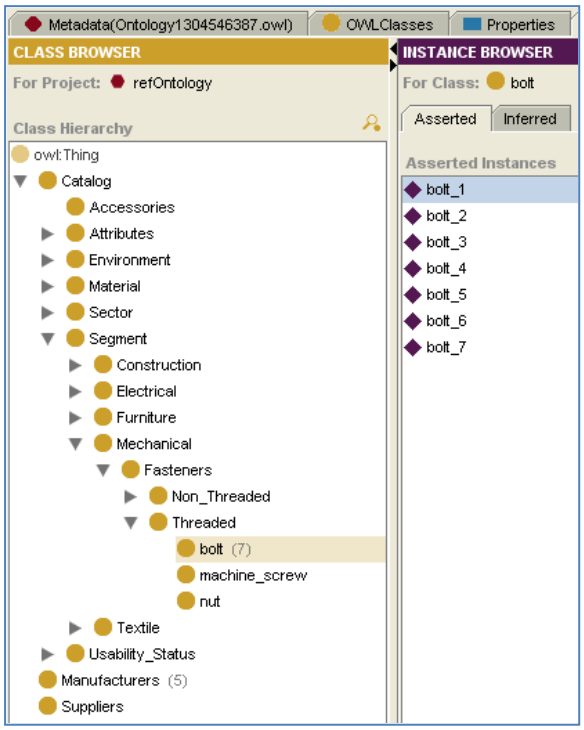

Figure 4.10: Reference Ontology classes and instances.

By other side, Figure 4.11 shows an example of a reference ontology instance, in this case representing a bolt, with its properties. (e.g. “product name” property, which in this case is “Metric bolts”).

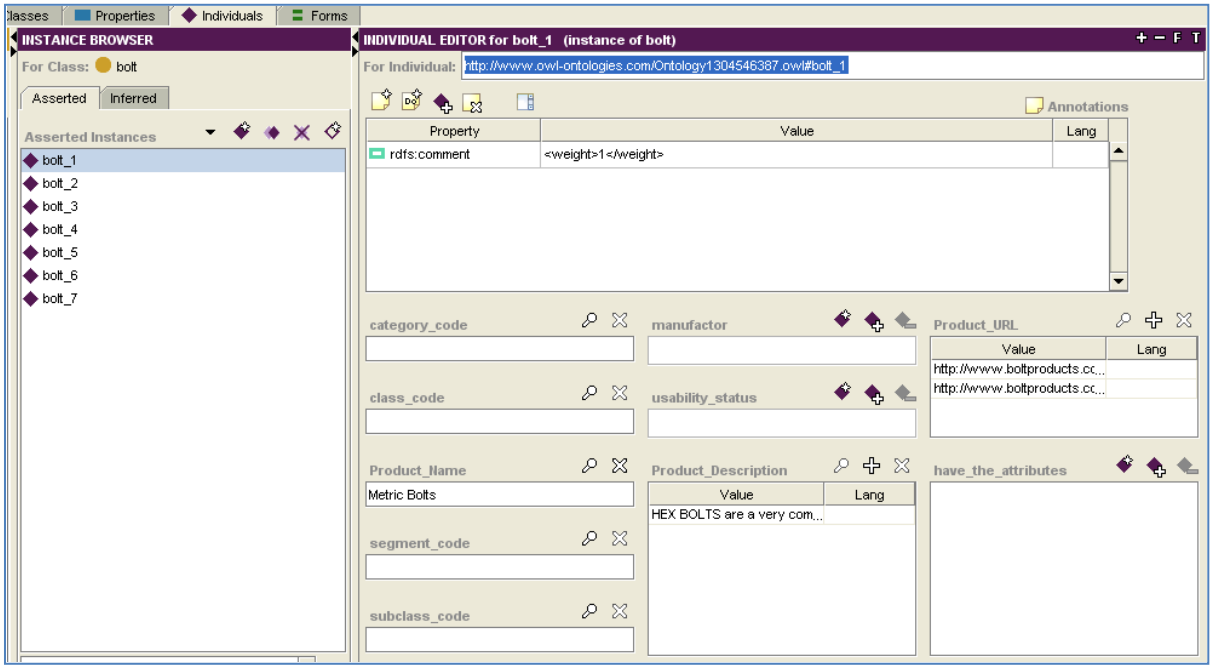

Figure 4.11: Reference Ontology instance with its properties.

All the concepts used in this ontology could have a respective weight to measure its importance, representing its appearance frequency in users searching processes. This

weight is applied into the reference ontology's classes and instances in rdfs comment property, as presented in middle upper of Figure 4.11. The ones used by the other concepts properties can also have its own weights but represented in the mediator ontology. When a reference ontology concept is searched then its weight is increased and the concept is returned to oLEARCH. If a user selects one of the searched results then oLEARCH will communicate back to the reference ontology the user choice in order to increment such selected concept weight. These interactions are made through Web Services, and then JENA, SPARQL and Protégé-to-class.

4.3. Mediator Ontology

The Mediator ontology (MO) has been built up as an extension to the Model Traceability Ontology defined in [94], which addresses traceability as the ability to chronologically interrelate the uniquely identifiable objects in a way that can be processed by a human or a system [95]. Thus it is used to represent mapping through tupple approach. The mapping tuple is defined accordingly to [96], consequently, its expression is of the form: <ID, MElems, KMType, MatchClass, Exp>, where ID is an identifier; MElems is the pair (a,b) that indicates the mapped elements; KMType represents the Knowledge Mapping Type (e.g. Conceptual; Semantics; and Instantiable Data); MatchClass stands for Match/Mismatch Classification depending on the KMType, where some of its possible values are —Equal||, —MoreGeneral||, —Disjoint|| etc.; finally Exp represents the formal expression that relates the mapping elements [97]. The structure of the mediator is presented in Figure 4.12 and described as follows.

The Mediator has two main classes: Object and Morphism. The Object represents any InformationModel (IM) which is the model/ontology itself and ModelElements (also belonging to the IM) that can either be classes, properties or instances. The Morphism basically represents the Mapping Tuple described before: it associates a pair of Objects (related and relating – Melems in MapT), and classifies their relationship with a MorphismType, KnowledgeMappingType (if the morphism is a mapping), and Match/Mismatch class (MatchClass in MapT). The Morphism is also prepared to store transformation oriented ExecutableCode that will be written in the Atlas Transformation Language (ATL) and can be used by several organizations to automatically execute the mapping, transforming and exchanging data with their business partners as envisaged in [98].[95]

In oLEARCH it is used only to represent mappings of conceptual types. Its main purpose is to record new relations between new concepts and/or between existing reference

ontology concepts and manage their weights.

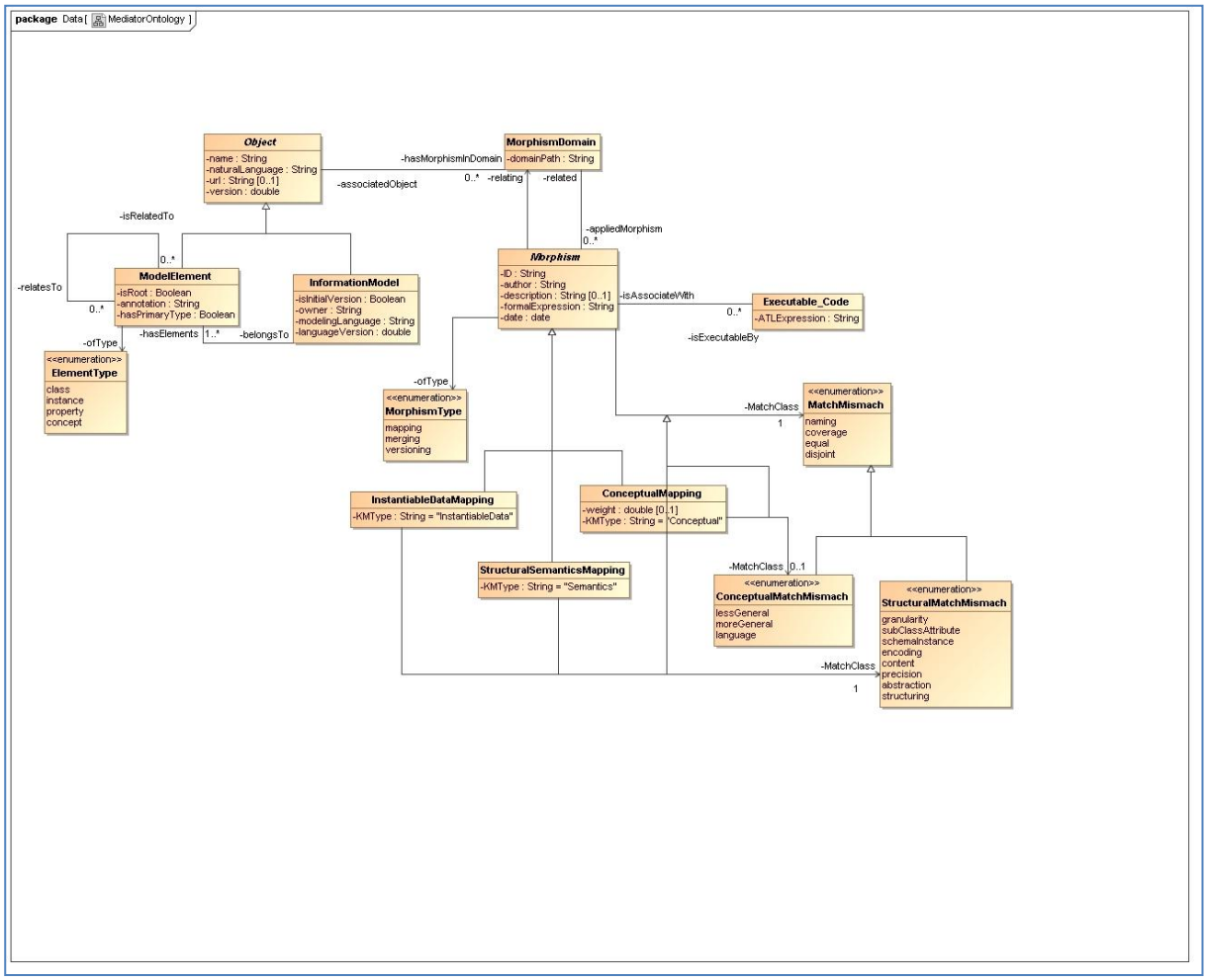

Figure 4.12: Mediator's model.

The association of MO with the reference ontology helps on the searching task because of the mappings (relations) it contains.

When a user searches for more than one concept, oLEARCH assumes that they are somehow connected and so it creates relations between them and records it in the MEDIATOR ontology. Figure 4.13 contains an example of how a relation is stored into MO. In this case is a relation between a "bolt" and a "parafuso" concept. Looking carefully to the example it is possible to see that such conceptual mapping has already a weight of four. This means that such concepts were searched together already four time, or after introduced together once the concept that is not present at a reference ontology was already searched three times. It is possible that one new concept in the MO can have a higher weight that the existing concept in the reference ontology but semantically equal. This can be viewed as a new concept due to be more often used than the one presented in the reference ontology.

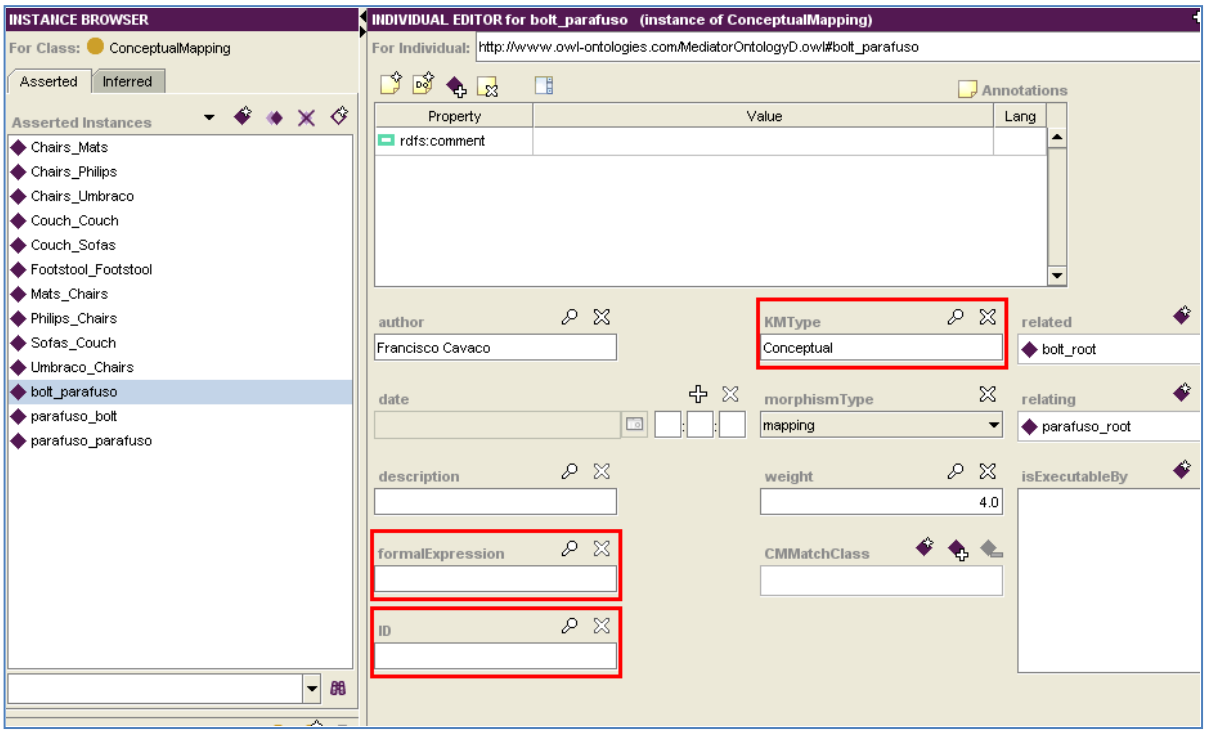

Figure 4.13: Mediator Instance representing a relation between ‘bolt’ and ‘parafuso’.

If it’s just one searched concept and it doesn’t exist in the reference ontology it will be also recorded in MEDIATOR ontology, with a potential of being a new product that users just started to look for.

All these relations have an interesting result, as they may be crucial for a user to find the desired product. A simple application of this property is for example a user that wants to find “couches.

If the reference ontology doesn’t have such concept the result will be null. However if there is a search that relates “couches” with “sofas”, Mediator will establish a relation between them. Next time a user searches for “sofas” or “couches” the result will be the same. As mentioned before, MEDIATOR is also responsible to provide the administrator some user searching patterns as for possible new products and new concepts. This functionality is exclusive of the Admin GUI.

To implement oLEARCH the author applied different technologies able to seamless communicate between each other. The developed oLEARCH architecture is presented in Figure 5.1. It has represented both ontologies and oLEARCH GUI. OLearch can be accessed by any user through oLEARCH Web application or by an administrator using oLEARCH Administrator application in local machine. OLearch GUIs interacts with the knowledge base, represented in the referred picture by the two ontologies, MEDIATOR and the reference ontology.

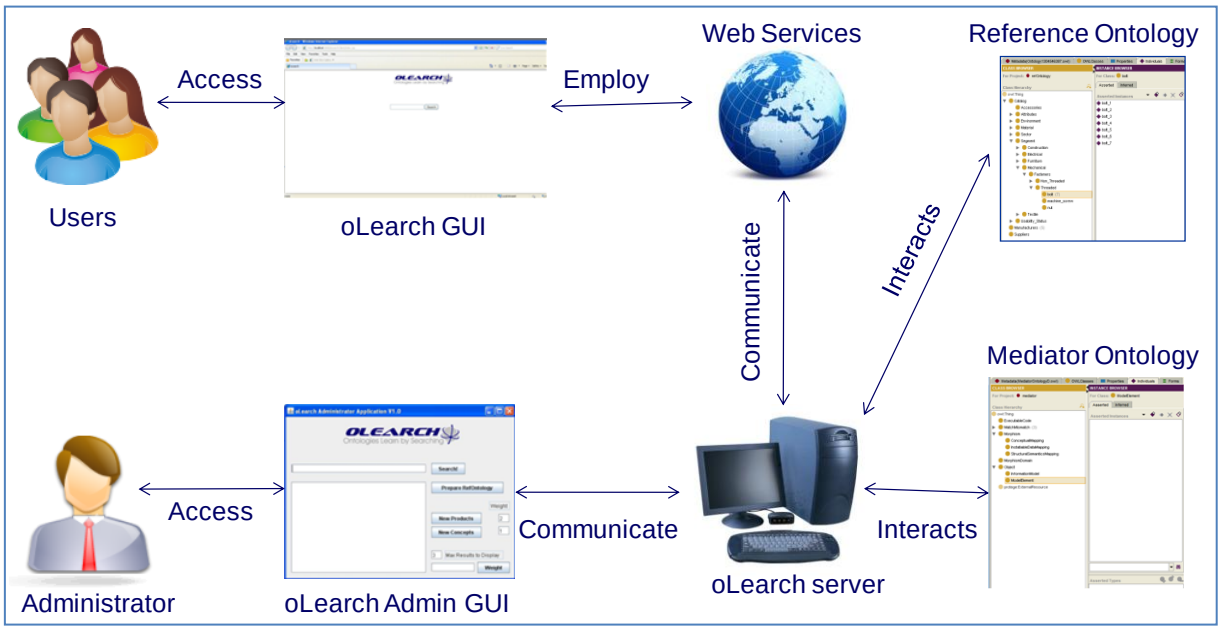

Figure 5.1: Project Architecture

"oLEARCH - **O**ntologies **LEA**rn by sea**RCH**ing" is a search engine application available to users at any part of the world by internet. This system learns from user's searched concepts improving the knowledge base. This knowledge base is consisted by two ontologies, MEDIATOR and a reference ontology.

5.1. OLearch Services

There are two oLEARCH services, one to get products and another one to update a product weight. These services are available to final users through web services and locally

to the administrator.

5.1.1. Get products service

This service is activated when a user searches for a single or multiple concepts. This service is represented by two main functions, one to take care of the searching concepts and another to filter and return the related products found. Each concept is handled accordingly if, in the knowledge base, it is a class, instance or a property.

The first thing oLEARCH does is to split all the entered concepts and check if they exist in the knowledge base adding them, if positive, to the respective global list to be processed later. These lists are used later in the program to build the correct graph. This check starts to search in overall MEDIATOR for existing relations with the entered concepts. Then each concept is compared with the existing ones in the reference ontology in order to see if they are an existing class, instance or property increasing its weight when found, except when it is a property; in these cases, the increased weight will be the corresponding class weight. Also, each concept will be compared with MEDIATOR ontology to check if they are an existing model element. In case the user enters more than one searched concept and the handled concept is not a model element, or if it has not other classification (class, instance or property), then a model element in the MEDIATOR ontology is created to add this new concept.

Next step in oLEARCH is to start joining all the entered concepts, two by two. For example, if a user enters the “A B C” sequence, then this function will return a list with “AB”, “AC” and “BC”. This list is then compared with MEDIATOR ontology elements to search for existing relations. To find these relations, oLEARCH verifies if the first concept is related with the second concept and increases its weight if positive. It also checks for the opposite, changing the position of the first concept with the second one. If no relation exists between these concepts, then oLEARCH will create it, except when just one searched concept is entered and that already exists in the reference ontology. These relations between two model elements are created as morphism of conceptual type.

The second function of this service is, as mentioned before, where the selected products are handled and returned to the user ordered by products importance (products with higher weights will come first). It’s in this step where the referred global lists are used, starting to remove all child classes existing in the class list to avoid having duplicate results. For each concept (node) a label is created with ID, name, type (Classe, Instance or MEDIATOR element) and URI. Also, graphs are created, with origin, destiny, weight and if it is or not an instance (product) properties. There are three kinds of global lists, one with reference instances, another one with reference classes and a last one with MEDIATOR

model elements. The logic here is if it's an instance (product) then it will be connected from "_inic" (origin) to "_out" (destiny). If it's a class, then the algorithm will search for each child classes and its instances to create all the possible connections. Regarding MEDIATOR elements, these will be separated as if they are related to a reference class or instance and then handled accordingly. Only instances are connected to the destiny "_out". The Figure 5.2 presents a possible graph with some mentioned connections.

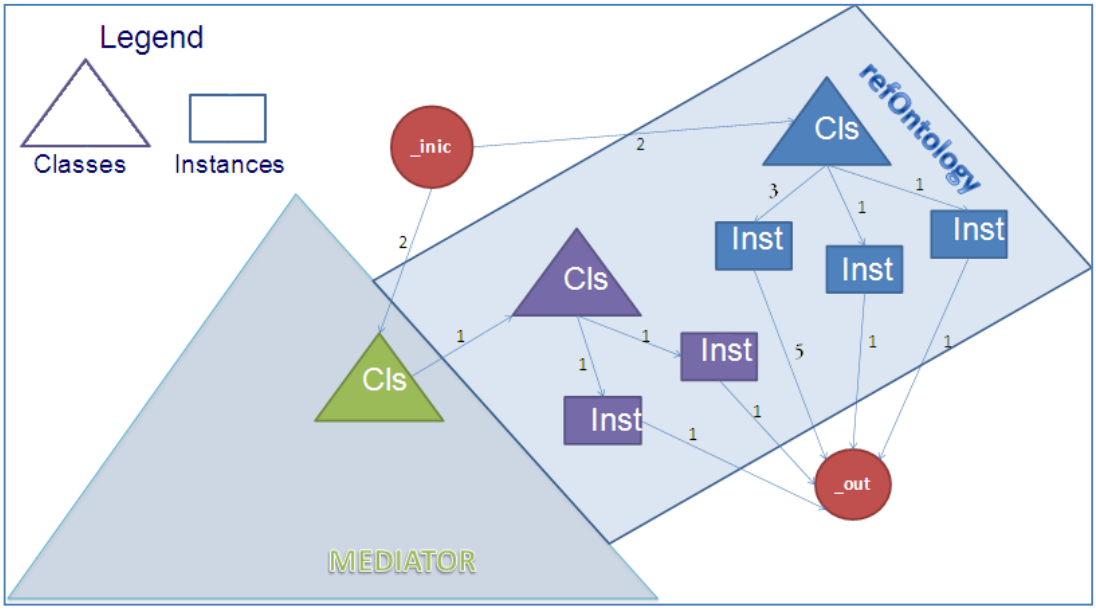

Figure 5.2: Graph representation

Note that properties are not represented because, as mentioned before, oLEARCH returns their classes instead. It is also possible to see that only instances can be connected to element "_out". This is because oLEARCH will only return instances (products).

The returned results are presented by its weight order, using Dijkstra algorithm for this decision. Usually Dijkstra is used in graphs to find the closest way. What oLEARCH does is to invert weights in order to have the biggest one with lower value. This will make the higher weights will come first than lower ones. This weight's transformation in distance basically consists in inverting all existing weights using the following formula:

$$\text{newWeight} = \text{maxWeight} + 1 - \text{oldWeight}$$

Figure 5.3 demonstrates the resulting scenario after this weight transformation be applied to the scenario shown in Figure 5.2.

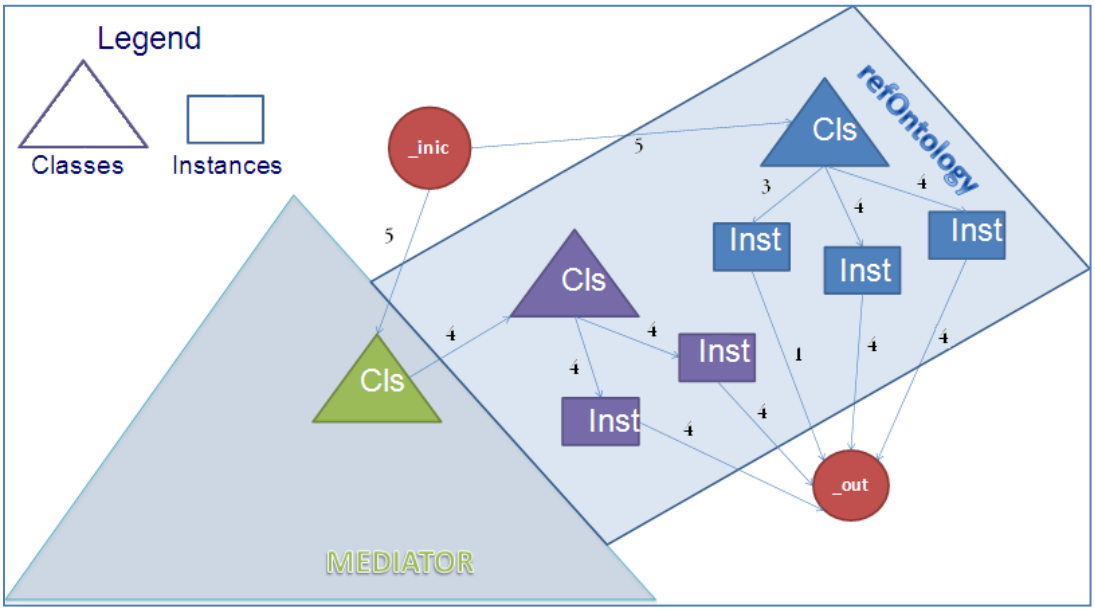

Figure 5.3: Graph representation after weight transformation

It is now possible to apply Dijkstra algorithm to the existing graph. It will be applied as many times as the value settled in the administrator GUI, Figure 5.9, returning each times a different product.

Next slides represent a practical example of the explained theory used in oLEARCH. Figure 5.5 presents part of the connections established when searching for “Chairs” starting in the initial oLEARCH state which is shown in Figure 5.4.

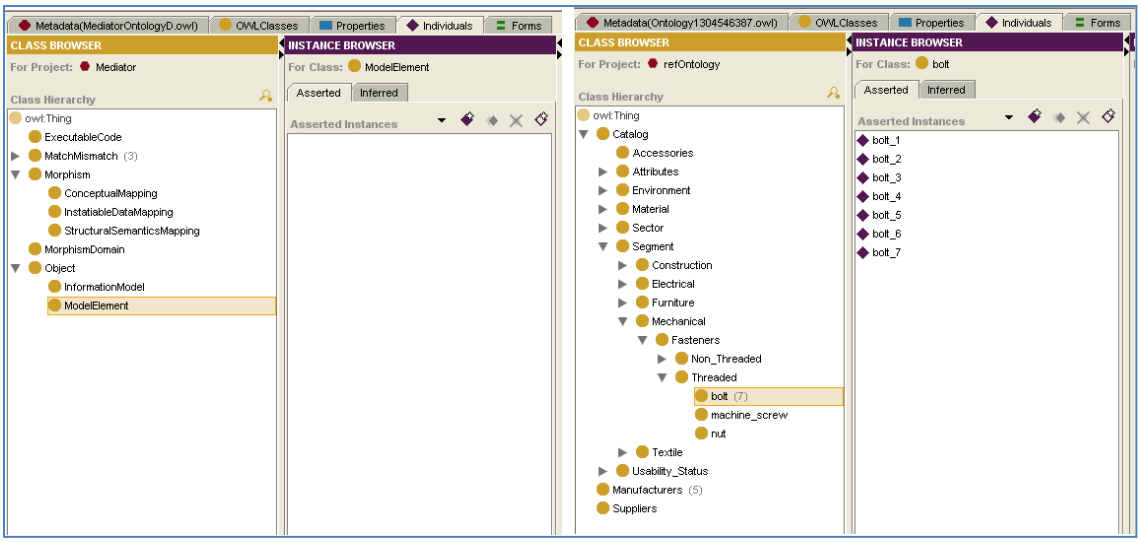

Figure 5.4: Mediator ontology and reference ontology.

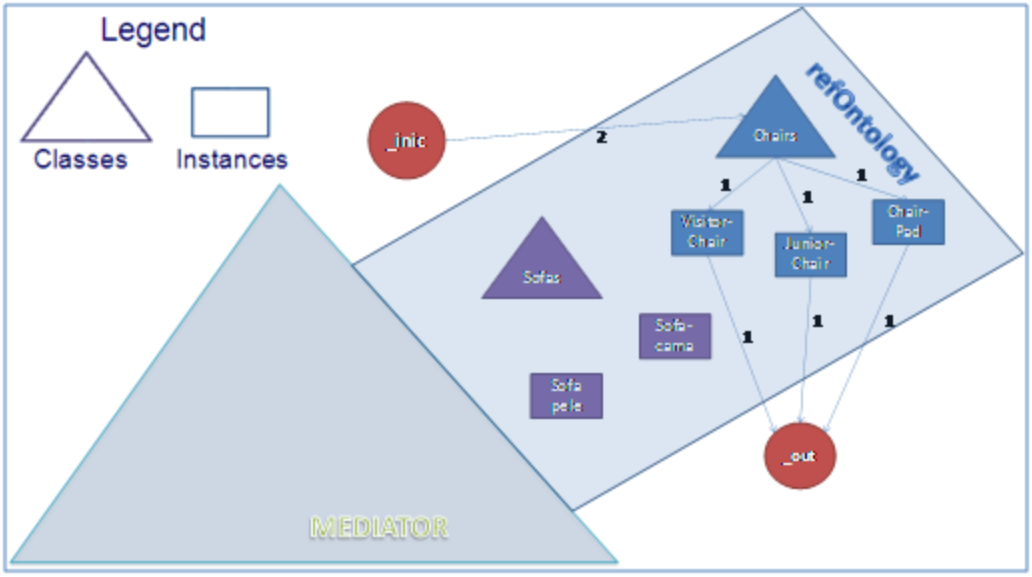

Figure 5.5: Graph created when user searched for “Chairs”.

Then, Figure 5.6 shows graph result when the user searches for “Couch”, which doesn’t exist in the knowledge base.

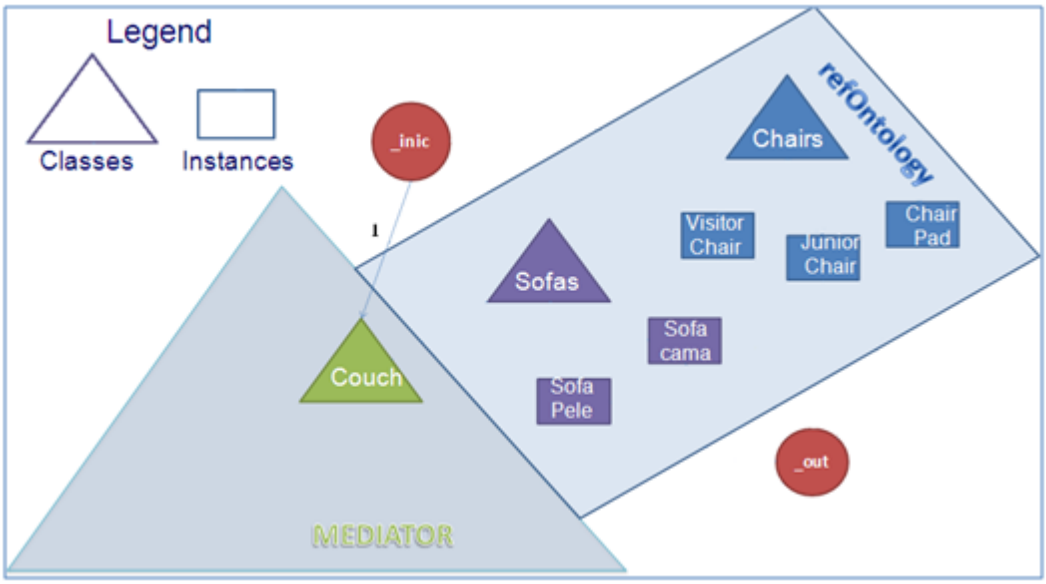

Figure 5.6: Graph created when user searched for “Couch”.

Finally, after the user create a relation, searching for “Couch + Sofas”, Figure 5.7 scenario represents a “Couch + Chairs” search. It is possible not only to see the new relations that are created but also the respective weights increasing.

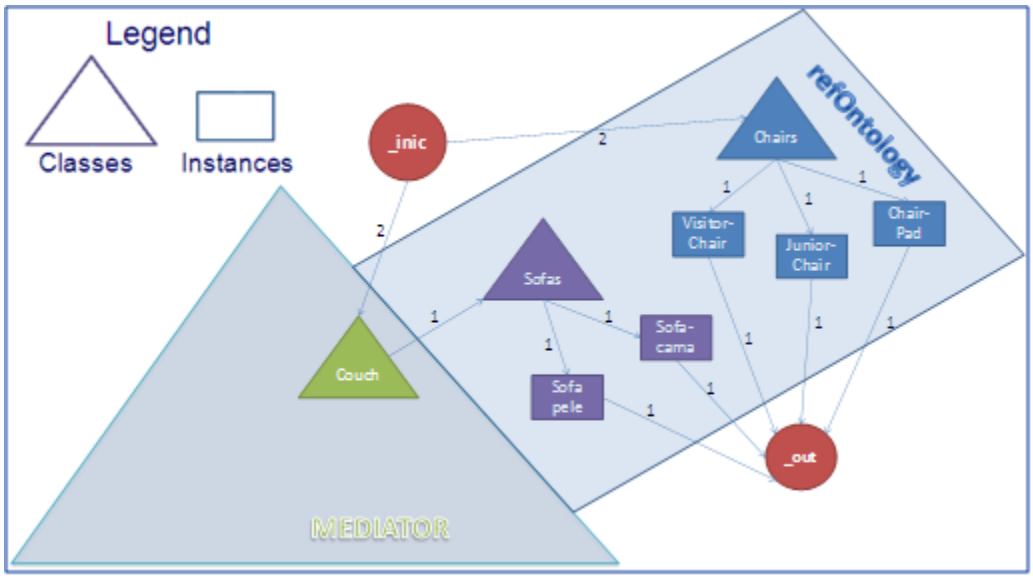

Figure 5.7: Graph created when user searched for “Couch + Chairs”.

After this process the weight transformation in distances is applied and the result is displayed in Figure 5.8.

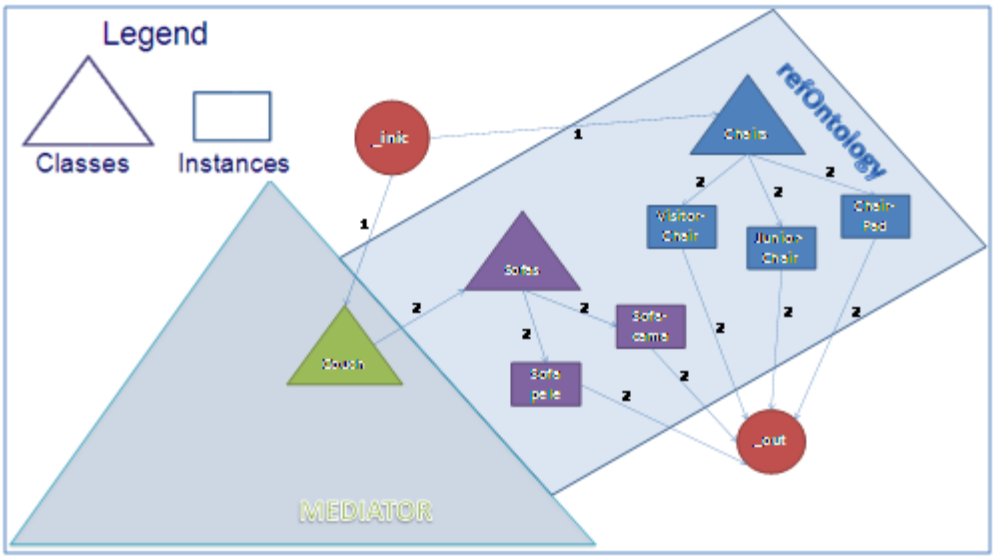

Figure 5.8: Graph after weight transformation.

After transformation Dijkstra algorithm is applied, returning all the products, till maximum products settings, starting first from the one with lower weight. From the example of Figure 5.8 the first products returning are the chairs because they all have weight sum, from “_inic” till “_out”, equal to 5 (less than Sofas which have weight sum equal to 7).

5.1.2. Increase weight service

After the searched products are returned to the user, he might select the desired one.

This selection activates the second service which is used to increment the product weight that he had chosen. It searches for the concept in the reference ontology and increases its weight. With this action, the selected product importance will grow which makes it appear first in future searches. This option can also be manually used in the oLEARCH administrator tool for testing purposes.

5.2. OLearch Administrator GUI

The oLEARCH Administrator GUI, Figure 5.9, is the administrator tool where it is possible to set some parameters such as the number of results to be displayed on the regular oLEARCH GUI, set the minimum value for a searching concept be considered as a possible new product or as a possible new concept and also to increase manually a concept weight. This tool has a display to show this possible new products and concepts.

This application also provides the possibility to search for an element, by pressing “Search” button.

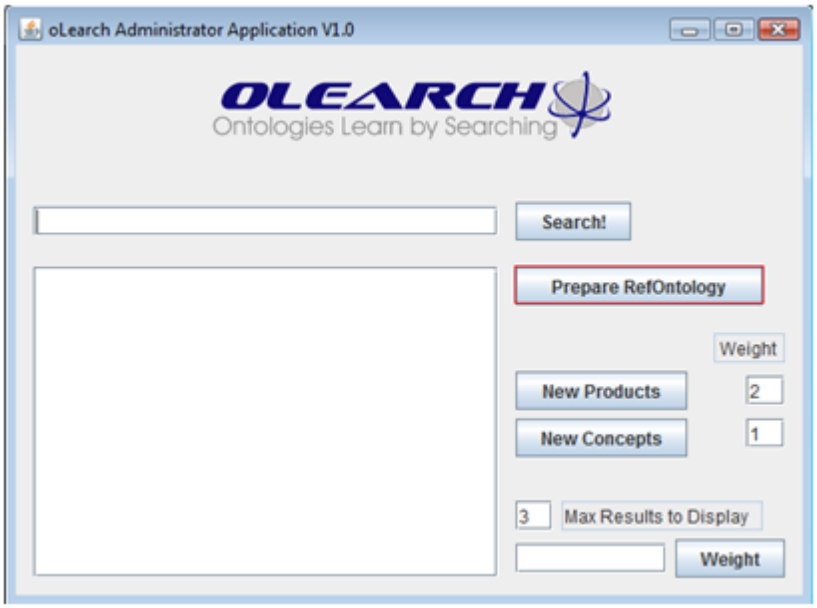

Figure 5.9: oLEARCH Administration application

There is other administrator functionality related to the button called “Prepare RefOntology” which prepares the reference ontology to be used in oLEARCH. Basically, it adds a property called “weight”, with value equal to one, to all classes and instances of the ontology. This weight is added in the comment property as shown in Figure 4.11. Also, in the mediator ontology it will create two new Information Models related to the chosen reference ontology data (PatternsInfoModel and refOntology shown in Figure 5.10), to prepare MO to receive information about searched terms,

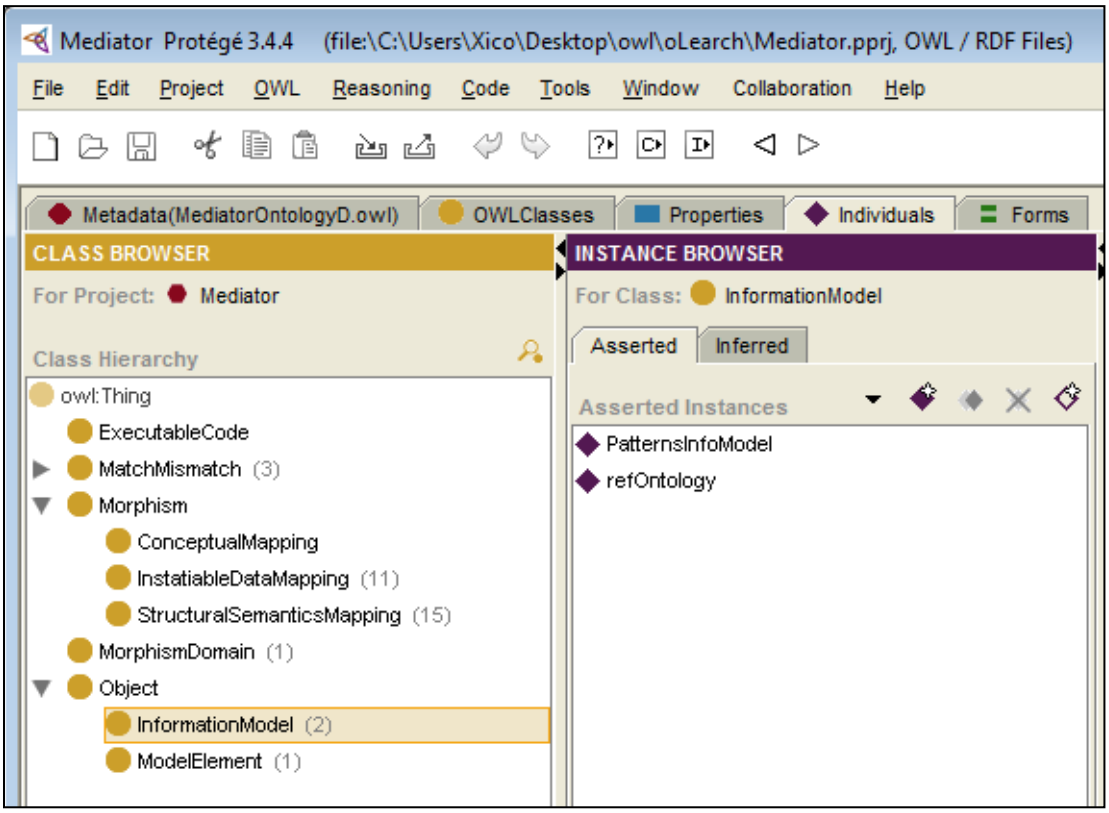

Figure 5.10: Information Models in Mediator.

5.2.1. New products

Possible new products are the concepts searched by users which don't exist in the reference ontology and which were not related to existing concepts. This will search in MEDIATOR for concepts that don't exist in the reference ontology and that were searched more than the minimum settled in the Administrator GUI.

If this value is defined for two, this means that when a word is searched more than two times without semantic relation to an existing one, it is probably a new product.

5.2.2. New concepts

New concepts are the concepts searched by users which have semantic equivalent in the reference ontology but are not the same. Using Figure 5.8 example, "Couch" has a relation with "Sofas" it is more used than "sofas", "couch" will become a new concept for sofas. OLearch implement this by getting all existing relations where one of the concepts doesn't exist in the reference ontology.

If administrator define one as a threshold for this functionality means that for a concept

be considered a new concept it just needs to be searched more than one time or have its weight bigger by one in the relation to the related concept of the reference ontology.

5.3. OLearch GUI

This application is a search engine built in JSP technology and using web services where users, with internet, can access it.

This GUI application is presented in Figure 5.11.

Figure 5.11: oLEARCH GUI.

This tool is the main oLEARCH application output. It returns from the reference ontology products related with the searched concepts.

The only function available is to search for a product. When the “SEARCH” button is pressed oLEARCH will compare the entered concepts with the knowledge base using the get products service explained before. When a result product is selected its weight is increased automatically.

A demonstration of this tool use is deeply described in the following chapter.

5.4. Conclusions

OLearch follow the classification learning thought the application of the instance-based learning. However due to its use of active statistics on the classification it can be also associated to other classification variant of the numeric prediction. As explained before in this dissertation, the decision to use Dijkstra’s algorithm was due to its simplicity, easy implementation in JAVA language, reliable and especially because it provides the desired results. There were just a few complications regarding some used technologies, specifically protégé-to java classes where it was found some issues already reported to protégé and SOAP which was a good challenge to implement in order to have the communication between JAVA language and web services.

6. DEMONSTRATOR TESTING AND HYPOTHESIS VALIDATION

OLearch aims to be an intelligent system able to record user's actions in order to collect data for knowledge management. This chapter describes the validation scenario for the architecture presented in the previous chapter, using an industrial case.

6.1. Search functionality and relation's creation.

Knowledge maintenance is related to advanced users that browse within a product family, requesting some specific aspects not available or visible to general public. Thus identifying key aspects that products should deploy but whose configuration or function was not a key point from design or manufacturer's point of view. As an example, choosing from a chair catalogue, only the items whose screw drive trade is a Philips or Umbraco driving feature is a promising enhancement. In oLEARCH when a user searches for a product and/or its properties it returns all the searched products matched and all the products using the entered properties. This is because if there's a product without those properties, the company should show also other products that might be interesting for the user.

Another example could be the attempt to identify the list of affected items when e.g. an M8 (kind of a bolt) should be present in some elements of furniture. Some of these points of view may sound strange to designer's perspective or manufacturer's commercial staff but have implicit knowledge that an intelligent system may track. This intelligent assistance could help on costumer's request optimisation; for example, some properties may be avoided or promoted according to customer needs when browsing product's catalogue, giving the feedback based in previous users' interaction, reducing selection time for customers and helping suppliers employees engaged in costumer assistance. In the following is described how oLEARCH deals with the results of the search functionality and relations creation.

To test the creation of relations between products, the author first searches for a product and then, to create a relation, he will search for two products at the same time. The first search keyword is "Chairs". This search returns all the Chairs existing in the Ontology, Figure 6.1.

Figure 6.1: Search results for keyword “Chairs”.

Then, the search keyword is “Chairs+Mats” which returns all Chairs and Mats in the Ontology, as presented in Figure 6.2. The reason why chairs are displayed before the mats it is due to last search, which had increase “Chairs” weight. In the Mediator, we have now a new relation between “Chairs” and “Mats”.

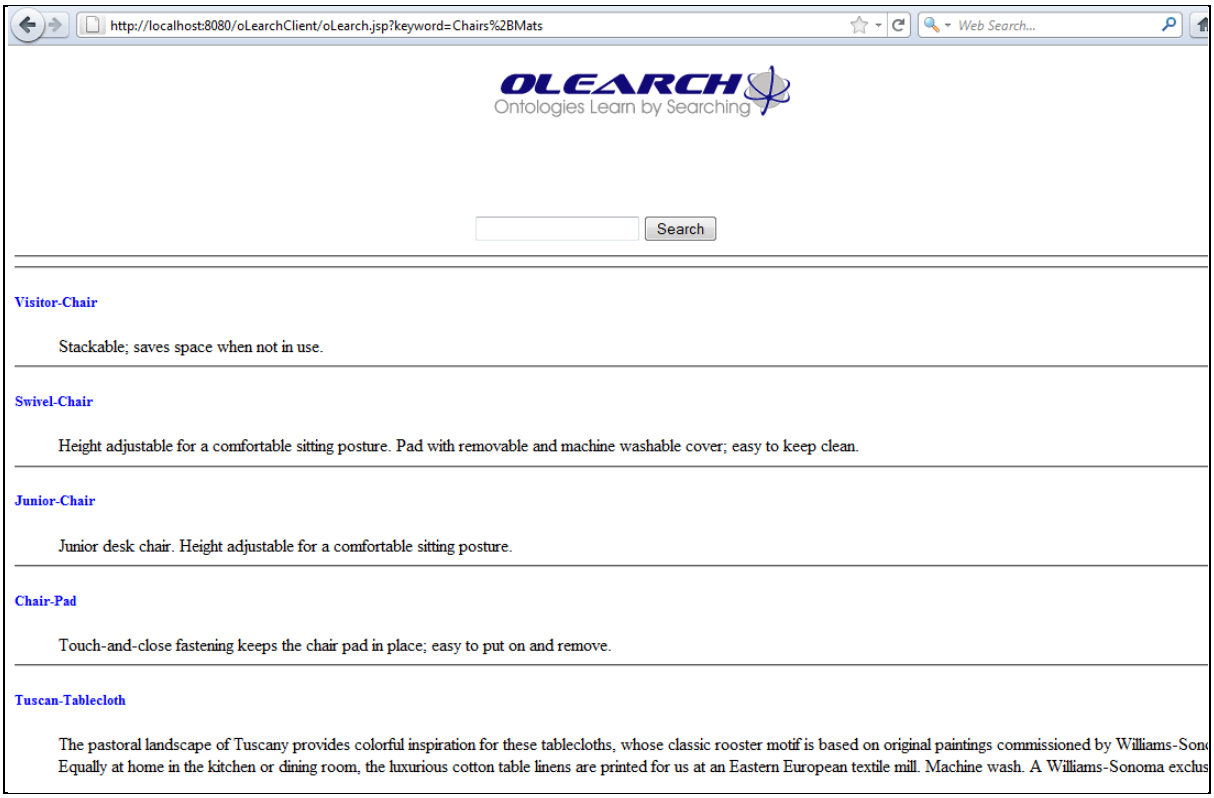

Figure 6.2: Search results for keywords “Chairs+Mats”.

Repeating the search using keyword “Chairs”, the result is the expected one that oLEARCH returned in Figure 6.3, not only chairs but also mats.

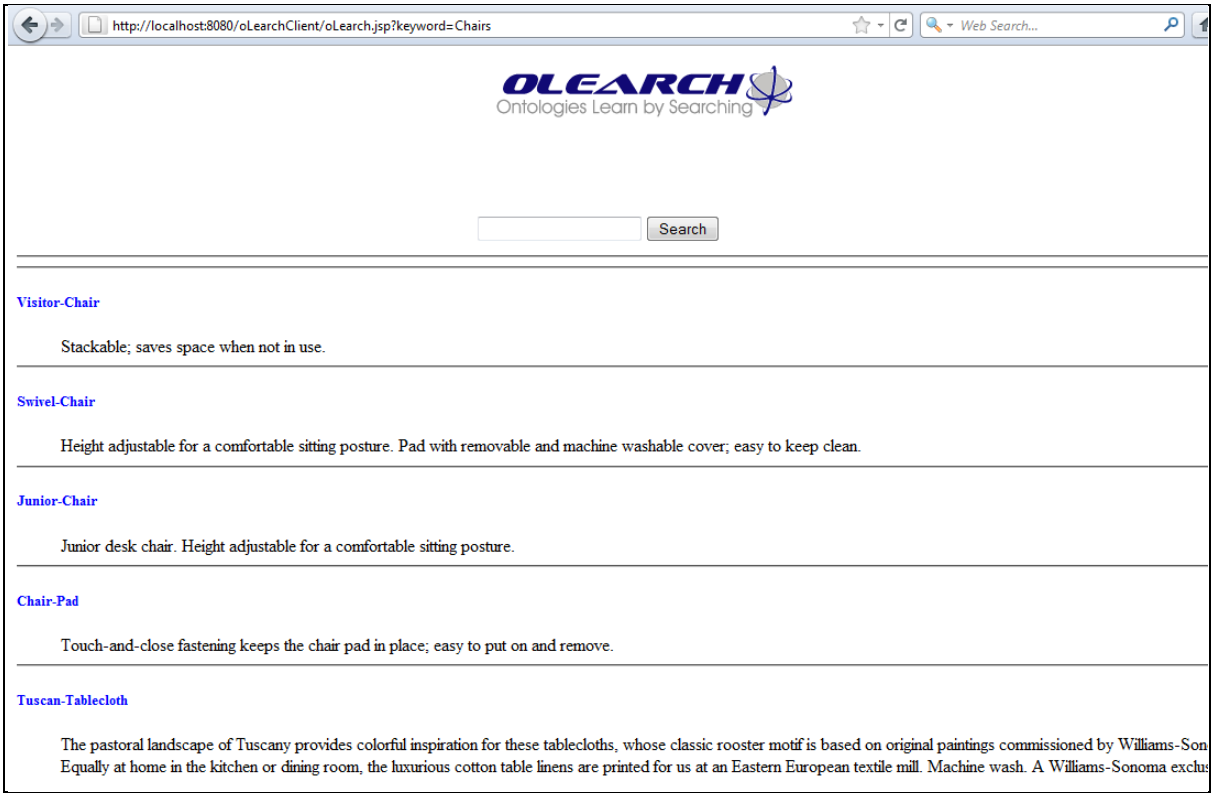

Figure 6.3: Search results for keyword “Chairs”.

This result is due to the relation that was created in the mediator, by the previous search, where “Chairs” was connected to “Mats”. In this scenario if “Mats” gains importance (weight) till it’s higher than “Chairs”, then it will appear first in the search result, even if the user searches only for “Chairs”.

Figure 6.4 and Figure 6.5 show two extra tests. The first one is that searching, for example, “Mats” and/or for “Chair-Pad” several times will increase its weight. This weight increase will cause a change position when displaying the results. So searching again for “Chairs” the result will have a chair-pad product before some chairs.

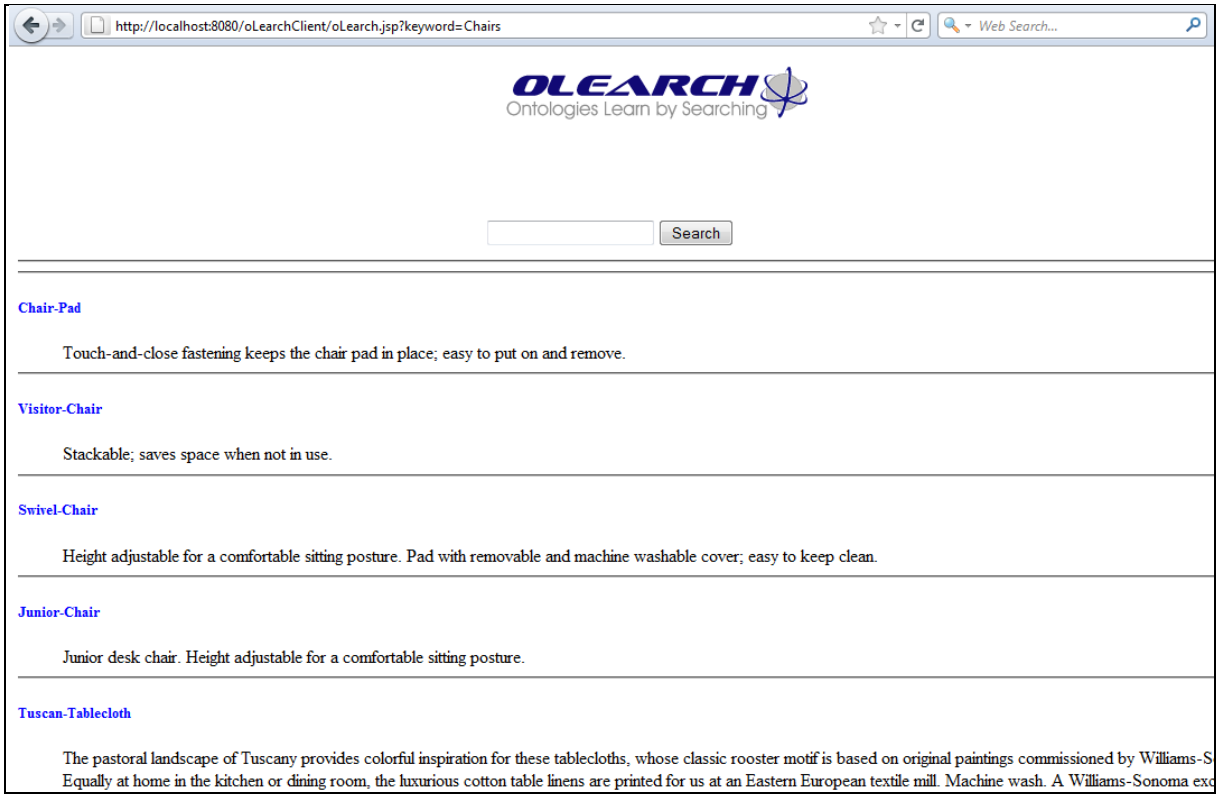

Figure 6.4: Search results for keyword “Chair-Pad” after searched several times.

Clicking on the product “Junior-chair” will lead to the second extra test which shows that searching again for the same keyword, the product “Junior-chair” is shown first than it was. This is presented in Figure 6.5.

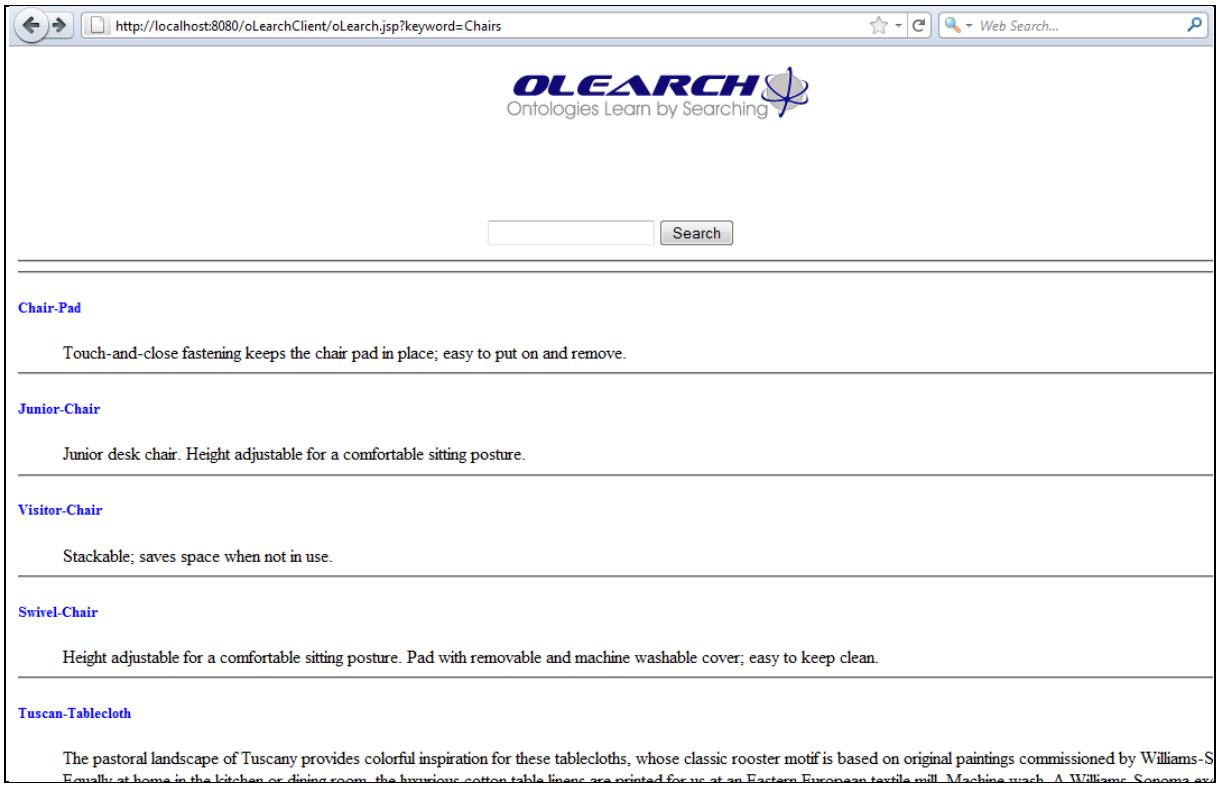

Figure 6.5: Search result for keyword “Junior-chair” after selected from the last search result.

This happens because of the searching frequency. It could be also explained as the following. Imagine that the weight of product A is three, weight of product B is two and weight of product C is one. They are all related to each other. Searching for A, the result will be A then B then C. After this search, product A has weight equal to four. If product C is searched two times, its weight will be equal to three. Now, when searching again for product A, the result will be A (with weight equal to five), C (with weight equal to three) and B (with weight equal to two).

6.2. New concepts Functionality

Considering that a client not familiar with the webservice starts browsing the furniture domain database, using terms not existent in reference ontology. Those terms may be considered either a mismatch or a different jargon that such client habitually uses. The track of such behaviour may induce the system to react ignoring repeatedly the entry or wisely understand that the term is new for the reference ontology but could be the new term to be used in near future. This case represents time and money saved since it avoids domain engineers meet constantly with knowledge engineers to discuss enterprise’s knowledge update as for new terms inclusion in the reference ontology. With this procedure, errors will be also reduced since the system could dynamically learn that a new term introduced by

users could initially be considered similar/equal in semantics to an existent one.

The used reference ontology has some sofas, although if search key word is “Couch”, oLEARCH won’t return any product. Searching for “Couch+Sofas” it will return all the sofas. Then if other users search for “couch” more than for “sofas” will result in a “couch” weight higher than “sofas” weight. Thus, if the “new concepts” threshold is set to five “couch” will be considered a new concept if it has a positive difference to “sofas” related weight of at least 6.

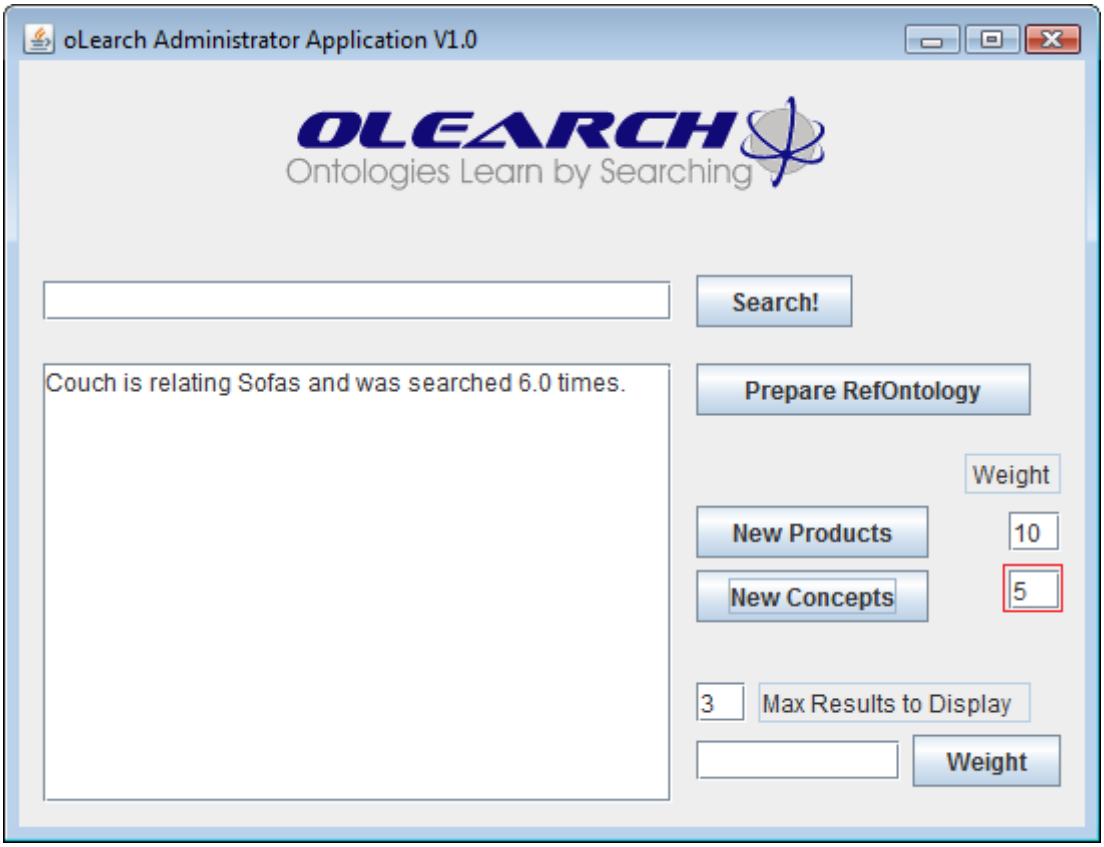

Figure 6.6: oLEARCH Administration Application result for “New Concepts” button.

6.3. New products functionality

Finally, another scenario is the search using a term for an inexistent product, moreover with combined specifications or others details. It is expected that the system intelligently react and suggest MENTOR managers that some kind of new product has been requested (and design teams should be aware). In this case, the added value is more related to the time saved that could also represent money saved, because if there is any new product on the market, enterprises aim to react promptly. This system is able to warn design teams for a possible new product appetence, anticipating precious market information to the company.

For this test scenario, ‘New Products’ weight is set to 10, as shown in last picture. This means that if a concept is searched more than ten times and it doesn’t exist on the ontology it is probably a new product.

Figure 6.7 presents a possible new product. As product “Couch” doesn’t exist, simply searching for it ten times and then clicking on ‘New Products’ button it will return it as a possible new product.

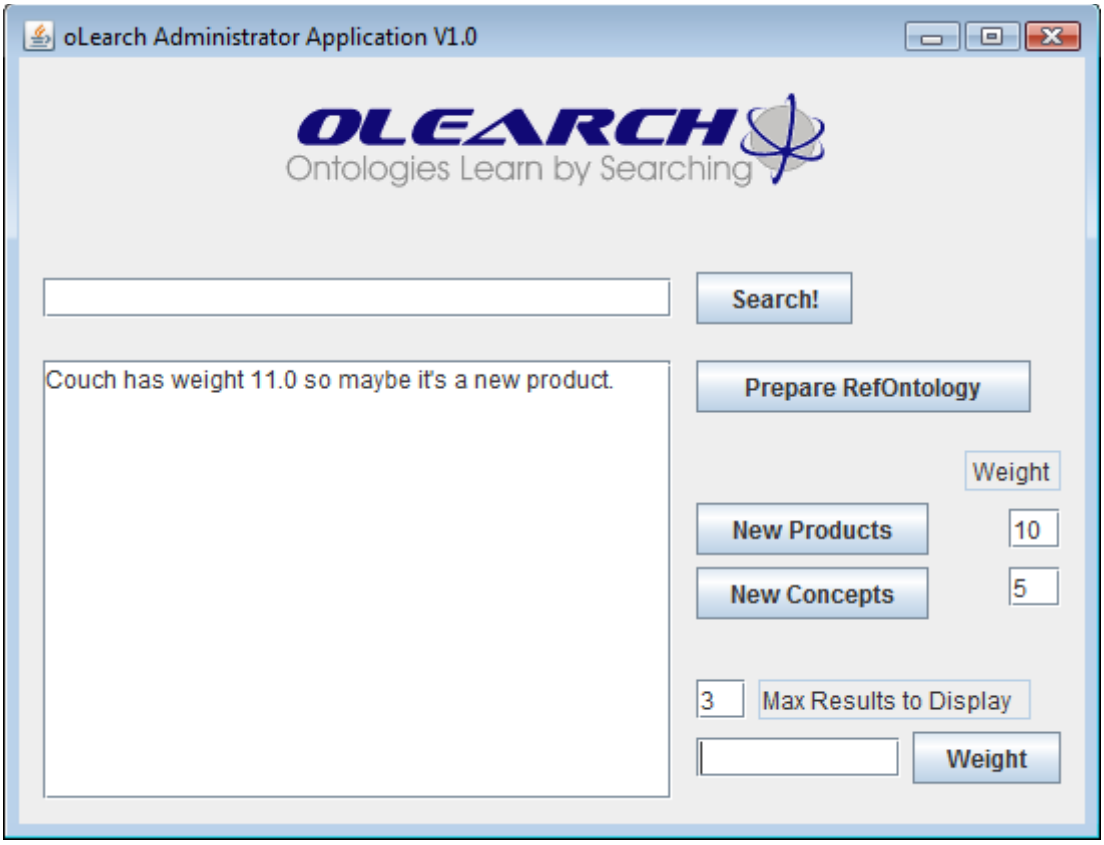

Figure 6.7: oLEARCH Administration Application result for “New Products” button.

6.4. Dissemination Executed and Hypothesis Validation

As defined in the first dissertation hypothesis, oLEARCH makes use of ontologies with some statistics associated to its concept and operational research methods. This facilitates to acquire knowledge from users in the sense that their introduced concepts received a statistic number related to its use frequency. Such process helped oLEARCH to propose new products and concepts, thus it results in an increase of the intelligence of its information system management.

Following the second dissertation hypothesis, oLEARCH also used mapping between reference ontology and Mediator ontology. It worked as a solution to facilitate further

knowledge evolution. A mapping between a “reference” concept and a “new” concept introduced by a user is used to propose MENTOR administrators a reconsideration of the reference concept. This results in a knowledge evolution. Ontology mapping confirmed to be a suitable solution for ontology enhancement. The presented methodology might be applied in any enterprise no matter its domain, as oLEARCH is able to interact between MEDIATOR and any reference ontology.

For intentional purposes of the research results of this dissertation, a scientific publication was accepted in the ASME International Mechanical Engineering Congress and Exposition, from 11th to 17th of November 2011 in Denver – United Stated of America, and it was published on the proceedings of the conference:

- Sarraipa J., Jardim-Gonçalves R., Mendonça da Silva J., Cavaco F., Knowledge Based Methodology Supporting Interoperability Increase in Manufacture Domain, Accepted In: ASME International Mechanical Engineering Congress and Exposition. Nov 11-17, Denver, United States of America, (2011).

.

7. CONCLUSIONS AND FUTURE WORK

To better respond to the requirements of the market, enterprises are changing the way they do business. SMEs started to realize that the small markets do not bring big benefits for them and do not allow them to grow as fast as they wish. To compete with large enterprises, SMEs must seek for collaboration between each other, e.g. using virtual enterprises organisations.

Interoperability is the keyword for success in such enterprise collaboration environments. When working together, enterprises need to communicate to understand each one's information. But most of the time these communications are not well succeeded due to semantic interoperability problems.

Tacit knowledge acquisition process isn't easy and clear. Furthermore, the knowledge maintenance stage should focus on the improvement of the knowledge base in order to be alive and updated accordingly to the knowledge evolution to which the system could be related to. This dissertation propose a knowledge maintenance stage to MENTOR in order to improve the management of the knowledge base along product life cycle in order to be adapted accordingly to the knowledge evolution of the system. Knowledge maintenance is ruled by the analysis of the users' interactions feedback, which works as the main trigger to the learning process on which such knowledge-based system is based on.

The knowledge maintenance is assuming a key role to dynamically update reference ontology, since enterprises environments and company's alliances are changing fast in today's competitive market. Based in knowledge database user's interaction, as an intelligent response to knowledge management and update, good results are expected to a major update of reference ontology within its specific domain.

OLEARCH tool is then the solution found by the author to tackle these problems. With this tool companies can easily monitor the constantly evolving concepts as well as monitor the trend of new products, getting thereby a better usability of the available resources. It is important to state again that all these dynamic behaviours handled by this knowledge maintenance solution results in an automatic contribution to reference ontology refinement and knowledge management enhancement.

OLEARCH development contributed to answer this dissertation research questions proving at the same time its proposed hypothesis. Knowledge acquisition from information system's users was enhanced due to the oLEARCH integration on the presented knowledge based methodology for semantic interoperability. It acts directly with users saving their introduced knowledge in a formal way in the Mediator ontology and integrating it with

previous existent knowledge of the system, which is formal represented in the reference ontology. Moreover, oLEARCH is able to learn through the users' feedback/interactions, representing in this way, an enhanced system able to acquire and learn from the users introduced knowledge. OLEARCH is also integrated with a specific algorithm that acts as an operational research method because it uses a statistical approach based on users' patterns, which is used for decision making related to the choice of the best product considering the knowledge inserted by the users. This process provides oLEARCH with a machine learning ability, not only able to return better search results, but as well as to propose new concepts and new products that contributes to the knowledge maintenance of the system. These learning abilities assist the system intelligence increase.

OLEARCH manages the knowledge by establishing mappings between concepts, for instance a "reference" existent concept and a "new" concept introduced by a user. This is used to make equivalences in reasoning related products from the reference ontology. It is also used in some patterns analysis like related to concepts use frequency. A higher use frequency of a "new" concept in comparison to a "reference" concept, will propose MENTOR administrators a reconsideration of the "reference" concept. Thus, it could be concluded that the reference ontology mapping with MEDIATOR ontology works as a facilitator to enhance knowledge re-use and adaptation, facilitating also its system intelligence improvement.

7.1. Future Work

As for future work, a few things regarding the implementation of the solution may be introduced. Starting with the Administrator GUI, it might have more functionalities related to the users patterns analysis, which could result in more knowledge management solutions for administrators. For example, if it has a registration procedure able to record personal data from users, then oLEARCH could have access to the top requested products by age, by city, etc. This could take enterprises to another level of performance regarding customer's requests analysis. The integration MENTOR-oLEARCH is other point that should be improved to better integrate knowledge changes/adaptation in the reference ontology resulted from the oLEARCH patterns analysis.

With actual technologic development and specially the fast grow of mobile devices it could be good to provide them the access to oLEARCH. Another future implementation is regarding to some basic knowledge base management, as creation or deletion of products. It would avoid wasting system memory with non-existing products (for example semantic mistakes) or to create new ones automatically. Also relevant could be provide a functionality that after a user selects a product, he would be redirected to the closest shop or the product would be sent to his address.

8. REFERENCES

[1] Camarinha-Matos L.: Scientific Research Methodologies and Techniques - Unit 2: Scientific Method, PhD Program in Electrical and Cumputer Engineering (2010).
[2] Sarraipa, J.: Uma solução para a Interoperabilidade Semantica em ambientes globais de negócios. Master Thesis, (2004).
[3] Stuart J. Russell; and Peter Norvig (1995). Artificial Intelligence: A Modern Approach, Prentice Hall, Englewood Cliffs, NJ (1995).
[4] Terrence J. Sejnowski; Christof Koch; Patricia S. Churchland (1988). Computational Neuroscience, Science, New Series, Vol. 241, No. 4871. (Sep. 9, 1988), pp. 1299-1306, (1988).
[5] D. Purves, G. J. Augustine, D. Fitzpatrick, et al., Neuroscience, Sinauer Associates, Sunderland, Mass, USA, 3rd edition, 2004.
[6] CNSORG, 2011. Retrieved from the web at June 2011: http://www.cnsorg.org/computational-neuroscience.
[7] Tulving, E. in Organisation of Memory (eds Tulving, E. & Donaldson, W.) 381-403 (Academic, New York, 1972).
[8] Patterson, K.; Nestor, P. J.; and Rogers, T. T. (2007). Where do you know what you know? The representation of semantic knowledge in the human brain. In: Nature Reviews Neuroscience 8, 976-987 (December 2007).
[9] Koch C. 1999. Biophysics of Computation. Information Processing in Single Neurons. Oxford, UK: Oxford Univ. Press. 562 pp.
[10] Rolls ET, Treves A. 1998. Neural Networks and Brain Function. Oxford, UK: Oxford Univ. Press. 418 pp.
[11] Rolls ET. (2000). Memory systems in the brain. In: Annu Rev Psychol. 2000; 51:599-630
[12] Wikipedia, 2011. Retrieved from the web: http://en.wikipedia.org/wiki/Hebbian_theory, at June 2011.
[13] V. Bombardier, C. Mazaud, P. Lhoste and R. Vogrig, “Contribution of fuzzy reasoning method to knowledge integration in a defect recognition system,” Computer in Industry, In Press, 2006.
[14] Christos Stergiou and Dimitrios Siganos. “Neural Networks” , Imperial College of London, Computer Department; http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html 02-05-2010
[15] Simon S. Haykin (2008). In the book: Neural networks and learning machines, 2008.
[16] Sarraipa, J.: Uma solução para a Interoperabilidade Semantica em ambientes globais de negócios. Master Thesis, (2004).
[17] Sarraipa, J.; Jardim-Gonçalves, R.: Semantic Adaptability for Systems Interoperability.
[18] Gruber, T.: Toward Principles for the Design of Ontologies Used for Knowledge Sharing. Originally in N. Guarino and R. Poli, (Eds.), International Workshop on Formal Ontology, Padova, Italy. Revised August 1993. Published in International Journal of Human-Computer Studies, Volume 43, Issue 5-6 Nov/Dec. 1995, pp 907-928, (1993).
[19] INTEROP NoE. Deliverable MoMo.2 - TG MoMo Roadmap. InterOP, 2005.
[20] Fishwick, P.; Miller, J., Ontologies for Modeling and Simulation: Issues and Approaches, Proceedings of the 2004 Winter Simulation Conference, 2004.
[21] Saias, J., Uma Metodologia para a construção automática de Ontologias e a sua aplicação em Sistemas de Recuperação de Informação, 2003
[22] Maedche, Alexander; Staab, Steffen (2001). "Ontology Learning for the Semantic Web," IEEE Intelligent Systems, vol. 16, no. 2, pp. 72-79, March/April, 2001.
[23] Sváb, O.; Svátek, V. (2007).Ontology Mapping enhanced using Bayesian Networks. Retrieved from the web at August 2010: http://nb.vse.cz/~svatek/znal07.pdf
[24] Ontology Matching (2011). Retrieved from the web: http://www.ontologymatching.org/, at June 2011.
[25] Albagli, S.; Ben-Eliyahu-Zohary, R.; Shimony, S., Markov Network based Ontology Matching, In Proceedings of the Twenty-First International Conference on Artificial Intelligence, 2009.
[26] Morse, P. M.; and G. E. Kimball: “Methods of Operations Research”, John Wiley & Sons, Inc., New York, 1951.
[27] L. Zhou, "Ontology learning: state of the art and open issues," Information Technology and Management, published unline 24 March 2007.

[28] Alexander Maedche, Steffen Staab, "Ontology Learning for the Semantic Web," IEEE Intelligent Systems, vol. 16, no. 2, pp. 72-79, March/April, 2001.
[29] Lanchester, Frederick William: Mathematics in Warfare, in J. B. Newman, “The World of Mathematics”, vol. 4, Simon and Schuster, Inc., New York, 1956.
[30] Wikipedia, 2011. Retrieved from the web: http://en.wikipedia.org/wiki/Operations_research, at June 2011.
[31] Craven, B. D. and Islam, SMN, Operations Research Methods – Related Production, Distribution and Inventory Management Applications, The Icfai University Press, Hyderabad, (2006).
[32] Bls, 2011. Retrieved from the web: http://www.bls.gov/oco/ocos044.htm, at July 2011.
[33] Labyrinth, 2011. Retrieved from the web: http://www.labyrinth.net.au/~bdc/ORMfirst.pdf, at July 2011.
[34] Oliveira, P.; and Gomes, P. (2009). Probabilistic Reasoning in the Semantic Web using Markov Logic. MSc Thesis - Knowledge and Intelligent Systems Laboratory - Cognitive and Media Systems Group - Centre for Informatics and Systems of the University of Coimbra.
[35] P. Domingos, S. Kok, D. Lowd, H. Poon, M. Richardson, and P. Singla, “Markov Logic,” Probabilistic Inductive Logic Programming, 2008, pp. 92-117.
[36] Mollinari, M., Aplicações das Cadeias de Markov na Genética, Seminários em Genética e Melhoramento de Plantas – LGN 5799, 2007.
[37] Page, Lawrence and Brin, Sergey and Motwani, Rajeev and Winograd, Terry (1999). The PageRank Citation Ranking: Bringing Order to the Web. Technical Report
[38] Albagli, S.; Ben-Eliyahu-Zohary, R.; and Shimony, S. E. (2009). Markov network based ontology matching. In: Proceedings of the 21st international Joint Conference On Artificial Intelligence - Pasadena, California, USA
[39] Vojtěch Svátek, Ontology Mapping enhanced using Bayesian Networks, 2006.
[40] Colin Harris and Vinny Cahill. Power management for stationary machines in a pervasive computing environment. In Proceedings of the Hawaii International Conference on System Sciences, 2005.
[41] Gustavo Arroyo-Figueroa and Luis Sucar. A temporal bayesian network for diagnosis and prediction. In Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99), pages 13–20, San Francisco, CA, 1999. Morgan Kaufmann Publishers.
[42] Ann Devitt, Boris Danev and Katarina Matusikova, Ontology-driven Automatic Construction of Bayesian Networks for Telecommunication Network Management, 2006.
[43] Ding, Z.; Peng, Y.; and Pan, R. November 2004. A Bayesian Approach to Uncertainty Modeling in OWL Ontology. In Proceedings of 2004 International Conference on Advances in Intelligent Systems - Theory and Applications (AISTA2004). Luxembourg- Kirchberg, Luxembourg.
[44] Ding, Z.; and Peng, Y. January 2004. A Probabilistic Extension to Ontology Language OWL. In Proceedings of the 37th Hawaii International Conference on System Sciences (HICSS-37). Big Island, Hawaii.
[45] Fenz, S.; and Neubauer, T. (2009). "How to Determine Threat Probabilities Using Ontologies and Bayesian Networks". Talk: 5th Annual Workshop on Cyber Security and Information Intelligence Research, Knoxville, TN; 04-13-2009 – 04-15-2009; in: "CSIIRW ‘09: Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research", ACM New York, Ny, Usa (2009), ISBN: 978-1-60558-518-5; 1 – 11.
[46] Wikipedia, 2011. Retrieved from the web: http://en.wikipedia.org/wiki/Machine_learning, at August 2011.
[47] Rob Schapire (2009). Princeton: Retrieved from the web at June 2011: http://www.cs.princeton.edu/courses/archive/spr08/cos511/scribe_notes/0204.pdf
[48] A. Gammerman and V. Vovk. Hedging predictions in machine learning: The second computer journal lecture. The Computer Journal, 50(2):151--163, 2007.
[49] I. H. Witten; and E. Frank, Data Mining: Practical Machine Learning Tools and Techniques (2nd edition), Morgan Kaufmann, San Francisco, 2005.
[50] Frank Keller. Connectionist and Statistical Language Processing (Lecture Slides). Computerlinguistik, Universitat des Saarlandes, Saarbrucken, 2001.
[51] R. Agrawal, T. Imielinski, and A. N. Swami. Mining Association Rules between Sets of Items in Large Databases. In Proc. of the ACM SIGMOD Intl. Conf. on Management of Data (SIGMOD’93), 1993.
[52] J. Hipp, U. Gu¨ ntzer, and G. Nakhaeizadeh, “Algorithms for Association Rule Mining—A General Survey and Comparison,” SIGKDD Explorations, vol. 2, no. 1, pp. 58-64, July 2000.

[53] X. Wu, V. Kumar, J.R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G.J. McLachlan, A.F.M. Ng, B. Liu, P.S. Yu, Z.-H. Zhou, M. Steinbach, D.J. Hand, and D. Steinberg, "Top 10 Algorithms in Data Mining," Knowledge and Information Systems, vol. 14, no. 1, pp. 1-37, 2008.
[54] Nguyen, N., Guo, Y.: Metric Learning: A Support Vector Approach. Machine Learning and Knowledge Discovery in Databases, 125–136 (2008).
[55] Anton Milev: KD Tree - Searching in N-dimensions, Part I. Retrieved from the web in January 2010: http://www.codeproject.com/KB/architecture/KDTree.aspx.
[56] Gartner, 2011. Retrieved from the web: http://www.gartner.com/technology/it-glossary/, at July 2011.
[57] Slidefinder, 2011. Retrieved from the web: http://www.slidefinder.net/D/Data_Mining_Timo_Knuutila_Department/994490, at March 2011.
[58] Anderson.ucla, 2011. Retrieved from the web: http://www.anderson.ucla.edu/faculty/jason.frand/teacher/technologies/palace/datamining.htm, at August 2011.
[59] Dijkstra, E. W. (1959). "A note on two problems in connexion with graphs". Numerische Mathematik 1: 269–271, 1959.
[60] Wikipedia, 2011. Retrieved from the web: http://en.wikipedia.org/wiki/Dijkstra's_algorithm, at January 2011.
[61] Commission of the European Communities. Communication from the Commission to the Council, The European Parliament, The European Economic and Social Committee and the Committee of the Regions "i2010 – A European Information Society for growth and employment" - {SEC(2005) 717}; Brussels. (2005).
[62] Jardim-Goncalves, R., Grilo, A., & Steiger, A. (2006). Challenging the interoperability between computers in industry with MDA and SOA. Computers in Industry, 57(8–9), 679–689.
[63] Panetto, H., Jardim-Gonçalves, R., & Pereira, C. (2006). EManufacturing and web-based technology for intelligent manufacturing and networked enterprise. Journal of Intelligent Manufacturing,17(6), 639–640.
[64] Agostinho, C.; Sarraipa, J., Goncalves, D., and Jardim-Goncalves, R. (2011). Tuple-based semantic and structural mapping for a sustainable interoperability. In: DOCEIS'11 2nd Doctoral Conference on Computing, Electrical and Industrial Systems –Costa de Caparica, Lisbon, February 2011.
[65] IEEE Standard Computer Dictionary (1990). In: A Compilation of IEEE Standard Computer Glossaries; New York, NY: 1990.
[66] Tolk, A., S. Y. Diallo, C. Turnitsa, L. S. Winters. (2006). Composable M&S Web Services for Net-centric Applications. Journal Defense Modeling and Simulation 3 (1): 27-44.
[67] Turnitsa, C.; Tolk, A. (2008). Knowledge Representation and the Dimensions of a Multi-Model Relationship. In Proceedings of the 40th Winter Simulation Conference, 2008: 1148-1156.
[68] Oxford Dictionaries: http://oxforddictionaries.com/view/entry/m_en_gb0447820#m_en_gb0447820.
[69] Bellinger, G., Castro, D., Mills, A. (2004). Data, Information, Knowledge and Wisdom, Retrieved from the web http://www.systems-thinking.org/dikw/dikw.htm at August 2010.
[70] Ackoff, R. L. (1989). "From Data to Wisdom", Journal of Applies Systems Analysis, Volume 16, 1989 p 3-9.
[71] Velthausz, D. (1998). In the Telematica Instituut Fundamental Research Series: Cost-effective network-based multimedia information retrieval. ISSN 1388-1795; Nº 003. ISBN 90-75176-16-3. Telematica Instituut, The Netherlands - 1998.
[72] Meystel, A.M. and Albus, J.S., (2001). Intelligent systems: architecture, design, and control. New York: John Wiley and Sons.
[73] Saridis, G.N. and Valavanis, K.P., (1988). Analytical design of intelligent machines. Automatica, 24 (2), 123–133.
[74] Zadeh, L.A., 1994. Fuzzy logic, neural networks, and soft computing. Communications to the ACM, 37 (3), 77–84. ISSN:0001-0782.
[75] Kasabov, N. and Filev, D., 2006. Evolving intelligent systems: methods, learning, & applications, Proc. Int. Symposium on Evolving Fuzzy Systems, pp. 8–18.
[76] Brewster, C. (2010). Ontology Learning. In Ontogenesis - An Ontology Tutorial – retrieved from the web at August 2010: http://ontogenesis.knowledgeblog.org/331.
[77] Gómez-Pérez, A. Staab, S. & Studer, R. (ed.) [Ontology Evaluation.] Handbook on Ontologies, Springer, 2004, 251-274

[78] Oltramari, A.; Gangemi, A.; Guarino, N. & Masolo, C. Restructuring WordNet's Top-Level: The OntoClean approach Proceedings of the Workshop OntoLex'2, Ontologies and Lexical Knowledge Bases, 2002
[79] Sarraipa, J.; Silva, J. P.; Jardim-Gonçalves, R.; Monteiro, A.: MENTOR – Methodology for Enterprise Reference Ontology Development. In: Intelligent Systems, 2008. (IS '08). 4th International IEEE Conference, pp 6-32 - 6-40, (2008)
[80] Sarraipa, J., Jardim-Goncalves, R. and Steiger-Garcao, A. (2010). MENTOR: an enabler for interoperable intelligent systems. International Journal of General Systems, Volume 39, Issue 5 July 2010 , pages 557 – 573.
[81] Ratliff, N.; Silver, D. (2009). Learning to Search: Functional Gradient Techniques for Imitation Learning. In Journal Autonomous Robots; Publisher: Springer Netherlands; 2009-07-01; Issn: 0929-5593; Issue 1; Pages 25-53
[82] Protégé, 2010. Retrieved from the web: http://protege.stanford.edu/, on 22nd July 2010.
[83] Mendonça da Silva J.; Cavaco F.; Sarraipa J., Jardim-Gonçalves R.; Knowledge Based Methodology Supporting Interoperability Increase in Manufacture Domain, Accepted In: ASME International Mechanical Engineering Congress and Exposition. Nov 11-17, Denver, United States of America, (2011).
[84] Protégé-OWL API Programmer's Guide, 2010. Retrieved from the web: http://protegewiki.stanford.edu/wiki/ProtegeOWL_API_ Programmers_Guide, on 7th August 2010.
[85] Collaborative Protégé, 2010. Retrieved from the web: http://protegewiki.stanford.edu/index.php/Collaborative_Protege, on December 2009.
[86] About Java Technology, available from http://www.sun.com/java/about/, accessed on 19th July 2010.
[87] Introduction to Web Services, available from http://www.w3schools.com/webservices/ /ws_intro.asp, accessed on 21st July 2010.
[88] IBM, 2010. Retrieved from the web: http://www.ibm.com/developerworks/webservices/library/ws-wsajax/, at November 2010.
[89] Escolacriatividade, 2010. Retrieved from the web: http://www.escolacriatividade.com/tutorial-jquery-o-que-e-e-como-usar-o-jquery/, at November 2010.
[90] W3schools, 2010. Retrieved from the web: http://www.w3schools.com/SOAP/soap_intro.asp, at November 2010.
[91] Java.sun, 2010. Retrieved from the web: http://java.sun.com/products/jsp/whitepaper.html, at November 2010.
[92] Protégé, 2010. Retrieved from the web: http://protege.stanford.edu/doc/sparql/, at November 2010.
[93] Topquadrant: Retrieved from the web: http://www.topquadrant.com/products/TB_Composer.html at November 2010.
[94] Sarraipa J., Zouggar N., Chen D and Jardim-Goncalves R., 2007, "Annotation for enterprise information management traceability," Proceeding of ASME 2007 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, Las Vegas, NV.
[95] Agostinho, C.; Sarraipa, J.; Goncalves, D.; Jardim-Goncalves, R.; Beca, M. (2011). Tuple-based Morphisms for E-Procurement Solutions. In the proceedings of IDETC/CIE ASME (2011) – International design Engineering Technical Conference & Computers and Information in Engineering Conference, 29-31 August 2011 – Washington, DC – USA.
[96] Agostinho, C.; Goncalves, D.; Jardim-Goncalves, R. (2011). Tuple-based semantic and structural mapping for a sustainable interoperability. In the proceedings of DoCEIS'11 – 2nd Doctoral Conference on Computing Electrical and Industrial Systems, 21-23 February 2011 – Costa da Caparica, Lisbon – Portugal.
[97] Sarraipa, J.; Jardim-Goncalves, R.; (2011). Knowledge-based System for Semantics Adaptability of Enterprises Information Systems. In the IWEI 2011 Third International IFIP Working Conference, "Interoperability and Future Internet for Next-Generation Enterprises.", 22-24 March 2011, Stockholm, Sweden.
[98] Agostinho, C. Correia, F. and Jardim-Goncalves, R., 2010, "Interoperability of Complex Business Networks by Language Independent Information Models",17th ISPE International Conference on Concurrent Engineering (CE 2010), Sep 6-10, Krakow, Poland.
[99] Sarraipa, J.; Zouggar, N.; Chen, D; Jardim-Goncalves, R. (2007). Annotation for Enterprise Information Management Traceability. In Proceedings of IDETC/CIE ASME (2007).

APPENDIX 1

package	class/java	Main Functions
mediator	Vários	Vários
ontoHandler	OntologyHandler	getElementsConceptualMappedWith(ElementA): LIST of Elements
ontoHandler	OntologyHandler	createModelElements (ListElements; InformationModel; type): void
ontoHandler	OntologyHandler	createMorphismOfConceptualType (ElementA; ElementB): void
ontoHandler	OntologyHandler	increaseConceptualMappingWeight (ElementA; ElementB): void
ontoHandler	OntologyHandler	isModelElementCreated(ElementA): boolean
ontoHandler	RefOntoHandler	prepareAllClassesAndIndividuals(OntologyRefURL):void
ontoHandler	RefOntoHandler	increaseWeightOfClass(ClassName):void
ontoHandler	RefOntoHandler	increaseWeightOfIndividual(ClassName):void
ontoHandler	RefOntoHandler	getPropertyDomains(NomedaPropriedade):ListofClasses
ontoHandler	RefOntoHandler	hasClassElement(name):boolean
ontoHandler	RefOntoHandler	hasInstanceElement(name):boolean
ontoHandler	RefOntoHandler	hasPropertyElement(name):boolean
ontoHandler	RefOntoHandler	cleanChildClassesFromRootClassesList(List of RootClasses):List of RootCLasses
machineLearning	Dijkstra	
machineLearning	Weightedgraph	
util	Graph	setOrigin(Origin:int):void
util	Graph	getOrigin(void):int
util	Graph	setDestiny(Destiny:int):void
util	Graph	getDestiny(void):int
util	Graph	setWeight(weight:int):void
util	Graph	getWeight(void):int
util	Graph	setAsInstance(void):void
util	Graph	isInstance(void):boolean

util	Products	setProductName
util	Products	setProductDescription
util	Products	setProductURL
util	Products	getProductName
util	Products	getProductDescription
util	Products	getProductURL
util	MorphismElements	setElementA(concept):void
util	MorphismElements	setElementB(concept):void
util	MorphismElements	getElementA():Concept
util	MorphismElements	getElementB():Concept
util	Label	setID(ID:int):void
util	Label	getID(void):int
util	Label	setName(Name:String):void
util	Label	getName(void):String
util	Label	setType(type:int):void
util	Label	getType(void):String
util	Label	setUri(uri:String):void
util	Label	getUri(void):String
util	Util	getIntroducedWords(input: String):List of Concepts
util	Util	createOwnGraph(ListOfRootClasses, ListOfRootInstances, ListOfRootMediatorClasses): (Vector of Labels) + (Vector of Graphs)
util	Util	inicConceptsToSearch(List of Concepts): List of "Morphisms"
util	Util	createDijkstraGraph(Vector of Labels, vector of Graphs): WeightedGraph
util	Util	transformWeightInDistance(VectorGraph: List of Graph):List of Graph

OntologyLearner	OntoLearn	getProductsRelatedWith(input:String; maxResults:int): List of (ProductName; ProductDescription; ProductURL)
OntologyLearner	OntoLearn	increaseWeightOfChosenProduct(ProductName):void
OntologyLearner	OntoLearn	showUsagePatterns(void):List of (List of Possible New Products)(List of Possible New Ref Concepts)
OntologyLearner	KnowledgeHandler	introductionOfSearchKnowledge(input:String):void
OntologyLearner	KnowledgeHandler	getSearchResults(maxResults): listOfProducts
OntologyLearner	KnowledgeHandler	setRootClasses(OwlNamedClass):void
OntologyLearner	KnowledgeHandler	setRootInstances(OwlNamedClass):void
OntologyLearner	KnowledgeHandler	setRootClassesFromMediator(Element):void
OntologyLearner	KnowledgeHandler	clearAllRootLists(void):void
OntologyLearner	KnowledgeHandler	searchOverallMediatorForRefClassesMappedWith(Concept):OwnGraphUntilRefClass -> Ref Class added to ListOfRoofClasses
OntologyLearner	KnowledgeHandler	searchRefClassMappedWith(Concept):OwnGraphUntilRefClass -> Ref Class added to ListOfRoofClasses
OntologyLearner	KnowledgeHandler	searchForProducts(WeightedGraph, maxResults): List of Products

APPENDIX 2

getElementsConceptualMappedWith(ElementA): LIST of Elements

```
[1]  public List getElementsConceptualMappedWith(MyFactory mf, String ElementA){
[2]
[3]        List lista = new ArrayList();
[4]        ModelElement thisModelElement = mf.getModelElement(ElementA);
[5]        if (thisModelElement != null){
[6]        Collection thisMECollection = thisModelElement.getHasMorphismDomain();
[7]        for (Iterator i = thisMECollection.iterator(); i.hasNext();){
[8]           MorphismDomain thisMEMorphismDomain = (MorphismDomain) i.next();
[9]           try{
[10]          Collection thisMEMorphismCollection =
      thisMEMorphismDomain.getAppliedMorphism();
[11]          for (Iterator j = thisMEMorphismCollection.iterator(); j.hasNext();){
[12]             ConceptualMapping thisMEConceptualMapping = (ConceptualMapping) j.next();
[13]             mediator.Object relatedObj =
      thisMEConceptualMapping.getRelating().getAssociatedObject();
[14]             if (relatedObj != null){
[15]             lista.add(relatedObj.getName());
[16]             }
[17]          }
[18]          }catch (Exception e){
[19]             System.out.println("getElementsConceptualMappedWith");
[20]          }
[21]          }
[22]   }
[23]             return lista;
[24]   }
```

createModelElements (ListElements; InformationModel; type): void

```
[25] public MyFactory createModelElements(MyFactory mf, List ListmodElement, InformationModel
[26] infoModInd, int indType) throws OntologyLoadException {
[27]
[28]    MorphismDomain myMF = null;
[29]    OWLIndividual prod_name = null;
[30]    RefOntoHandler refOntologyHandler = new RefOntoHandler();
[31]
[32]      for (Iterator i = ListmodElement.iterator();  i.hasNext(); ) {
[33]          String me = i.next().toString();
[34]
[35]          //check if is a ref element, and if so create with its name
[36]          try{
[37]          prod_name = refOntologyHandler.getRefInd(me);
[38]          }catch(Exception e){
[39]          }
[40]
[41]          if (prod_name != null){
[42]          mf.createModelElement(me);
[43]          }else{
[44]          mf.createModelElement(me);
[45]          }
[46]          myMF = mf.createMorphismDomain(me+"_root");
[47]          if (infoModInd.getName().contains(refInfoModel)){
[48]             String domainRefPath = refOntologyHandler.getPath(me);
[49]           myMF.setDomainPath(domainRefPath);
```

```
[50]          }else{
[51]          myMF.setDomainPath(domainPath+infoModInd.getPrefixedName());
[52]         }
[53]    }
[54]      saveFile(mf);
[55]      owlMediatorModel  = readOntology("file:///C:/", fileName);
[56]      OWLIndividual IM = owlMediatorModel.getOWLIndividual(infoModInd.getName());
[57]      OWLIndividual ind = null;
[58]
[59]      for (Iterator j = ListmodElement.iterator();  j.hasNext(); ) {
[60]          String name = j.next().toString();
[61]          ind = owlMediatorModel.getOWLIndividual(name);
[62]          prod_name = null;
[63]         try{
[64]          prod_name = refOntologyHandler.getRefInd(name);
[65]         }catch(Exception e){
[66]         }
[67]          ind.setPropertyValue(nameProperty, ind.getPrefixedName());
[68]          if (indType == 1){
[69]          ind.setPropertyValue(elementTypeProperty, elementTypeInstance);
[70]         }else if (indType == 2){
[71]          ind.setPropertyValue(elementTypeProperty, elementTypeMediator);
[72]         }else if (indType == 0){
[73]          ind.setPropertyValue(elementTypeProperty, elementTypeClass);
[74]         }else if (indType == 3){
[75]          ind.setPropertyValue(elementTypeProperty, elementTypeProp);
[76]         }
[77]
[78]          ind.setPropertyValue(hasPrimaryTypeProperty, false);
[79]          ind.setPropertyValue(naturalLanguageProperty, naturalLanguage);
[80]          ind.setPropertyValue(versionProperty, version);
[81]          ind.setPropertyValue(isRootProperty, true);
[82]          ind.setPropertyValue(belongsToProperty, IM);
[83]          OWLIndividual morph = owlMediatorModel.getOWLIndividual(myMF.getName());
[84]          ind.setPropertyValue(hasMorphismDomainProperty, morph);
[85]
[86]    }
[87]       owlMediatorModel.save(new     File("C:/"+fileName).toURI(),     FileUtils.langXMLAbbrev,
       errors);
[88]
[89]          mf = new MyFactory(owlMediatorModel);
[90]          return mf;
[91]}
```

createMorphismOfConceptualType (ElementA; ElementB): void

```
[92] public  void  createMorphismOfConceptualType  (MyFactory  mf,  String  ElementA,  String
     ElementB){
[93]
[94]    ModelElement IndA = mf.getModelElement(ElementA);
[95]    Collection colA = IndA.getHasMorphismDomain();
[96]    String CMname = ElementA +"_"+ ElementB;
[97]    ConceptualMapping myCM = mf.getConceptualMapping(CMname);
[98]    if (myCM == null){
[99]    myCM = mf.createConceptualMapping(CMname);
[100]           }
[101]           if (colA.isEmpty() == false){
[102]           MorphismDomain myMFA = (MorphismDomain) colA.iterator().next();
[103]           myCM.setRelated(myMFA);
```

```
[104]            }else{
[105]            MorphismDomain myMFA = mf.createMorphismDomain(ElementA+"_root");
[106]            myCM.setRelated(myMFA);
[107]            }
[108]
[109]            ModelElement IndB = mf.getModelElement(ElementB);
[110]            Collection colB = IndB.getHasMorphismDomain();
[111]            if (colB.isEmpty() == false){
[112]            MorphismDomain myMFB = (MorphismDomain) colB.iterator().next();
[113]            myCM.setRelating(myMFB);
[114]            }else{
[115]            MorphismDomain myMFB = mf.createMorphismDomain(ElementB+"_root");
[116]            myCM.setRelating(myMFB);
[117]            }
[118]
[119]            myCM.setKMType("Conceptual");
[120]            myCM.setAuthor(owner);
[121]            myCM.setMorphismType(morphismType);
[122]            myCM.setWeight(weightInic);
[123]
[124]            saveFile(mf);
[125]        }
```

increaseConceptualMappingWeight (ElementA; ElementB): void

```
[1] public void increaseConceptualMappingWeight (MyFactory mf, String ElementA, String
    ElementB){
[2]  //comparar o information model
[3]
[4]          ConceptualMapping name = getConceptualMappingBetween2Elem(mf, ElementA,
    ElementB);
[5]
[6]          float weight;
[7]          float f = (float) 1.0;
[8]          if (name != null){
[9]             try{
[10]               weight = name.getWeight();
[11]               float weight_value = weight + f;
[12]               mf.getConceptualMapping(name.getName()).setWeight(weight_value);
[13]            } catch (Exception e) {
[14]              mf.getConceptualMapping(name.getName()).setWeight(f);
[15]            }
[16]   }
[17]         saveFile(mf);
[18]   }
```

isModelElementCreated(ElementA): boolean

```
[1]  public boolean isModelElementCreated(MyFactory mf, String ElementA){
[2]
[3]        boolean var_boolean = false;
[4]        try{
[5]        ModelElement results = mf.getModelElement(ElementA);
[6]           if (results != null){
[7]           var_boolean = true;
[8]           }
[9]        }catch (Exception e) {
[10]       var_boolean = false;
```

```
[11]        }
[12]
[13]     return var_boolean;
[14]    }
```

prepareAllClassesAndIndividuals(OntologyRefURL):void

```
[1]  public void prepareAllClassesAndIndividuals(String uri) throws OntologyLoadException{
[2]
[3]      init();
[4]      startAllClasses(owlModel);
[5]      startAllInstances(owlModel);
[6]      saveFile();
[7]
[8]  }
```

increaseWeightOfClass(ClassName):void

```
[1]  public void increaseWeightOfClass(OWLNamedClass cls){
[2]      OWLNamedClass thisClass = owlModel.getOWLNamedClass(cls.getPrefixedName());
[3]      Collection commentCol = thisClass.getComments();
[4]          for (Iterator ic = commentCol.iterator(); ic.hasNext();){
[5]              String Weight = ic.next().toString();
[6]              String comment_str = Weight;
[7]              if (Weight.contains("<weight>") == true){
[8]                Weight = Weight.replace("<weight>", "");
[9]                Weight = Weight.replace("</weight>", "");
[10]               int weight_value = Integer.parseInt(Weight);
[11]               thisClass.removeComment(comment_str);
[12]               weight_value = weight_value + 1;
[13]               String final_weight = "<weight>"+weight_value+"</weight>";
[14]               thisClass.addComment(final_weight);
[15]             }
[16]         }
[17]     saveFile();
[18]}
```

increaseWeightOfIndividual(ClassName):void

```
[1]  public void increaseWeightOfIndividual(OWLIndividual ind){
[2]      OWLIndividual thisInd = owlModel.getOWLIndividual(ind.getPrefixedName());
[3]      Collection commentCol = thisInd.getComments();
[4]          for (Iterator ic = commentCol.iterator(); ic.hasNext();){
[5]              String Weight = ic.next().toString();
[6]              String comment_str = Weight;
[7]              if (Weight.contains("<weight>") == true){
[8]                Weight = Weight.replace("<weight>", "");
[9]                Weight = Weight.replace("</weight>", "");
[10]               int weight_value = Integer.parseInt(Weight);
[11]               thisInd.removeComment(comment_str);
[12]               weight_value = weight_value + 1;
[13]               String final_weight = "<weight>"+weight_value+"</weight>";
[14]               thisInd.addComment(final_weight);
[15]             }
[16]         }
[17]         saveFile();
[18]}
```

getPropertyDomains(NomedaPropriedade):ListofClasses

```
[1]  public List getPropertyDomains(String prop){
[2]
[3]     boolean is_objprop = true;
[4]     boolean is_datatype = true;
[5]     List lst =  new ArrayList();
[6]     Collection col = null;
[7]     Collection rangeCol = null;
[8]
[9]     try{
[10]    OWLDatatypeProperty DTprop = owlModel.getOWLDatatypeProperty(prop);
[11]    col = DTprop.getUnionDomain();
[12]    }
[13]    catch (Exception e){
[14]       is_datatype = false;}
[15]    try{
[16]    OWLObjectProperty ObjPro = owlModel.getOWLObjectProperty(prop);
[17]    col = ObjPro.getUnionDomain();
[18]    rangeCol = ObjPro.getRanges(false);
[19]    }
[20]    catch (Exception e){
[21]      is_objprop = false; }
[22]
[23] if (is_datatype == true){
[24]    for (Iterator i = col.iterator(); i.hasNext();){
[25]       OWLNamedClass cls = (OWLNamedClass) i.next();
[26]       lst.add(cls.getName()); //getPrefixedName());
[27]    }
[28] }
[29]
[30] if (is_objprop == true){
[31]    for (Iterator i = col.iterator(); i.hasNext();){
[32]       OWLNamedClass cls = (OWLNamedClass) i.next();
[33]       lst.add(cls.getName());//.getPrefixedName());
[34]    }
[35]    for (Iterator j = rangeCol.iterator(); j.hasNext();){
[36]       OWLNamedClass cls = (OWLNamedClass) j.next();
[37]       lst.add(cls.getName());//.getPrefixedName());
[38]    }
[39] }
[40]    return lst;
[41] }
```

hasClassElement(name):boolean

```
[1]  public Boolean hasClassElement(String cls_name ){
[2]
[3]     boolean is_class = false;
[4]     OWLNamedClass cls = null;
[5]     try{
[6]     cls = owlModel.getOWLNamedClass(cls_name);
[7]      }catch (Exception e){}
[8]
[9]     if (cls != null){
[10]       is_class = true;
[11]    }
[12]
[13]   return is_class;
```

```
[14]}
```

hasInstanceElement(name):boolean

```
[1]  public Boolean hasInstanceElement(String inst_name ){
[2]      boolean is_ind = false;
[3]      OWLIndividual ind = null;
[4]      String str = null;
[5]
[6]      try{
[7]          ind = owlModel.getOWLIndividual(inst_name);
[8]      }catch (Exception e){}
[9]
[10]     if (ind == null){
[11]         try{
[12]         //let see if is there any individual with this name
[13]         str = getIndividualName(inst_name);
[14]         ind = owlModel.getOWLIndividual(str);
[15]     }catch (Exception e){}
[16]     }
[17]
[18]     if (ind != null){
[19]         is_ind = true;
[20]     }
[21]
[22]    return is_ind;
[23]}
[24]
```

hasPropertyElement(name):boolean

```
[1]  public Boolean hasPropertyElement(String prop_name ){
[2]      boolean is_objprop = false;
[3]      boolean is_datatype = false;
[4]      boolean is_prop = false;
[5]
[6]      try{
[7]      OWLDatatypeProperty DTprop = owlModel.getOWLDatatypeProperty(prop_name);
[8]      if (DTprop != null){
[9]          is_datatype = true;
[10]     }
[11]     }
[12]     catch (Exception e){
[13]        is_datatype = false;}
[14]     try{
[15]     OWLObjectProperty ObjPro = owlModel.getOWLObjectProperty(prop_name);
[16]     if (ObjPro != null) {
[17]         is_objprop = true;
[18]     }
[19]     }
[20]     catch (Exception e){
[21]        is_objprop = false; }
[22]
[23]     if (is_objprop == true || is_datatype == true){
[24]         is_prop = true;
[25]     }
[26]     return is_prop;
[27]}
```

cleanChildClassesFromRootClassesList(List of RootClasses):List of RootCLasses

```
[1]  public List cleanChildClassesFromRootClassesList(List rootClasses ){
[2]      List lst = new ArrayList(rootClasses);
[3]      List removedLst = new ArrayList();
[4]      List bothLists = new ArrayList();
[5]      Collections.copy(lst, rootClasses);
[6]      List lstCompleted = lstWithAllSubclasses;
[7]
[8]      for (Iterator i = rootClasses.iterator(); i.hasNext();){
[9]          OWLNamedClass cls = (OWLNamedClass) i.next();
[10]         if (cls != null){
[11]         Collection col = getSubClass(cls);
[12]         getCollectionClass(col);
[13]
[14]         for (Iterator j = lstCompleted.iterator(); j.hasNext();){
[15]             OWLNamedClass subcls = (OWLNamedClass) j.next();
[16]             for (Iterator k = rootClasses.iterator(); k.hasNext();){
[17]                 OWLNamedClass clsMatch = (OWLNamedClass) k.next();
[18]                 if (clsMatch.toString().contentEquals(subcls.toString())){
[19]                     lst.remove(subcls);
[20]                     removedLst.add(subcls);
[21]                 }
[22]             }
[23]         }
[24]         lstCompleted.clear();
[25]     }
[26]     }
[27]     bothLists.add(lst);
[28]     bothLists.add(removedLst);
[29]     return bothLists;
[30] }
```

getProductsRelatedWith(input:String; maxResults:int): List of (ProductName; ProductDescription; ProductURL)

```
[1]  public List getProductsRelatedWith(String str, int maxResults) throws OntologyLoadException{
[2]          List productsList = new ArrayList();
[3]          KnowledgeHandler kh = new KnowledgeHandler();
[4]
[5]          kh.ModelElementList = new ArrayList();
[6]          kh.owlIndividualList = new ArrayList();
[7]          kh.owlNamedClassList = new ArrayList();
[8]
[9]          if (str.isEmpty() == false){
[10]         //Here all data is prepared
[11]         kh.introductionOfSearchKnowledge(str);
[12]
[13]         //After data is prepared, the graph is buid and it return maxResults products
[14]         productsList = kh.getSearchResults(maxResults);
[15]
[16]         return productsList;
[17]         }else{
[18]           productsList.add("No search keyword");
[19]           return productsList;
[20]         }
[21]     }
```

showUsagePatterns(void):List of (List of Possible New Products)(List of Possible New Ref Concepts)

```
[1]  public List showUsagePatterns() throws OntologyLoadException{
```

```
[2]         List possibleNewProd = new ArrayList();
[3]         List possibleNewRefConcepts = new ArrayList();
[4]         List finalList = new ArrayList();
[5]
[6]         RefOntoHandler refOntoHandler = new RefOntoHandler();
[7]         RefOntoHandler.init();
[8]         OntologyHandler ontologyHandler = new OntologyHandler();
[9]         MyFactory myF = ontologyHandler.initOWL();
[10]        Collection indCol = myF.getAllModelElementInstances();
[11]
[12]        //new prod
[13]        for (Iterator i = indCol.iterator(); i.hasNext();){
[14]            try{
[15]            ModelElement me1 = (ModelElement) i.next();
[16]                  boolean me1IsClass = refOntoHandler.hasClassElement(me1.getName());
[17]                  boolean me1IsInstance = refOntoHandler.hasInstanceElement(me1.getName());
[18]                  boolean me1IsProp = refOntoHandler.hasPropertyElement(me1.getName());
[19]           if ((me1IsClass == false) && (me1IsInstance == false) && (me1IsProp == false)){
[20]            Collection hasMorphCol = me1.getHasMorphismDomain();
[21]            for (Iterator j = hasMorphCol.iterator(); j.hasNext();){
[22]                MorphismDomain md1 = (MorphismDomain) j.next();
[23]                Collection morphCol = md1.getAppliedMorphism();
[24]                for (Iterator k = morphCol.iterator(); k.hasNext();){
[25]                    ConceptualMapping cm1 = (ConceptualMapping) k.next();
[26]                    float conceptualWeight = cm1.getWeight();
[27]                    if (conceptualWeight > Integer.parseInt(Form.oLearch.newObj.getText())){
[28]                    mediator.Object me2 = cm1.getRelating().getAssociatedObject();
[29]
[30]                    boolean isClass = refOntoHandler.hasClassElement(me2.getName());
[31]                    boolean isInstance = refOntoHandler.hasInstanceElement(me2.getName());
[32]                    boolean isProp = refOntoHandler.hasPropertyElement(me2.getName());
[33]
[34]                    if ((isClass == false) && (isInstance == false) && (isProp == false)){
[35]                        if (me1.getName().contentEquals(me2.getName())== true){
[36]                        possibleNewProd.add(me1.getName()+ " has weight "+conceptualWeight+" so maybe it's a new p
[37]                        }else{
[38]                            //if already exists, do not add again. If exists but with high weight, then replace
[39]                            List auxPatterns = new ArrayList();
[40]                            auxPatterns = possibleNewProd;
[41]                            boolean addedNewProd = false;
[42]                            for(Iterator x = auxPatterns.iterator(); x.hasNext();){
[43]                                String member = (String) x.next();
[44]                                if (member.contains(me2.getName() + " and "+ me1.getName()) == true){
[45]                                    addedNewProd = true;
[46]                                    String aux = member;
[47]                                    aux = aux.replace(me2.getName() + " and "+ me1.getName()+ " searched ", "");
[48]                                    aux = aux.replace(" times.", "");
[49]                                    float weight = new Float (aux);
[50]                                    if (weight < conceptualWeight){
[51]                                    possibleNewProd.remove(member);
[52]                                    possibleNewProd.add(me1.getName() + " and "+ me2.getName()+ " has weight "+concep
    so maybe it's a new product.");
[53]                                    }
[54]                                }
[55]                            }
[56]                            if (addedNewProd == false){
[57]                                possibleNewProd.add(me1.getName() + " and "+ me2.getName()+ " has weight "+conceptua
    maybe it's a new product.");
[58]                          }
[59]                        }
```

```
[60]                }
[61]                }
[62]             }
[63]          }
[64]          }
[65]          }catch (Exception e){
[66]          System.out.println("line 152 OntoLearn");}
[67]       }
[68]
[69]       //ref concepts
[70]       for (Iterator i = indCol.iterator(); i.hasNext();){
[71]          try{
[72]          ModelElement me1 = (ModelElement) i.next();
[73]          Collection hasMorphCol = me1.getHasMorphismDomain();
[74]          for (Iterator j = hasMorphCol.iterator(); j.hasNext();){
[75]             MorphismDomain md1 = (MorphismDomain) j.next();
[76]             Collection morphCol = md1.getAppliedMorphism();
[77]
[78]             //get concept weight of own relation
[79]             float compareWeight = 0;
[80]             for (Iterator z = morphCol.iterator(); z.hasNext();){
[81]                ConceptualMapping cm = (ConceptualMapping) z.next();
[82]                mediator.Object obj2 = cm.getRelating().getAssociatedObject();
[83]                if ((me1.getName().contentEquals(obj2.getName()) == true)){
[84]                compareWeight = cm.getWeight();
[85]
[86]
[87]             //get relating weight
[88]             for (Iterator k = morphCol.iterator(); k.hasNext();){
[89]                ConceptualMapping cm1 = (ConceptualMapping) k.next();
[90]                if (compareWeight > Integer.parseInt(Form.oLearch.newConcept.getText())){
[91]                mediator.Object me2 = cm1.getRelating().getAssociatedObject();
[92]
[93]                boolean isClass = refOntoHandler.hasClassElement(me2.getName());
[94]                boolean isInstance = refOntoHandler.hasInstanceElement(me2.getName());
[95]                boolean isProp = refOntoHandler.hasPropertyElement(me2.getName());
[96]                int weight_value = 0;
[97]                if (isClass == true){
[98]                   OWLNamedClass refCls = refOntoHandler.getRefClass(me2.getName());
[99]                   weight_value = refOntoHandler.getClassesWeight(refCls);
[100]                        if ((compareWeight > weight_value) && (me1.getName().contentEquals(m
     false)){
[101]                        possibleNewRefConcepts.add(me1.getName() + " is relating "+ me2.getN
     searched "+compareWeight+" times.");
[102]                        }
[103]
[104]                  }
[105]                if (isInstance == true){
[106]                   OWLIndividual refInd = refOntoHandler.getRefInd(me2.getName());
[107]                   weight_value = refOntoHandler.getInstancesWeight(refInd);
[108]                        if ((compareWeight > weight_value) && (me1.getName().contentEquals(m
     false)){
[109]                        possibleNewRefConcepts.add(me1.getName() + " is relating "+ me2.getN
     searched "+compareWeight+" times.");
[110]                        }
[111]                     }
[112]                if(isProp == true){
[113]                   }
[114]
[115]                   }
```

```
[116]                           }
[117]                           }
[118]                       }
[119]                       }
[120]                   }catch (Exception e){
[121]                   System.out.println("line 235 OntoLearn");}
[122]                   }
[123]
[124]               finalList.add(possibleNewProd);
[125]               finalList.add(possibleNewRefConcepts);
[126]
[127]               return finalList;
[128]
[129]           }
[130]       }
```

introductionOfSearchKnowledge(input:String):void

```
[1]  public void introductionOfSearchKnowledge(String str) throws OntologyLoadException{
[2]
[3]          List wordslst = new ArrayList();
[4]          List listOfConcepts = new ArrayList();
[5]          Util util = new Util();
[6]          boolean isClass = false;
[7]          boolean isPropertyElement = false;
[8]          boolean isInstanceElement = false;
[9]
[10]         RefOntoHandler refOntoHandler = new RefOntoHandler();
[11]         RefOntoHandler.init();
[12]         OntologyHandler ontologyHandler = new OntologyHandler();
[13]         mediator.MyFactory myF = ontologyHandler.initOWL();
[14]
[15]         //from String to list
[16]         wordslst = util.getIntroducedWords(str);
[17]         //search for ref elements related with mediator elements.
[18]         //Data will be inserted in a table with related obj, object and weight.
[19]         for (Iterator medElem = wordslst.iterator(); medElem.hasNext();){
[20]             String compareString = (String) medElem.next();
[21]             searchOverallMediatorForRefClassesMappedWith(compareString);
[22]         }
[23]         listOfConcepts = util.inicConceptsToSearch(wordslst);
[24]
[25]         //sequence diagram 1
[26]             for (Iterator i = wordslst.iterator(); i.hasNext();){
[27]                 String concept = (String) i.next();
[28]                 isClass = refOntoHandler.hasClassElement(concept);
[29]                 if (isClass == true){
[30]                     OWLNamedClass cls = refOntoHandler.getRefClass(concept);
[31]                     refOntoHandler.increaseWeightOfClass(cls);
[32]                     addRootClasses(cls);
[33]                 }else if (isClass == false){
[34]                     isPropertyElement = refOntoHandler.hasPropertyElement(concept);
[35]                     if (isPropertyElement == true){
[36]                         List propDomainList = refOntoHandler.getPropertyDomains(concept);
[37]
[38]                         for (Iterator j = propDomainList.iterator(); j.hasNext();){
[39]                             String propName = (String) j.next();
[40]                             try{
[41]                             OWLNamedClass clsName = refOntoHandler.getRefClass(propName);
```

```
[42]                        refOntoHandler.increaseWeightOfClass(clsName);
[43]                        addRootClasses(clsName);
[44]                        }catch (Exception e){
[45]                        }
[46]                    }
[47]                }
[48]            }
[49]            isInstanceElement = refOntoHandler.hasInstanceElement(concept);
[50]            if (isInstanceElement == true){
[51]                OWLIndividual indName = refOntoHandler.getRefInd(concept);
[52]                refOntoHandler.increaseWeightOfIndividual(indName);
[53]                addRootInstances(indName);
[54]            }
[55]            boolean isModelElement = ontologyHandler.isModelElementCreated(myF, concept);
[56]            if (isModelElement == true){
[57]                ModelElement me = myF.getModelElement(concept);
[58]                addRootClassesFromMediator(me);
[59]            }else if ((isModelElement == false) && ((wordslst.size() > 1) || ((isModelElement == false) &&(isI
    false ) && (isPropertyElement == false) && (isClass == false))) ){
[60]                List auxList = new ArrayList();
[61]                auxList.add(concept);
[62]                InformationModel myIM;
[63]                if ((isInstanceElement == false) && (isClass == false) && (isPropertyElement == false)){
[64]                myIM = myF.getInformationModel(infoModel);
[65]                }else{
[66]                myIM = myF.getInformationModel(refOntology);
[67]                }
[68]                if (isInstanceElement == true){
[69]                myF = ontologyHandler.createModelElements(myF, auxList, myIM, 1);
[70]                }
[71]                else if(isClass == true)
[72]                {
[73]                myF = ontologyHandler.createModelElements(myF, auxList, myIM, 0);
[74]                }
[75]                else if(isPropertyElement == true)
[76]                {
[77]                myF = ontologyHandler.createModelElements(myF, auxList, myIM, 3);
[78]                }
[79]                else
[80]                {
[81]                myF = ontologyHandler.createModelElements(myF, auxList, myIM, 2);
[82]                }
[83]            }
[84]        }
[85]        //sequence diagram 2
[86]        boolean conceptAconceptBincreased = false;
[87]        boolean conceptBconceptAincreased = false;
[88]        String concept1;
[89]        String concept2;
[90]        boolean oneElement = false;
[91]        for (Iterator i = listOfConcepts.iterator(); i.hasNext();){
[92]            List firstConcepts = (List) i.next();
[93]            if (firstConcepts.size() == 1){
[94]            concept1 = firstConcepts.get(0).toString();
[95]            firstConcepts.add(concept1);
[96]            oneElement = true;
[97]            }
[98]
[99]            for (Iterator j = firstConcepts.iterator(); j.hasNext();){
[100]                       concept1 = (String) j.next();
```

```
[101]                         concept2 = (String) j.next();
[102]                         // concept1 relating concept2
[103]                         List     listOfElementsMappedWith     =     ontologyHandler.getElementsConceptualMapp
    concept1);
[104]                         for (Iterator k = listOfElementsMappedWith.iterator(); k.hasNext();){
[105]                            String relatingObj = k.next().toString();
[106]                         //if related obj is concept2...
[107]                         if (relatingObj.contentEquals(concept2)){
[108]                           ontologyHandler.increaseConceptualMappingWeight(myF, concept1, concept2);
[109]                           conceptAconceptBincreased = true;
[110]                         }
[111]                         }
[112]
[113]                         // concept2 relating concept 1
[114]                         List                                    listOfElementsMappedWith2
    ontologyHandler.getElementsConceptualMappedWith(myF,concept2);
[115]                         for (Iterator k = listOfElementsMappedWith2.iterator(); k.hasNext();){
[116]                            String relatingObj = k.next().toString();
[117]                         //if related obj is concept1 and cocnept1 is not equal concept2...
[118]                         if ((relatingObj.contentEquals(concept1) == true) && (concept1.contentEquals(concept2)=
[119]                           ontologyHandler.increaseConceptualMappingWeight(myF, concept2, concept1);
[120]                           conceptBconceptAincreased = true;
[121]                         }
[122]                         }
[123]
[124]                         //if conceptAconceptBincreased = false, so these concepts may not have relation
[125]                         if (conceptAconceptBincreased == false){
[126]                            //confirmar concept1 e concept2 sao ModelElements
[127]                            ModelElement concept1ME = myF.getModelElement(concept1);
[128]                            ModelElement concept2ME = myF.getModelElement(concept2);
[129]                            if (((concept1ME != null) || (concept2ME != null))){
[130]                               if ((oneElement == true) && (wordslst.size() <= 1) && ((isInstanceElement == true) |
    true))){
[131]                                  //vem um elemento da ref sozinho, nao deve criar relaçao entre ele proprio.
[132]                               }else{
[133]                            ontologyHandler.createMorphismOfConceptualType(myF, concept1, concept2);
[134]                            }}else if ((concept1ME == null) && (oneElement == false)){
[135]                               List aList = new ArrayList();
[136]                               aList.add(concept1);
[137]                               InformationModel myIM = myF.getInformationModel(refOntology);
[138]                               boolean isModelElement = ontologyHandler.isModelElementCreated(myF, concept1)
[139]                               isClass = refOntoHandler.hasClassElement(concept1);
[140]                               isPropertyElement = refOntoHandler.hasPropertyElement(concept1);
[141]                                  if (isModelElement == true){
[142]                                  myF = ontologyHandler.createModelElements(myF, aList, myIM, 1);
[143]                                  }
[144]                                  else if(isClass == true)
[145]                                  {
[146]                                  myF = ontologyHandler.createModelElements(myF, aList, myIM, 0);
[147]                                  }
[148]                                  else if(isPropertyElement == true)
[149]                                  {
[150]                                  myF = ontologyHandler.createModelElements(myF, aList, myIM, 3);
[151]                                  }
[152]                                  else
[153]                                  {
[154]                                  myF = ontologyHandler.createModelElements(myF, aList, myIM, 2);
[155]                                  }
[156]                             //else if concept2 is not mediator element and has search word
[157]                            }else if ((concept2ME == null) && (oneElement == false)){
```

```
[158]                                   List aList = new ArrayList();
[159]                                   aList.add(concept2);
[160]                                   InformationModel myIM = myF.getInformationModel(refOntology);
[161]                                   boolean isModelElement = ontologyHandler.isModelElementCreated(myF, con
[162]                                   isClass = refOntoHandler.hasClassElement(concept2);
[163]                               if (isModelElement == true){
[164]                               myF = ontologyHandler.createModelElements(myF, aList, myIM, 1);
[165]                               }
[166]                               else if(isClass == true)
[167]                               {
[168]                               myF = ontologyHandler.createModelElements(myF, aList, myIM, 0);
[169]                               }
[170]                               else
[171]                               {
[172]                               myF = ontologyHandler.createModelElements(myF, aList, myIM, 2);
[173]                               }
[174]                           }
[175]
[176]                           if (conceptBconceptAincreased == false){
[177]                               // check that we're handling with 2 different ModelElements
[178]                               if (concept1.contentEquals(concept2) == false){
[179]                               ontologyHandler.createMorphismOfConceptualType(myF, concept2, concept1);
[180]                               }
[181]                           }
[182]                           }
[183]                       }
[184]                   }
[185]                   ontologyHandler.saveFile(myF);
[186]           }
```

getSearchResults(maxResults): listOfProducts

```
[1]  public List getSearchResults(int maxResults) throws OntologyLoadException{
[2]
[3]          List auxProductsList = new ArrayList();
[4]          finalProductsList = searchForProducts(owlNamedClassList, owlIndividualList, medRef
     medRefIndPlusWeightList, maxResults);
[5]
[6]          if (maxResults > finalProductsList.size()){
[7]              maxResults = finalProductsList.size();
[8]          }
[9]
[10]         for (int i = 0; i < maxResults;){
[11]             Products prod  = (Products) finalProductsList.get(i);
[12]             auxProductsList.add(prod);
[13]             i = i + 1;
[14]         }
[15]         return auxProductsList;
[16]   }
```

searchOverallMediatorForRefClassesMappedWith(Concept):OwnGraphUntilRefClass

```
[1]  public void searchOverallMediatorForRefClassesMappedWith(String concept) throws OntologyLoadExcept
[2]
[3]          RefOntoHandler ontoHandlerRef = new RefOntoHandler();
[4]          OntologyHandler ontologyHandler = new OntologyHandler();
[5]          RefClassInMediator mediatorRefClass = new RefClassInMediator();
[6]          RefIndividualInMediator mediatorRefInd = new RefIndividualInMediator();
```

```
[7]         MediatorRefClassPlusWeight mediatorRefClsPlusWeight = new MediatorRefClassPlusWeight();
[8]         MediatorRefIndPlusWeight mediatorRefIndPlusWeight = new MediatorRefIndPlusWeight();
[9]         MyFactory mf;
[10]        String compareConcept;
[11]
[12]        mf = ontologyHandler.initOWL();
[13]        List mappedWithList = new ArrayList();
[14]        List clsList = new ArrayList();
[15]        List indList = new ArrayList();
[16]        mappedWithList = ontologyHandler.getElementsConceptualMappedWith(mf, concept);
[17]        for (Iterator i = mappedWithList.iterator(); i.hasNext();){
[18]           compareConcept = i.next().toString();
[19]
[20]           boolean isClassElem = ontoHandlerRef.hasClassElement(compareConcept);
[21]           if (isClassElem == true){
[22]              OWLNamedClass cls = ontoHandlerRef.getRefClass(compareConcept);
[23]              mediatorRefClass.setMediatorRefClass(cls.getName());
[24]              //get relation weight
[25]              ConceptualMapping cm = ontologyHandler.getConceptualMappingBetween2Elem(mf,
    compareConcept);
[26]              String auxWeight = Float.toString(cm.getWeight());
[27]              auxWeight = auxWeight.replace(".0", "");
[28]              int clsWeight = Integer.parseInt(auxWeight);
[29]              //ontoHandlerRef.getClassesWeight(cls);
[30]              mediatorRefClass.setMediatorRefClassWeight(clsWeight);
[31]              clsList.add(mediatorRefClass);
[32]           }
[33]           boolean isInstElem = ontoHandlerRef.hasInstanceElement(compareConcept);
[34]           if (isInstElem == true){
[35]              OWLIndividual ind = ontoHandlerRef.getRefInd(compareConcept);
[36]              mediatorRefInd.setMediatorRefInd(ind.getName());
[37]              //get relation weight
[38]              ConceptualMapping cm = ontologyHandler.getConceptualMappingBetween2Elem(mf,
    compareConcept);
[39]              String auxWeight = Float.toString(cm.getWeight());
[40]              auxWeight = auxWeight.replace(".0", "");
[41]              int indWeight = Integer.parseInt(auxWeight);
[42]              //int indWeight = ontoHandlerRef.getInstancesWeight(ind);
[43]              mediatorRefInd.setMediatorRefIndWeight(indWeight);
[44]              indList.add(mediatorRefInd);
[45]           }
[46]
[47]           //add objects to list
[48]           if ((clsList.size() > 0) && (isClassElem == true)){
[49]              ConceptualMapping thisConceptMap = ontologyHandler.getConceptualMappingBetween2Elem(m
    concept);//.getMediatorWeight(thisModelElement);
[50]              float clsWeight = (float) 1.0;
[51]              if (thisConceptMap != null){
[52]              clsWeight = thisConceptMap.getWeight();
[53]              }
[54]              String auxWeight = Float.toString(clsWeight);
[55]              auxWeight = auxWeight.replace(".0", "");
[56]              int weight = Integer.parseInt(auxWeight);
[57]           mediatorRefClsPlusWeight.setMediatorRefClassWeight(weight);
[58]           mediatorRefClsPlusWeight.setRefClassInMediator(clsList);
[59]           mediatorRefClsPlusWeight.setMediatorRefCls(concept);
[60]           }
[61]
[62]           if ((indList.size() > 0) && (isInstElem == true)){
[63]              ConceptualMapping thisConceptMap = ontologyHandler.getConceptualMappingBetween2Elem(m
```

```
    concept);
[64]                float indWeight = (float) 1.0;
[65]                if (thisConceptMap != null){
[66]                indWeight = thisConceptMap.getWeight();
[67]                }
[68]                String auxWeight = Float.toString(indWeight);
[69]                auxWeight = auxWeight.replace(".0", "");
[70]                int weight = Integer.parseInt(auxWeight);
[71]            mediatorRefIndPlusWeight.setMediatorRefIndWeight(weight);
[72]            mediatorRefIndPlusWeight.setRefIndInMediator(indList);
[73]            mediatorRefIndPlusWeight.setMediatorRefInd(concept);
[74]            }
[75]
[76]         }
[77]
[78]         medRefClsPlusWeightList.add(mediatorRefClsPlusWeight);
[79]         medRefIndPlusWeightList.add(mediatorRefIndPlusWeight);
[80]   }
```

searchForProducts(List of (Classes, Individuals, ReferenceCls, MediatorCls, maxResults): List of Products

```
[81] public List searchForProducts(List namedClassList, List individualList, List modelRefCls, List
    modelRefInd, int maxResults) throws OntologyLoadException{
[82]         final java.util.ArrayList path = new java.util.ArrayList();
[83]         List productsList = new ArrayList();
[84]         List labelsList = new ArrayList();
[85]         List graphList = new ArrayList();
[86]         List ownGraphList = new ArrayList();
[87]         //list with only root classes
[88]         List rootClasses = new ArrayList();
[89]
[90]         int source = 0;
[91]         Util util = new Util();
[92]         RefOntoHandler refOntoHandler = new RefOntoHandler();
[93]         RefOntoHandler.init();
[94]         rootClasses                                                                    =
    refOntoHandler.cleanChildClassesFromRootClassesList(namedClassList);
[95]         List rootCls = (List) rootClasses.get(0);
[96]         List removedCls = (List) rootClasses.get(1);
[97]
[98]         ownGraphList    =    util.createOwnGraph(rootCls,    individualList,    removedCls,
    modelRefCls, modelRefInd);
[99]
[100]                //get graph list
[101]                List auxGraph = new ArrayList();
[102]                auxGraph.add(ownGraphList.get(1));
[103]                for (Iterator j = auxGraph.iterator(); j.hasNext();){
[104]                   List lst_1 = (List) j.next();
[105]                for (Iterator i = lst_1.iterator(); i.hasNext();){
[106]                   graphList.add(i.next());
[107]                   }
[108]                }
[109]
[110]                //Transform weight in distance
[111]                List transformedWeight = util.transformWeightInDistance(graphList);
[112]                ownGraphList.remove(1);
[113]                ownGraphList.add(1, transformedWeight);
[114]
```

```
[115]                   //get label list
[116]                   List auxLabel = new ArrayList();
[117]                   auxLabel.add(ownGraphList.get(0));
[118]                   for (Iterator j = auxLabel.iterator(); j.hasNext();){
[119]                       List lst_1 = (List) j.next();
[120]                   for (Iterator i = lst_1.iterator(); i.hasNext();){
[121]                       labelsList.add(i.next());
[122]                       }
[123]                   }
[124]
[125]                   while (maxResults > 0){
[126]                       int intSize = labelsList.size();
[127]                       WeightedGraph wg = new WeightedGraph(intSize);
[128]                       wg = util.createDijkstraGraph(labelsList, graphList);
[129]                       boolean noMoreOut = checkOut(graphList);
[130]                       if ((graphList.isEmpty() == true) || (noMoreOut == true)){
[131]                           break;
[132]                       }
[133]                       final int [] pred = Dijkstra.dijkstra (wg, source);
[134]                       for (int n=0; n<2; n++) {
[135]                        Dijkstra.printPath (wg, pred, 0, n);
[136]                       }
[137]                       productsList = Dijkstra.globalList;
[138]                       int prodListSize = productsList.size();
[139]                       try{
[140]                       OWLIndividual product = null;
[141]                       System.out.println(productsList.get(prodListSize - 1).toString());
[142]                       if (productsList.get(prodListSize - 1).toString().contentEquals("_out")){
[143]                       product    =    refOntoHandler.getRefInd(productsList.get(prodListSize    -
    2).toString());
[144]                           }else{
[145]
[146]                           }
[147]                       //Set product
[148]                       OWLDatatypeProperty                descProperty                =
    refOntoHandler.owlModel.getOWLDatatypeProperty(description);
[149]                       OWLDatatypeProperty                URLProperty                 =
    refOntoHandler.owlModel.getOWLDatatypeProperty(URL);
[150]                       OWLDatatypeProperty                nameProperty                =
    refOntoHandler.owlModel.getOWLDatatypeProperty(prod_name);
[151]                       Products                products                =                new
    util.Products(product.getPropertyValue(nameProperty).toString(),
    product.getPropertyValue(descProperty).toString(),
    product.getPropertyValue(URLProperty).toString(), product.getName());
[152]                       //Add to final list only if it's not yet added
[153]                       String prodName = products.getProductRefURL();
[154]                       boolean addProd = true;
[155]                       for (Iterator cp = path.iterator(); cp.hasNext();){
[156]                           Products addedProdName = (Products) cp.next();
[157]                           if (addedProdName.getProductRefURL().contentEquals(prodName)){
[158]                               addProd = false;
[159]                           }
[160]                       }
[161]                       if (addProd == true){
[162]                       path.add(products);
[163]                       } else{
[164]                           maxResults = maxResults + 1;
[165]                       }
[166]                       }catch (Exception e) {
[167]                       System.out.println("erro: "+ e.getMessage().toString());
```

```
[168]                     }
[169]                     //remove product from graph
[170]                     for(Iterator lab = labelsList.iterator(); lab.hasNext();){
[171]                         Label prodLabel = (Label) lab.next();
[172]                         if    (prodLabel.getName().contentEquals(productsList.get(prodListSize    -
          2).toString())){
[173]                             int prodID = prodLabel.getID();
[174]                                 List auxGraphList = new ArrayList(graphList);
[175]                                 Collections.copy(auxGraphList, graphList);
[176]                             for (Iterator gra = auxGraphList.iterator(); gra.hasNext();){
[177]                                 Graph prodGraph = (Graph) gra.next();
[178]                                 if((prodGraph.getDestiny()   ==   1)   &&   (prodGraph.getOrigin()   ==
          prodID)){
[179]                                     graphList.remove(prodGraph);
[180]                                 }
[181]                             }
[182]                         }
[183]                     }
[184]
[185]
[186]                     maxResults = maxResults - 1;
[187]                 }
[188]                 //finalProductsList = path;
[189]                 return path;
[190]             }
[191]
}
```