International Doctorate School in Information and Communication Technologies

DIT - University of Trento

A LARGE SCALE DISTRIBUTED KNOWLEDGE ORGANIZATION SYSTEM

SHEAK RASHED HAIDER NOORI

April 2011

Advisor:

Prof. Fausto Giunchiglia

Università degli Studi di Trento

Abstract

The revolution of Internet and the Web takes the computer and information technology into a new age. The information on the web is growing very fast. The progress of information and communication technologies has made accessible a large amount of information, which have provided each of us with access to far more information than we can comprehend or manage. This emphasizes the difficulty with the resulting semantic heterogeneity of the diverse sources. Human knowledge is a living organism and as such evolves in time where different people having different viewpoints and using different terminology among people of different cultures and languages, intensify the heterogeneity of the sources even more. These introduce some concrete problems like natural language disambiguation, information retrieval and information integration. Nevertheless, the problem is quite well known in almost every branch of knowledge and has been independently approached by several communities for several decades. To make this huge amount of existing information accessible and manageable while also solving the semantic heterogeneity problem, namely the problem of diversity in knowledge, and therefore support interoperability, it is essential to have a large scale high quality collaborative knowledge base along with a suitable structure as a common ground on which interoperability among people and different systems should be possible. It will play the role of a reference point for communication, assigning clear meaning by accurate disambiguation to exchanged information, communication and automating complex tasks. However, successfully building large scale knowledge bases with maximum coverage is not possible by a single person or a small group of people without collaborative support. It extremely depends on expert community based support. Therefore, it is necessary for experts to work together on knowledge base building. Furthermore, it is very natural that these expert users will be geographically distributed. Web 2.0 has the potential to support information sharing, interoperability and collaboration on the Web. Simplicity, flexibility and easy to use services make it an interactive and collaborative platform which allows them to create or edit their content. The exponential expansion of the Web users and the potentials of Web 2.0 make it the natural platform of choice for developing knowledge bases collaboratively. We propose a highly flexible knowledge base system, which takes into account diversity of knowledge and its evolution in time. The work presented in this thesis is part of a larger project. More specifically the goal of this thesis is to create a powerful and easy to use knowledge base management system to help people in building, organizing a high quality knowledge base and making accessible their knowledge and to support interoperability in real world scenarios.

Keywords
Knowledge, management, maintenance, evolution, visualization, user interface, diversity, p2p search, semantic search, knowledge management.

Contributions

The material presented in the thesis has been developed in collaboration with Fausto Giunchiglia, Ilya Zaihrayeu, Uladzimir Kharkevich, Vincenzo Maltese, Tabin Hasan , Biswanath Dutta.

This dissertation makes the following contributions:

- A survey of state of the art knowledge representation and visualization techniques;
- Provides an overview of different components of a large scale knowledge organization system.
- Design and development of a tool for the linguistic and ontology knowledge management, search and navigation.
- Design and development of a tool for the domain knowledge management, search and navigation.
- Design and development of a tool for the entity type management, search and navigation.
- User based evaluation of the developed system.
- A survey of state of the art p2p information retrieval approaches
- A prototype implementation of p2p semantic indexing and search
- Empirical evaluation of the developed prototype of p2p semantic indexing and search system.

Acknowledgments

This dissertation would not have been possible without the guidance and the help of several individuals who in one way or another contributed their valuable assistance and supported in many ways in the preparation and completion of this study.

First and foremost, sincerest gratitude to my scientific advisor, Prof Fausto Giunchiglia, who has supported me throughout my PhD with his patience, knowledge and encouragement. I will never forget these supports. Prof Fausto Giunchiglia has been my inspiration as I hurdle all the obstacles in the completion this research work. Without his support, inspiration, and encouragement it would have been impossible to finish this thesis.

I am thankful to Ilya Zaihrayeu for the lessons and suggestions he has provided me during these years.

I am thankful to Vincenzo Maltese for his contribution, collaboration, and continuous feedback and meaningful discussions in designing the knowledge base systems.

I also wish to thank Uladzimir Kharkevich for his contribution, collaboration, and meaningful discussions in designing the p2p system.

I am also thankful to Tabin Hasan for all the discussions we had about visualization in designing User Interface.

I am also thankful to Biswanath Dutta for all the discussions we had about conceptual analysis in building domain knowledge.

I gratefully thank Elena for her constructive comments on this thesis.

I am tempted to individually thank all of my friends but as the list might be long and for fear I might omit someone, I will simply and genuinely say: Thank you to you all for your love, care and trust, as well as expressing my apology that I could not mention personally one by one. In particular I would like to cite: Tabin Hasan, Quazi Delwar Hossain, Juel Rana, Feroz Farazi, Ju Qi, Francis Palma for their constant presence and care and instant moral support.

Finally, I thank my parents and family for supporting me throughout all my studies, for love, the tremendous support and inspiration which they gave me in all these years.

I cannot finish without acknowledging how eternally grateful and thankful I am to The One, The Everlasting The Trustee, , The All Determiner, The Dependable and The Protecting Friend that guides me, teaches me to be patient and to never give up. Thank you Dear Allah!

Contents

CHAPTER 1 ..... **1**

1. INTRODUCTION **1**

1.1. The Context 1
1.2. The Problem 4
1.3. The Solution 4
1.4. Structure of the Thesis 6

CHAPTER 2 ..... **8**

2. STATE OF THE ART **8**

2.1. Ontology tools 8
2.1.1 Protégé *8*
2.1.2 OntoEdit *9*
2.1.3 OilEd *9*
2.1.4 WebOnto *9*
2.1.5 WebODE *9*
2.1.6 OntoLingua *10*
2.1.7 OntoSaurus *10*
2.1.8 TopBraid Composer *10*
2.1.9 WordNet *10*
2.2. Ontology Visualization 11
2.3. Conclusion 11

CHAPTER 3 ..... **13**

3. UNIVERSAL KNOWLEDGE BASE **13**

3.1. Universal Knowledge Organization 13
3.2. Knowledge integration, evolution and maintenance 15
3.3. Knowledge Management System Description 17
3.4. UK Management System 17
3.5. System Architecture 18
3.6. Evaluating the platform in real world scenarios 19

CHAPTER 4 ..... **20**

4. LINGUISTIC AND CONCEPT KNOWLEDGE **20**

4.1. Organization of the Linguistics Knowledge 20
4.1.1 Word *21*
4.1.2 Synset *21*
4.2. Organization of the Conceptual Knowledge 22
4.2.1 Concept *22*
4.2.2 Concept hierarchy *23*
4.3. Conclusion 24

CHAPTER 5 ..... **26**

5. DOMAIN KNOWLEDGE **26**

5.1. DERA 27
5.1.1 Domain *27*
5.1.2 Entity *28*
5.1.3 Relation *28*
5.1.4 Attribute *29*
5.2. Facet Hierarchy 30
5.3. Conclusion 31

CHAPTER 6 ..... **32**

6. ENTITY TYPE **32**

6.1. FUNDAMENTAL BASIS OF ENTITY AND ETYPE 32
6.2. PRIMITIVE NOTION OF ENTITY TYPE 32
6.3. OVERALL ASPECTS OF ETYPE MODEL 32
6.3.1 Data Types 32
6.3.2 Domain Restrictions 33
6.3.3 Attribute 34
6.3.4 Entity Types 35
6.3.5 Entity 35
6.4. TYPES OF ATTRIBUTES 35
6.5. META ATTRIBUTES 36
6.5.1 Necessity Properties: 36
6.5.2 Permanence Properties: 36
6.6. ETYPE ATTRIBUTES CATEGORIES 36
6.7. ETYPE SYSTEM DESIGN REQUIREMENTS 36
6.8. CONCLUSION 37

CHAPTER 7 38

7. VISUALIZATION THEORY 38

7.1. INFORMATION VISUALIZATION 38
7.2. FROM DATA TO INFORMATION VISUALIZATION 38
7.2.1 Data 38
7.2.2 Information 38
7.2.3 Knowledge 38
7.2.4 Wisdom 39
7.2.5 Information visualization 39
7.3. PERCEPTION 39
7.3.1 Gestalt Principals 39
7.4. MEMORY 43
7.4.1 Sensory Memory 44
7.4.2 Short-Term Memory (STM) 44
7.4.3 Long-Term Memory (LTM) 45
7.5. A REFERENCE MODEL 45
7.5.1 Raw Data 45
7.5.2 Data Structure 46
7.5.3 Visual Mapping 46
7.5.4 Views 47
7.6. CONCLUSION 48

CHAPTER 8 49

8. LINGUISTIC AND CONCEPT KNOWLEDGE MANAGEMENT CONSOL 49

8.1. FUNCTIONAL OVERVIEW AND REQUIREMENT DESCRIPTION 49
8.2. SEARCH AND NAVIGATION 51
8.3. SEARCH AND NAVIGATION OF WORDS 51
8.4. SEARCH AND NAVIGATION OF SYNSETS 52
8.5. SEARCH AND NAVIGATION OF CONCEPTS 53
8.6. MANAGEMENT FUNCTIONALITIES (CREATE/UPDATE/DELETE) 55
8.7. MANAGING WORDS, SYNSETS AND CONCEPTS 56
8.7.2 Managing Relations 63
8.8. DEALING WITH MULTIPLE LANGUAGES 64
8.8.1 General View 65
8.8.2 Spotlight View 65
8.9. DESIGN PRINCIPLE AND IMPLEMENTATION OF THE CONSOLE 66
8.10. DETAILED DESCRIPTION OF THE SEARCH AND NAVIGATION PANELS 67
8.10.1 Words panel 67
8.10.2 Senses panel 68
8.10.3 Concepts panel 70
8.11. DEALING WITH MULTIPLE LANGUAGES 74
8.11.1 General View: 74
8.11.2 Spotlight View: 74
8.12. DETAILED DESCRIPTION OF THE CUD PANEL 75

8.12.1 Manage Word: ... *75*
8.12.2 Manage Sense: ... *75*
8.12.3 Manage Relation: ... *76*
8.13. MANAGING WORDS AND SENSES/SYNSETS ... 76
8.13.1 Create a synset (and corresponding concept) as child of an existing concept ... *76*
8.13.2 Create a synset to be associated with a concept with no synset in the language ... *78*
8.13.3 Create a synset as a new root concept: ... *78*
8.13.4 Update a Synset: ... *78*
8.13.5 Delete a Synset ... *80*
8.14. MANAGE RANK ... 81
8.14.1 Update Word Rank: ... *81*
8.14.2 Update Sense Rank: ... *81*
8.15. MANAGING CONCEPTS AND RELATIONS ... 81
8.16. LIST OF ICONS AND THEIR MEANING ... 82
8.17. CONCLUSION ... 82

CHAPTER 9 ... 83

9. DOMAIN MANAGEMENT ... 83

9.1. INTRODUCTION ... 83
9.2. FUNCTIONAL DESCRIPTION OF THE DOMAIN MANAGEMENT SYSTEM ... 83
9.3. SEARCH AND NAVIGATION ... 85
9.3.1 Search and navigation of Domains ... *85*
9.3.2 Search and navigation of Facets ... *86*
9.3.3 Search and navigation of Facet hierarchy ... *86*
9.4. DOMAIN MANAGEMENT FUNCTIONALITIES (CREATE/UPDATE/DELETE) ... 87
9.4.1 Managing Domains, Facets and Facet hierarchy ... *87*
9.5. GENERAL OVERVIEW OF THE CONSOLE ... 88
9.6. DETAILED DESCRIPTION OF THE SEARCH AND NAVIGATION PANEL ... 89
9.6.1 Domains Panel ... *89*
9.6.2 Facets Panel ... *90*
9.6.3 Facets hierarchy Panel ... *92*
9.6.4 Pop Up window for concept selection ... *93*
9.6.5 Pop Up window for facet selection ... *93*
9.7. DETAILED DESCRIPTION OF THE CUD PANEL ... 94
9.7.1 Manage Domain ... *95*
9.7.2 Manage Domain facet ... *95*
9.7.3 Manage Facet Relation ... *95*
9.7.4 Create Facet ... *96*
9.8. MANAGING DOMAINS AND FACETS ... 96
9.8.1 Create a Domain ... *96*
9.8.2 Update a Domain ... *97*
9.8.3 Delete a Domain ... *97*
9.8.4 Add a facet to a domain ... *97*
9.8.5 Removing a Facet from a Domain ... *98*
9.8.6 Updating the category of a facet: ... *98*
9.8.7 Create a Facet ... *98*
9.8.8 Delete a Facet ... *99*
9.8.9 Add a node to a facet hierarchy ... *99*
9.8.10 Remove a node from a facet hierarchy ... *100*

CHAPTER 10 ... 101

10. ENTITY TYPE MANAGEMENT CONSOL ... 101

10.1. FUNCTIONAL OVERVIEW AND REQUIREMENT DESCRIPTION ... 101
10.2. SEARCH AND NAVIGATION ... 102
10.2.1 Search and navigation of attributes ... *102*
10.3. SEARCH AND NAVIGATION OF ETYPES ... 103
10.4. SEARCH AND NAVIGATE ATTRIBUTES OF A ETYPE ... 103
10.5. MANAGEMENT FUNCTIONALITIES (CREATE/UPDATE/DELETE) ... 103
10.6. GENERAL OVERVIEW OF THE CONSOLE ... 103
10.6.1 Etype navigation and management Tabs ... *104*

10.6.2 Attribute navigation and management Tab *104*
10.6.3 Pop Up window for concept selection *105*
10.6.4 Pop Up window for selecting domain restriction *105*
10.7. DETAILED DESCRIPTION OF ETYPE NAVIGATION AND MANAGEMENT TAB 106
10.7.1 Etypes panel *106*
10.7.2 Manage Etype panel *107*
10.7.3 Manage Etype Attribute panel *107*
10.8. DETAILED DESCRIPTION OF ATTRIBUTE NAVIGATION AND MANAGEMENT TAB 108
10.8.1 Attributes search and navigation panel *108*
10.8.2 Add/Edit Attribute panel *110*
10.9. MANAGE ATTRIBUTE 110
10.9.1 Create an Attribute *110*
10.9.2 Update an Attribute *111*
10.9.3 Delete an Attribute *111*
10.10. MANAGING ETYPES AND THE ATTRIBUTES ASSOCIATED WITH AN ETYPE 111
10.10.1 Create an Etype *111*
10.10.2 Update an Etype *112*
10.10.3 Delete an Etype *112*
10.10.4 Add an attribute to an Etype *113*
10.10.5 Update the property of an attribute of an Etype *113*
10.10.6 Removing an Attribute from an Etype *114*

CHAPTER 11 115

11. EVALUATION FOR LINGUISTIC AND CONCEPT KNOWLEDGE MANAGEMENT CONSOL 115

11.1. EVALUATION METHOD 115
11.2. PERFORMANCE EVALUATION 116
11.3. EFFECTIVENESS EVALUATION 118
11.4. SATISFACTION EVALUATION 118

CHAPTER 12 121

12. CANET: A STRUCTURED DISTRIBUTED BACKGROUND KNOWLEDGE INFRASTRUCTURE FOR SUPPORTING SEMANTIC SEARCH 121

12.1. INTRODUCTION 121
12.2. SEARCH PROBLEMS 122
12.2.1 Low Precision *122*
12.2.2 Low Recall *122*
12.3. SEMANTIC SEARCH APPROACH 122
12.3.1 From natural language to formal language *123*
12.3.2 From words to phrases *123*
12.3.3 From string similarity to semantic similarity *123*
12.4. LIMITATIONS OF CURRENT APPROACHES 124
12.5. STRUCTURED P2P 124
12.6. CANET 125
12.6.1 Distributed Background Knowledge *126*
12.6.2 Indexing *126*
12.6.3 Query Answering *127*
12.7. EXAMPLE SCENARIO 128
12.8. RELATED WORK 129
12.9. PERFORMANCE EVALUATION 130
12.10. CONCLUSION 131

CHAPTER 13 132

13. CONCLUSION 132

BIBLIOGRAPHY 135

APPENDIX A: USABILITY EVALUATION TASKS 141

APPENDIX B: HEURISTIC FEEDBACK 144

List of Tables

Table 1 List of possible relations maintained by the concept part of our system thus classifying it according to their hierarchical and associative feature. 24
Table 2 List of result for the search token “comput” 51
Table 3 Keyboard Navigation 72
Table 4: List of domains 85
Table 5 Aggregate of task package 1, navigation on words, sense and concepts. 116
Table 6Aggregate of task package 2, CRUD on ontology 117
Table 7 Satisfaction matrix being used. 118
Table 8 This summary table shows a very high satisfaction level of using the system. 119

List of Figures

Figure 1 is shown a conceptual view of the UK. The whole structure is built around the concept core which can be seen as the core which glues all the other components. 5
Figure 2 is shown a conceptual view of the UK. The whole structure is built around the concept core which can be seen as the core which glues all the other components. 14
Figure 3 High level architecture of knowledge management system 18
Figure 4 Language independent concept representation of the synset of Italian word fiore and the English word flower 23
Figure 5 Conceptual model of Linguistic and Concept Knowledge 24
Figure 6 : Space domain (partial view) 30
Figure 7 Principale of Proximity 40
Figure 8 Principle of Uniform Connectedness 40
Figure 9 Principle of Similarity 41
Figure 10 Principle of Continuity 41
Figure 11 Principle of clouser 42
Figure 12 Principle of Symmetry 42
Figure 13 Principle of Figure and Ground 43
Figure 14 Memory model explaining some steps involved in sensory, long-term and short-term memory 44
Figure 15 Hierarchical tree browser with a horizontal layout for a file system 52
Figure 16 Concepts Panel 55
Figure 17 Gap in the language 59
Figure 18 Linguistic and Conceptual Knowledge management console 66
Figure 19 Words Panel 68
Figure 20 Senses Panel 69
Figure 21 Sense block with multi language 70
Figure 22 Concepts Panel 71
Figure 23 Concept node details 72
Figure 24 Multilanguge Selection 75
Figure 25 CUD Panel 75
Figure 26 Manage Rank 81
Figure 27 CUD panel (Manage relation) 81
Figure 28 Domain Management Console 88
Figure 29 Domain Palen 89
Figure 30Facet Panel 91
Figure 31 Facet Hierarchy Panel 92
Figure 32 Select concept selection pop up window 93
Figure 33 Select facet pop up window 94
Figure 34 CUD Panel 94
Figure 35 Etype Management Console 104
Figure 36 Etype Attribute Management Console 105
Figure 37 Domain Select Window 105
Figure 38 Etypes Panel 106
Figure 39 Manage Etype Panel 107
Figure 40 Manage Etype Attribute Panel 107
Figure 41 Attribute search and navigation 109
Figure 42 Manage Attribute 110
Figure 43 Comparison of performance for each task. 117
Figure 44 CRUD on ontology. 117
Figure 45 Features rated by User 2 119
Figure 46 Features rated by User 1 120
Figure 47 CANet Layers 125
Figure 48 An example of the peers index information stored on the peer responsible for a single word canine and for a single atomic concept canine-2 128
Figure 49 Query answering 129

Figure 50 Query propagation time over number of peers and concept depth. 130
Figure 51 Time increases with the increase of concept depth. 131

Chapter 1

1. Introduction

1.1. The Context

The amount of knowledge and information in today's world is growing very fast. The progress of information and communication technologies has given access to this huge amount of information, which is increasingly becoming difficult to comprehend or manage. Different people having different viewpoints and using different terminology among people of different cultures and languages intensify the interoperability problem even more. In the current era, especially with the start of the Web and the consequential information explosion people face the concrete problem to retrieve, disambiguate and integrate information coming from a wide variety of sources. There are many communities which independently try to solve the same problem, namely making information available, and understandable, to those who seek for it. Nevertheless, these problems are very well known in almost every branch of knowledge and have been independently approached by many communities for several decades. Unfortunately, this is not an easy task due to differences among systems and among users which lead to the more general problem of interoperability.

Hypertext and hypermedia systems have gathered a discrete success. As known, they provide a way to explore texts, documents and other kinds of media, such as images, videos and music, in a not sequential order. The web is a clear example of the success of this approach, mainly due to its simplicity of use. People can easily add new content and links between pages. Unfortunately, these systems force the user to follow only predefined paths and provide poor semantics for document relations moreover and typically reflects the creator mental model rather than the user model [23]. Therefore, it is extremely difficult to locate relevant information [9]. More recently, in the spirit of the Web 2.0, people try to approach the problem collaboratively. An example is represented by social tagging initiatives[1]. Tags are labels that users can utilize for annotating objects in order to categorize information. These categories can be seen as different orthogonal perspectives in which users can progressively add new objects. Unfortunately, users tend to invent personally meaningful tags not always comprehensible to everyone (this problem is usually referred as the vocabulary problem) and to reuse the same tags over and over. For these and other reasons, many researchers [9][14] claim that these and similar techniques, usually referred as classical Information Retrieval (IR) techniques, are becoming inadequate. Among other things, Soergel observed that in effect there are not substantial differences among them, but their differences are only a matter of degree [16]. However, according to Hjorland [24] IR is just one of the means to Knowledge Organization (KO), which in turn is seen as the set of activities such as document description, indexing and classification. Broughton et al. [25] say that a regular way to distinguish about KO and IR is that the former is more about indexing, while the other is more about searching. According to Vickery, the aim of KO is to make knowledge available to those who seek it [26]. This purpose is achieved by using effective 'finding aids' such as classifications and search engines.

In the early 1990's, an attempt to make interoperability standards identified a technology stack that called out the ontology layer as a standard constituent of knowledge systems [5]. Different disciplines define ontology in different ways. In philosophy, ontology refers to the *subject of existence*. In Computer Science it defined as a *specification of a conceptualization* [3]. A conceptualization is an abstract spectacle of the objects, concepts, entities that are assumed to be present in some area of interest and the relationships that hold among them which we aimed to represent for some purpose. In Semantic Web Ontologies are core artifacts for representing information, in which information is prearranged with formal semantics so that computers can use inference rules to perform automatic reasoning on pieces of this in-

[1] http://del.icio.us/; http://www.facetag.org/; http://www.facebook.com/

formation [2]. Main constrain of the formal ontology is it often takes tremendous effort to create due to the level of detail and complexity required [7], also required highly trained knowledge engineers with the support of domain specialists. It's time-consuming and difficult task. Ontology tools also require users to be trained knowledge representation and predicate logic.

The area of classification has been used for centuries cataloguing and searching large sets of objects , primarily books in the early days and lately becomes very popular in order to provide a structure for organizing and navigating information such as Web objects (e.g., web pages, documents, pictures), commercial products. Documents are indexed according to a hierarchical tree structure, under topic nodes. The information base can be trivially used as a browsing structure to inspect all the items classified under a specific node. In the web environment, concrete examples are represented by DMOZ, Yahoo and Google[2] web directories. In the field of LIS we can mention traditional enumerative classification systems, like the Dewey Decimal Classification[3] (DDC) and the Library of Congress Classification[4] (LCC), which tend to exhaustively list all known subjects. However, the resulting structure is too rigid and a seeker generally browses among hundreds of information atoms before locating significant information [13]. Moreover, the contents of the classifications are using natural language labels which has proved very successful in manual classification but show their limitations when one tries to automate the process, as they are not formally defined. Informalities make it almost impossible to reason about classifications and their contents. In LIS domain specific thesauri play a similar role. Their aim is to provide knowledge about a single domain, in terms of basic subjects and relationships between them. Among them a central role is represented by faceted classifications, proposed and formalized at the beginning of the last century by Ranganathan, an Indian librarian, in the theory of facet analysis. This theory is widely recognized as the fundamental methodology that guides in the creation of a faceted classification in a given domain [9]. According to his analythico-syntetic approach [11], the first step is to examine the field to identify relevant terms (analysis). They can be gained consulting domain experts and all sorts of information sources over the domain, taking into account some fundamental categories (originally they were Personality, Matter, Energy, Space and Time). The second step is to group (synthesis) the terms in hierarchies according to their common properties or characteristics, and arranged in a helpful sequence. These hierarchies are known as facets, which represent the distinctive dimensions of knowledge in that domain. The whole process starts in the so called "idea plane", the language independent conceptual level, where concepts are identified; it is expressed in the "verbal plane" in a given language trying to articulate the idea *coextensively*, namely identifying a term which exactly and unambiguously expresses the concept; and ends in the "notational plane" where a unambiguous notation is used to synthetically attach meaning and provide order to managed objects, typically books on the shelves.

Lightweight ontologies [3], tried to bridge the gap between informal classifications and formal ontologies. *Lightweight ontologies* are simple; therefore it is easy to create. A classification can be converted into *Lightweight ontology* by converting its natural language node labels into concepts, which are represented in a formal language, using Description Logic languages [22]. Based on their content type *lightweight ontologies* can further be classified into two categories: *descriptive lightweight ontology* and *classification lightweight ontology*. *Descriptive lightweight ontologies* are basically Thesauri, controlled vocabularies, which used to specify the semantics of terms, and the nature and structure of the domain the terms belong to [1]. Web directories, user classifications are the example of *classification lightweight ontologies* . They are basically used to describe, categorize, and access collections of documents. One of the main differences is in the later one the ontology includes only high level concepts also called *complex concept* consist with compound noun phrase. The interpretation of this *complex concept* is captured as context descriptions outside the ontology whereas the former one usually have a noun word or a simple noun phrase in each of their node labels as an atomic concept. Some key applications of *formal lightweight ontologies* include (but are not limited to) document classification, semantic search, data integration [8]. However to make use of a lightweight ontology for interoperability purposes requires agreements on the same meaning of the concepts between all the parties, which introduce the new challenge.

[2] http://dmoz.org/; http://dir.yahoo.com/; http://directory.google.com/
[3] http://www.oclc.org/dewey/
[4] http://www.loc.gov

Semantics is going to play a more and more principal role in the future world. The famous vision paper by Tim Berners-Lee [2] on the Semantic Web underlines the advantages in moving towards this direction. Natural language is ambiguous. Before applying any form of automated reasoning it is fundamental to shift to a formal language. Attaching the meaning to each single information item allows automating complex tasks and is the only way to make manageable the huge amount of available information. In many fields, for instance in KO, this is achieved by providing some authority controlled vocabularies (e.g. thesauri) and asking users to annotate, index and search information items using standard terms. According to Vickery [26], even if there is still controversy, the knowledge of a domain in terms of concepts and relationships between them helps the organization and retrieval of documents mainly because it allows dealing with different levels of specificity, allows linking related terms together, helps in the identification of a semantically meaningful sequence and fixes the controlled vocabulary. On the other hand, he agrees with Soergel that current Knowledge Organization systems (KOS) lack of conceptual abstraction, there is no distinction between concepts and their lexicalization in words, they use a limited undifferentiated set of relationships, and as a consequence they provide limited support for automated processing (see also [26] for an extensive description). Even if not perfect, they represent very valuable expert knowledge, usually over a restricted domain. In fact, their construction is very costly and attempts on automatically building domain specific thesauri are not so promising [28].

Matching performs a major task in approaches that rely on ontologies to solve the semantic heterogeneity problem between information systems [17, 18, 19, 20]. An essential pre-requisite for the accomplishment of such matchers is the availability of *background knowledge* sources with a sufficient coverage of the information. Background knowledge represents a set of true facts that used by semantic tools as an external resource to bridge the semantic gap between the matched ontologies and reach to conclusions. For example it may possibly contain that tiger is an animal or that Venice is a city and it is part of Italy. Latest evaluations of matching systems demonstrate that insufficient background knowledge is one of the major problems of matching systems these days. In particular background knowledge mostly builds for supporting specific domain. In practice many semantic match applications relay on readily available, non-specific resources, such as WordNet, as background knowledge [19]. Unfortunately, despite their broad coverage, they show insufficient coverage for many areas, for example if we consider geo-spatial systems, information like latitude, longitude coordinates are the primary importance.

The knowledge is reproducible and sharable. The reproducible nature of knowledge makes it very dynamic. Something we know as true today, tomorrow appear as false. It keeps on changing every moment of time, in the same time growing as well. To get the proper benefit of this ever-growing knowledge it becomes essential to develop efficient and easy to use systems to access and navigate information [13,14,15,16]. This in turn requires the building of a high quality but flexible knowledge base system, which takes into account diversity of knowledge and its evolution in time. The aspiration of the building of a full agreement universal knowledge schema, describing the reality in an undisputable way, has been cultivating in fields spanning from Philosophy, Library Information Science (LIS), Artificial Intelligence (AI), and more recently in the Semantic Web as we have mentioned already. As an expression of the problem itself they use different terms to name it. In Philosophy it is known as the theory of categories[5] which originates from Aristotle two thousand years ago; in LIS this is called Universal Thesaurus 10 or Universal Classification and has been addressing with particular fervor in the last two centuries; in AI it is called Universal Knowledge Base; in the Semantic Web it is named Universal Ontology or Global/Upper Schema. The universal schema is meant to provide a common ground on which interoperability among people and different systems should be possible. It plays the role of a standard for communication, assigning clear meaning to exchange information. Unfortunately, all attempts till now failed because it is practically impossible to reach the full agreement, unless in very limited scope [2]This is the reason why in many fields some criteria to limit the scope are defined, for instance the notion of domain in LIS[11]and the notion of context and generality in AI [12].

[5] http://plato.stanford.edu/entries/aristotle-categories/

1.2. The Problem

From this preliminary analysis we identified some problem areas of current knowledge management systems which need to improve to support the interoperability among different cultures and languages in an unambiguous way.

Lack of background knowledge: As we said in Section 1.1, the applications of formal lightweight ontologies, semantic matching techniques highly depend upon the reasoning on the axioms extracted from a knowledge base. Due to the insufficiency of the background knowledge all the necessary axioms cannot be extracted. This is also a main reason for the relatively low recall and precision of semantic search applications [21].

Disambiguation accuracy of natural language processing: The creation process of *lightweight ontologie* from informal *classifications* requires the processing of natural language labels for the identification of their concepts. NLP tools are basically use for processing natural language labels and disambiguate them. However, even the state of the art NLP's disambiguation precision is limited to a certain degree because of the lack of coverage of the lexical knowledge base.

Lack of Entity management tools: Ontology conceptualization process includes also entities i.e., real world objects. Entities are defined in terms of some meta-properties which characterize the entity called *entity type*. However, there are no tools that can use to define the *entity type*.

Dynamics : The knowledge is reproducible and sharable. The reproducible nature of knowledge makes it very dynamic. Something we know as true today, tomorrow appear as false. It keeps on changing every moment of time, growing as well. To get the proper benefit of this ever growing knowledge it becomes fundamental to develop efficient and easy to use systems that would enable us to cope up with this dynamics while ensure the consistency.

Limitation of integrated management environment: There exist many tools to facilitate the knowledge Management but no one comes with the complete support of lexicon, concepts, entity types and ontology as an integrated environment. Most of them come with partial support, many of them are restricted to organizations or domain specific. They are largely based on superimposed models, structures and terminology. They are not so suitable for managing knowledge in a collaborative way thus support interoperability. They lack of semantics, namely users cannot accurately express they needs, annotate their objects nor interpreter information they handle and exchange in an unambiguous way.

Lack of support for distributed management: Due to the difficulties and costs involved in building ontologies manually, the adoption and their use is limited. Building large scale knowledge base with maximum coverage is not possible without collaborative support. The more people will get involve to share the knowledge base will get enriched the more. It also requires distributed management support and knowledge synchronization. Unfortunately, most of the existing systems do not have this support.

1.3. The Solution

Human knowledge is a living organism and as such evolves in time. It is obvious that in order to let people interoperate a certain degree of conformity between them is required. In our opinion, the only meaningful way to approach the semantic heterogeneity problem is to allow and support knowledge diversity in language, culture, purpose and belief. This can be achieved by helping people to organize and make accessible their knowledge and create effective and efficient tools to support interoperability. However, effectively building large scale knowledge base with maximum coverage is not possible for a single person or a small group of people without collaborative support. It is extremely depends on expert community based support. Therefore, it is necessary for experts to work together on knowledgebase building. Furthermore, it is very natural that these expert users will be geographically distributed. Web 2.0 has the potential to support information sharing, interoperability and collaboration [30] on the Web. The Web 2.0 services [30] enable users not only interact with the content of the web pages but also between themselves in a decentralized and more open manner, which make it a planet where connectivity plays the key role which makes the users to be an integral part of this planet. Simplicity, flexibility and easy to use services make it an interactive and collaborative platform, which allows them to create or edit their content. Current statistic shows that there are over 1.9 billion internet users worldwide [29]. The exponential expansion of the Web users and the potentials of Web 2.0 make it the natural platform of choice for developing Knowledge base collaboratively.

What we propose is a system with three different levels of knowledge, the universal, background and user knowledge:

- ***The universal knowledge (UK):*** it represents the knowledge composed from a wide variety of sources over different domains. It is universal in the sense that
  - [1] it represents expert validated knowledge over as much domains as possible;
  - [2] it is supposed to be widely shared by the majority of (but not necessarily all) users;
  - [3] it can be used as a starting reference point to construct specialized schemes and can be further personalized locally;
- ***The background knowledge (BK):*** it represents a subset of the knowledge acquired from UK (for example selecting one or two domains) which is customized locally by a user or a restricted community of users;
- ***The user knowledge***: in order to manage their collection of documents, users can build their own classifications. The system will support the construction process and the interpretation of their semantics using BK. This is achieved by translating them into lightweight ontologies (see for instance [31]).

The system will be able to manage knowledge allowing users to personalize and evolve it at different levels, taking into account diversity dimensions and forward customizations back and forth the knowledge chain providing support for synchronization at different levels. Different users will be able to perform local and global searches on their and other users' classifications, establish unidirectional semantic relations between them, navigate across classifications, and run complex semantic services on top of them.

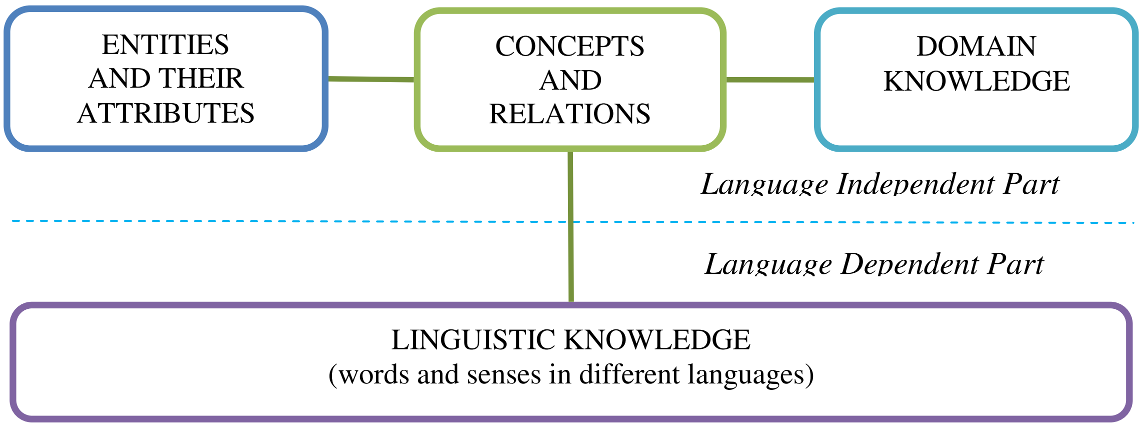

ENTITIES AND THEIR ATTRIBUTES
CONCEPTS AND RELATIONS
DOMAIN KNOWLEDGE
Language Independent Part
Language Dependent Part
LINGUISTIC KNOWLEDGE
(words and senses in different languages)

Figure 1 is shown a conceptual view of the UK. The whole structure is built around the concept core which can be seen as the core which glues all the other components.

Since this is a huge amount of work, in particular this thesis focused on developing a set of tools for the construction and maintenance of the ***universal knowledge (UK)*** part. Precisely, it will not be an isolated tool for the development of ontologies, rather would be driven towards the creation of a common integrated environment that provides various knowledge base related services, and gives support to most of the activities involved in the knowledge base construction, evaluation and management. Figure 1 shows a conceptual view of the UK. The whole structure is built around the concept core that can be seen as the core, which glues all the other components. Notice that, *UK* and *BK* will have the same structure and the construction of the *universal* and *local* background knowledge (i.g., UK and BK) will provide that right amount of flexibility necessary to take into account all dynamics connected to the knowledge management (creation, indexing, browsing and searching, evolution and transfer) taking into account different viewpoints. Domain expert will manage UK whereas BK will be managed and customized locally by any expert or no expert user locally. BK will represent a subset of the knowledge acquired from UK. At the best of our knowledge, no comparable systems have been built till now.

1.4. Structure of the Thesis

The thesis is organized as follows:

In Chapter 2 we discuss the state-of-the-art ontology editing tools and visualization methods.

In Chapter 3 we discuss the high level architecture of the Universal Knowledge Base , In Section 3.1 we describe the knowledge aacquisition and integration as a part of bootstrapping the UK and the system descriptions.

In Chapter 4 we discuss the basic building blocks of linguistic and concept knowledge and finally the requirement specification of the linguistic and concept knowledge management system.

In Chapter 5 we discuss about the domain knowledge. In Section 5.1 we describe the DERA framework, In Section 5.2 we describe the faceted hierarchy build under DERA framework, we also show an example of “Space” domain modeled according to the theory of DERA and domain knowledge management system.

In Chapter 6 we discuss about the etype theory. In Section 6.1 and 6.2 we describe the fundamental basis and primitive notion of etype. In Section 6.3 we describe the etype model. In Section 6.3 and 6.4 we describe the type of attributes and meta attributes respectively. In Section 6.5 we describe the etype attribute category and finally the requirement specification of the etype management system.

In Chapter 7 we present the basic visualization theories, which includes gestalt principles, we also discussed about different types of memory from cognitive psychology point of views and a reference model.

In Chapter 8 we present the detail description of the Linguistic and Concept Knowledge Management Consol

In Chapter 9 we present the detail description of the Domain Knowledge Management Consol

In Chapter 10 we present the detail description of the Entity Type Management Consol.

In Chapter 11 we present the Evaluation for Linguistic and Concept Knowledge Management Consol

In Chapter 12 we describe a model, Distributed Background Knowledge Infrastructure to enable semantic search. This Chepter organized as follows. Section 12.1 expands more on the Search problems. Section 12.3 describes the Semantic search approach, specifically *CSearch*. Section 12.4 described the limitation current approaches. Section 12.5 describes how syntactic search can be implemented on top of Structured P2P. Section 12.6 describes our solution *CANet* overlay followed by a example scenario. Section 12.8 we compare our work with other related work. In Section 12.9 we presented an evaluation of our current implementation and thus present the conclusion and the future work

Chapter 2

2. State of the Art

Ontologies describe the concepts and the relationships between concepts which are used to represent a variety of domain knowledge, different viewpoints, scopes, and linked heterogeneous information sources, providing a vocabulary for those domain and a specification of the meaning of terms used in the vocabulary. Ontologies range from straightforward simple taxonomies, classifications, database schemas, to very complex with rich constructors of Description Logic, highly structured knowledge bases with complex relations, which may differ not only in their content, but also in their way of organization and implementation. In these days ontologies have been taking on not only in the research communities but also in many business communities as a means to share, reuse and process domain knowledge. It has proven to be a useful tool in the area of Knowledge management to build knowledge bases. As a result, there is a growing need for a system that provides various knowledge base related services, effective visualization, search and navigation and gives support to most of the activities involved in the knowledge base construction process. Many ontology models support multiple inheritances in the concept and relation hierarchies, Effective presentations of ontology's interlinking concepts and relations with multiple inheritances are challenging. There exist many tools with ontology building methodologies, several visualization methods for ontology visualization and navigation and also a number of languages are by now available for knowledge representation. The purpose of this chapter is to present these tools, techniques and their features in order to understand current trained and future research in the area of Knowledge management.

2.1. Ontology tools

2.1.1 Protégé

Protégé [32] is a free, open-source platform that provides the system developers and domain experts with a suite of tools to construct domain models and knowledge-based applications with ontologies. Stanford Medical Informatics develops it. It can be extended by way of a plug-in architecture and a Java-based API. For example OntoViz [33] is visualization plug-in which use the GraphViz[6] library to create 2D graph visualization. It shows classes and instances grouped with their properties and their information. The relationship between objects is presented by the connected edges of the groups. However, Katifori et al [34] shown by an evaluation using an ontology of approximately 250 nodes that when the number of nodes increases OntoViz becomes cluttered. To browse the ontology another visualization plug-in for the Protégé is Jambalaya[7], comes with zooming feature, it shows information of classes grouped by their properties. Instances are showed as nested nodes in their corresponding class in the graph where instances and classes are differentiated with color. Zooming feature helps the user to see details in a convenient way. It also provides keyword search. One good aspect of Protégé is it is possible to customize for creating knowledge models and entering data. Hence, it can be easily extended to use knowledge-based embedded applications. However [35] found that visualization support in Protégé and its customization models are very complicated which do not reflect users' models of what they would usually desire to see. At its core, its support the construction, visualization, and manipulation of ontologies in different representation formats by a set of knowledge-modeling structures and actions. It provides two different editors namely *Protégé-Frames editor* and *Protégé-OWL editor* for modeling ontologies in two different ways. The former one enables users to construct and populate ontologies that are frame-based, in accordance with the Open Knowledge Base Connectivity protocol (OKBC). In this model, ontology consists of a set of classes organized in a subsumption hierarchy to represent a domain's salient concepts, a set of slots associated to classes to describe their properties and relationships, and a set of instances of those classes. The later one, i.e., *Protégé-OWL editor,* is to build ontologies in the W3C's Web Ontology Language

[6] http://graphViz.org
[7] http://www.thechiselgroup.org/jambalaya

(OWL). There is also a web tool available, WebProtege[8], which is a web client for Protégé, a Java-based Web application that allows users to share, browse Protégé ontologies, browse, and do some basic editing of Protege knowledge bases via the internet without having to install the Protégé application locally. Protégé 2.0 has new multi-user capabilities deliberate for experienced users. However, there's no support for multiple users attempting to change the same objects of a knowledge base or notification of changes made by other users that might cause severe problems. It also suffers from lacks of support for collaborative ontology engineering except partial methodological and collaborative support. [36] Has conducted a usability evaluation which shows that users in general having difficulties with description logic based formalisms.

2.1.2 OntoEdit

OntoEdit [37] is developed by AIFB in Karlsruhe University that supports the development and maintenance of ontologies by graphical means. There are two versions of OntoEdit are available: freeware and professional versions. It is an extensible plugin architecture based environment, which provides supports to browse and edit ontologies. It allows importing and exporting FLogic, XML, DAML+OIL and RDF(S) ontologies. One of its plug-in is Mind2Onto for supporting brainstorming and discussion about ontology Structures. The commercial version use databases for storing ontologies.

2.1.3 OilEd

OilEd [38] is developed as a freeware ontology editor for OIL ontologies as a part of the European IST On-To-Knowledge project. It is an editor that allows a user to create, edit and browse OIL ontologies, but it does not provide a full ontology-development support for large-scale ontologies. In particular, it doesn't support activities such as the construction of large-scale ontologies, version control, augmentation, and the integration of ontologies that are involved in ontology building as stated in [39]. Though it was designed to be the very simple ontology editor, it has evolved and at the present it is an editor of DAML + OIL ontologies. Additionally it allows the user connect to the Fast Classification of Termi-nologies (FaCT) [40] reasoner which provides ontology consistency checking and automatic concept classification features and export ontologies in a number of formats. Regarding changes of the ontology it gives an activity log with records connections to the reasoner, but not all ontology modifications.

2.1.4 WebOnto

WebOnto [41] is an ontology editor for ontologies represented in the knowledge modeling language OCML(Operational Conceptual Modelling Language), developed by the Knowledge Media Institute of the Open University, England. The main advantage of WebOnto is that it supports the collaborative browsing using a graphical interface, editing and creation of ontologies. It also provides automatic generation of instance editing forms from class definitions and consistency checking.

2.1.5 WebODE

WebODE [42] is developed in the Artificial Intelligence Lab from the Technical University of Madrid, Spain. It was made to use and test the methontology methodology. It is an ontology-engineering suite build with an extensible architecture which supports ontology-development and management; WebODE is used as a Web server with a Web interface which is build on three tiers architecture where the first tier provides the user interface, the second tier provides the business logic, and the third tier consists of the data. WebODE provides exportation and importation services from and into XML for the ontologies build with it [42]. It also provides a set of translation services into and from various ontology specification languages (RDF(S) , OIL , DAML+OIL , X-CARIN and FLogic) [8].

[8] http://protegewiki.stanford.edu/wiki/WebProtege

2.1.6 OntoLingua

OntoLingua [43] is developed by the Knowledge Systems Lab at Stanford University.It is an OntoLingua is an ontology library and server , which provides a collaborative environment, ontology-authoring tools, and a library of ontologies. The main module of the Ontolingua Server is the ontology editor. Some other modules are such as equation solver, Webster, Open Knowledge Based Connectivity server etc. The environment is available as a WWW service. The ontology-authoring tools are to assist authoring ontologies by assembling and extending ontologies obtained from Ontolingua's library. It also provides translators to languages, such as Loom, Prolog, CORBA's IDL, CLIPS, etc.

2.1.7 OntoSaurus

OntoSaurus[44] was developed by the Information Sciences Institute (ISI) at the University of South California. It provides the users to search and annotate HTML documents with ontological information. Search expressions are improved by using domain-specific thesauri , ontologies. It use Loom as its knowledge representation system and have a web browser

2.1.8 TopBraid Composer

TopBraid Composer [45] is implemented as a standard Eclipse environment plug-in, is a professional modeling environment for developing the W3C's Semantic Web standards RDF Schema, the OWL Web Ontology Language and the SPARQL Query Language. Since Eclipse environment is widely used as a software development tools that makes the TopBraid also to be familiar to the users. There are three versions are available - Free Edition, Standard Edition and Maestro Edition. Free Edition does not include Support and Maintenance The Standard Edition comes with visual editors for RDF graphs and class diagrams together with SPARQL generator in the graph view. It also provides automated conversion of XML, XSD, Excel and UML. While the Free Edition provides support only to work with RDF/OWL files. Maestro Edition is the most comprehensive version of TopBraid Composer, which is optimized for developing web applications and services based on the TopBraid Live platform.

2.1.9 WordNet

WordNet [46] is a large lexical database of English, aimed to create a source of lexical knowledge whose design is inspired by psycholinguistic theories of human lexical memory. WordNet is developed under the direction of Prof. George A. Miller, began in 1985 at Princeton University. In the last two decades, WordNet has evolved as the widespread computational lexicon of general English. In WordNet words and their meanings are related to one another via semantic and lexical relations. Atomic building elements in WordNet are words and synsets. A synset, consisting of all the words with the same meaning whose can be substituted for one another in given types of sentential contexts. Every synset contains a short description (also called gloss) with additionally attached one or more example sentence illustrating the usage of the synset members (i.g., words). Example sentences were added to WordNet synsets because of the meanings of words are depend on the contexts in which they are used. Synsets are linked to one another by means of semantic relations in such a way as to form a lexical semantic network. Because of its well represented lexical meaning it has been used in many natural language processing applications, such as many NLP applications depend highly on word sense disambiguation (WSD) [47,48,49] , Information Retrieval (IR), Information Extraction (IE), text categorization, part of speech tagging [50] as well as applications induced by aligned WordNet and efforts oriented towards enriching WordNet. however many of the advantages offered by wordnet were not initially envisaged by its authors. A number of the main applications of the wordnet over the last ten years are presented in [51].

2.2. Ontology Visualization

For design, management, and browsing of ontology effective visualization is needed. In general, bunches of work in visualization of ontologies exist. IsAViz[9] is a standalone application for browsing and authoring RDF, built on AT&T's Graphviz graph visualization software, is a widely used tool used to visualize RDF metadata. However it has limitations showing overall structure of a set of instances due to their layout. Katifori et al. [34] presented a very comprehensive survey on ontology visualization methods. They tried to review the research that has been done so far on ontology visualization, providing an summary of the existing methods with their pose and cons. They grouped the methods according to the six general categories of visualization types: Indented list, Node–link and tree, Zoomable, Space-filling, Focus+context or distortion and 3D Information landscapes which were further categorized according to the number of space dimensions they employ: 2D or 3D. They investigated how those relate to the special requirements of an ontology visualization tool in relation to the tasks a user would like to perform with an ontology visualization tool. 3D methods exploit the third dimension to improve usage of space and/or usability whereas 2D methods don't. However they argued, 3D visualization in general requires increased system resources in order for navigation and view-ing to be smooth and without delays and, as a result, is probably not suitable for Web use. Fur-thermore, the 3D methods presented here employs more complex navigation methods and may be a little frustrating and disorienting for a novice user. In their discussion part they have shown according to [52] ontology features like the class hierarchy, the role relations, the properties, and the instances, 3D offers the possibility of a better representation while 2D can be somewhat restrictive. As a counter they have said, 3D representations only a little bit improve the screen space problem while increasing the complexity of the interaction [53], further more navigation in a 3D space can be difficult for a novice user, while even simple tasks such as selecting an object can be problematic [54].

Herman et al. [55], presented representations of structured data, in particular the graph visualization techniques were discussed. They categorized graph visualization from the graph drawing or graph layout point of view. They point out the limitation of graph visualization raised due to size of the graph, in particular for big data sets i.e., thousands of nodes, as its makes the graph so dense that interaction with the graph becomes difficult and often make it impossible to navigate. To address the problems with 3D graph visualization techniques they mention inherent cognitive difficulties of 3D navigation in our current systems.

Shneiderman [56] categorizes visualization methods based on the data-type of the elements to be represented in the interface (linear, 2 dimensional, 3 dimensional, temporal, multidimensional, tree, network, workspace) and the task characterize based on how users interact with the visualization of a large amount of information: (overview, zoom, filter, details-on-demand, relate, history, extract). If we summarize the idea, it would be, users would begin from an overview of the information space, overview approaches include zoomed out views of each data type to view the whole collection. The part of the collection that seemed to be of important, they zoom in the part. Filtering allow the users to control the displayed contents by removing things that are not needed. When filtering has minimized the collection, it is convenient to look through the particular details and further inspect the relationships among items. During this progressive refinement it is also important to keep the history of actions so that the users can retrace their steps. Once users have found the thing they need, they should have a to save it in any convenient way. These thoughts are interesting as they present information with user controlled discovery.

2.3. Conclusion

In the last few years, many ontology-building tools have been developed, some of which we have presented above. A survey that covers very large numbers (more than 50) of tools that have ontology editing capabilities with comparisons can be found in[57], they have reviewed and analyze into a dozen different categories which covered the important functions and features of those tools . Ontology visualization is a particular sub area of graph visualization that still have many implications because of the inherent com-

[9] http://www.w3.org/2001/ii/IsaViz

plex structure of the ontologies. However it is clear that the main problem of current tools, regardless whether meant for restricted organizations or large communities, is that they are largely based on superimposed models, structures and terminology and many of them have a strong bond with a specific language (e.g., Ontolingua, OCML, LOOM etc). However all these tools and applications have contributed to a lot to the emergent research and development of the ontology community.

Chapter 3

In this chapter we discuss the high level architecture of the Universal Knowledge Base , In Section 3.1 we describe the knowledge aacquisition and integration as a part of bootstrapping the UK and the descriptions of the UK management system.

3. Universal Knowledge Base

The universal schema is meant to provide a common ground on which bases interoperability among people and different systems should be possible. It plays the role of a standard for communication, assigning clear meaning to exchange information. Solving the semantic heterogeneity problem, namely the problem of diversity in knowledge, and therefore support interoperability will require the construction of a high quality knowledge base. In fact, the availability of an appropriate amount of knowledge is clearly a fundamental step for any application that deals with semantics.
Many attempts have been made to build huge knowledge bases. One of the most famous is CYC[59], a huge and complex collection of concepts and axioms. Its declared aim is to provide a critical mass of commonsense knowledge serving as a standard universal schema, underlying the web to make more efficient the retrieval and the integration of information coming from different sources. However, such huge quantity of knowledge is difficult to manage, maintain in time, access, validate, keep consistent and difficult to use mainly due to the complexity of the formal language and reasoning. For this reason, knowledge is typically *partitioned* and reasoning is *localized* in small contexts by decreasing the level of generality. Our goal is to create an integrated environment to facilitate the development and sharing the universal knowledge base that must also assist the user in the advanced development tasks of creating new knowledge, maintaining, sharing, and using them.

3.1. Universal Knowledge Organization

Existing systems tend to internally use heavy representations, for example RDF and OWL, which make reasoning very slow and impractical in real world applications. We think that these formalisms should be used only for knowledge exchange (import/export facilities) while more lightweight representations should be considered for reasoning. We carefully separate UK/BK into four distinctive parts:

- ***Linguistic part***: terms, senses and lexical relations between them in multiple languages;
- ***Ontological part***: concept and semantic hierarchical and associative relations between them;
- ***Domain knowledge***: concepts and relations between them arranged in facet hierarchies codifying the knowledge about a specific domain;
- ***Entity part***: the instances of defined concepts and their attributes;

This system make simpler and scalable the access to and management of the distinct parts and speed up reasoning improving the overall performance of the system. ***UK*** is mainly organized in two distinct parts: a language independent and a language dependent part. Figure 1 shows a simplified view of the Universal Knowledge base. In the figure links between the objects within a part are shown as solid straight arrows and links across the parts are shown as dashed curved arrows. In the language dependent part, linguistic knowledge in several languages (e.g. in English and Italian) is organized, similarly to WordNet, as a set of words clustered into synsets, i.e. groups of words with the same meaning. Each synset is attached with the information about the part of speech it represents, i.e. noun, verb, adjective and adverb.

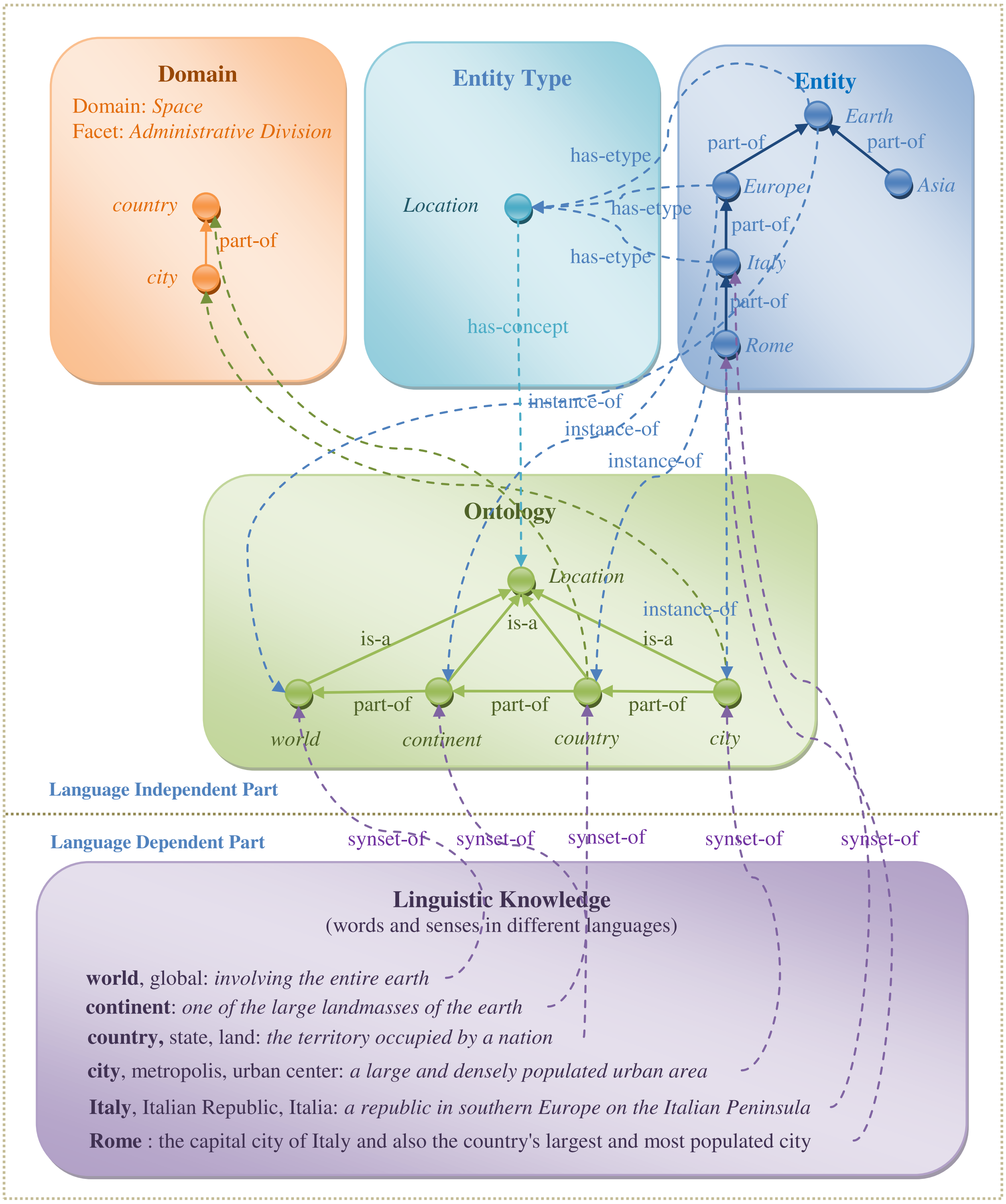

Domain
Domain: *Space*
Facet: *Administrative Division*
country
part-of
city
Entity Type
Location
has-etype
has-etype
has-etype
has-concept
Entity
Earth
part-of
part-of
Europe
Asia
part-of
Italy
part-of
Rome
instance-of
instance-of
instance-of
instance-of
Ontology
Location
is-a
is-a
is-a
part-of
part-of
part-of
world
continent
country
city
Language Independent Part
Language Dependent Part
synset-of
synset-of
synset-of
synset-of
synset-of
Linguistic Knowledge
(words and senses in different languages)
world, global: *involving the entire earth*
continent: *one of the large landmasses of the earth*
country, state, land: *the territory occupied by a nation*
city, metropolis, urban center: *a large and densely populated urban area*
Italy, Italian Republic, Italia: *a republic in southern Europe on the Italian Peninsula*
Rome : the capital city of Italy and also the country's largest and most populated city

Figure 2 is shown a conceptual view of the UK. The whole structure is built around the concept core which can be seen as the core which glues all the other components.

In the language independent part we codify on one side the concepts and semantic intentional relations between them (the ontological part), structured into a set of domains (the domain knowledge), and on the other side their instances, extensional relations between them and corresponding metadata (the entity part), according to their entity type. Notice that each etype corresponds to a concept in the ontological part, and therefore the entities of that kind can be seen as instances of the corresponding concept. The ontological part corresponds to what in logics is known as the T-Box, while the entity part corresponds to the A-Box. This difference is also stressed in linguistics where the former mainly correspond to common names (e.g. Country), but also verbs, adverbs and adjectives, while the latter mainly correspond to proper names (e.g. Italy).

However, differently from the classic approach, we split the conceptual part from the specific language used to express it. Each synset from a specific language (in the language dependent part) is connected either to exactly one concept (in the ontological part of the language independent knowledge) or to exactly one entity (in the entity part of the language independent knowledge). Synsets of the first kind constitute what we could call the linguistic knowledge. It groups all words with the same meaning, including common nouns, verbs, adverbs and adjectives denoting classes of entities. Synsets of the second kind form what we could call the entity vocabulary, which substantially specify entity names, i.e. proper nouns, and their informal description (gloss). Given the symmetry between the two notions, the same data structures (even if not necessarily physically the same) can be used to store both kinds of synsets. However, in this thesis we didn't consider the *entity* management and *entity vocabulary* but the *entity type*. We add them here just for the sake of completeness to give a global view. Notice that some concepts/entities may not have a lexical representation for each of the supported languages. This phenomenon is well known in linguistics as the problem of gaps between languages.

Entities and e-types are described in Chapter 6. In extreme synthesis, e-types provide constraints about the metadata (i.e., kind of attributes) associated to the entities of that kind. It is important here to underline that attributes can be used to codify implicit relations between entities (for example to codify a part-of hierarchy) and between an entity and a concept (for example to codify the instance-of relation).

Organizing knowledge by domains offers several advantages. First of all, at design time it allows concentrating on the definition of one domain at a time. Facet Analysis is the well established methodology used for decades in Library and Information Science for this purpose. The result of the analysis is a set of hierarchies, called facets, which encode the structure of the domain which is consequently easier to understand and maintain in time. Facets are typically grouped according to the DERA pattern, namely a set of fundamental categories, i.e. Domain, Entity, Relation and Attribute. In addition facets in the UK are composite objects where nodes represent concepts and arcs represent subsumption between the nodes in classification semantics. This substantially means that both concepts and entities from the UK are facet constituents. An example of how a facet can be represented in the UK is given in the picture above which is an exemplification of the Administrative Division facet in the Space domain. The backbone structure of the domain is given by the concepts (which, in this case, are the so called characteristics according to facet terminology) and the intentional relations between them. The facet is completed by the corresponding entities (their logical instances) and the extensional relations between them.

There are multiple dimensions of diversity. People speak different languages, belong to different cultures and communities, have different levels of expertise, purposes and opinions. However, even if they use different terms to name things most of the time they mean the same, or similar, concepts. Our infrastructure will help users to share as much knowledge as possible, helping them to disambiguate and standardize the terms and descriptions they use to annotate objects they handle. This will definitely facilitate communication.

3.2. Knowledge integration, evolution and maintenance

The process of synthesizing new knowledge, usually referred as the knowledge acquisition process, is very tough and expensive. Acquisition complexity varies from process to process. However, the truth is there is no easy and straightforward way. There have developed several methods for acquiring knowledge. In below we present three alternative methods for knowledge acquisition process:

a. **Scanning text**: For linguistic data acquisition the dictionary is a potential source as it focuses on the linguistic manners of terms and provides widespread lexicon-semantic information. By manually scanning printed texts of a dictionary and then manipulating the scanned data to the desired format is the least expensive way. However the output from the scanner needs to be edited at least superficially moreover some data are difficult or even impossible to obtain, documents that use graphics also a problem [60]. This process is also time-consuming.

b. **Using existing sources, corpus**: Another method is to create knowledge base from the existing resources those are already build as a part of different project manually or automated way and merge them together. The advantage of this method is the data are structured and sometimes ready to use when the source schema is nearly similar to the intended schema. However, merging data from different sources are not always straightforward as the different sources have their own way of knowledge organization and thus different schemas. Often this diversity makes the integration task quite complicated and sometimes impossible where the schemas are very different. Sources could be also very domain specifics and sometimes they just don't fit with each other. Thus, the integration requires rigorous analysis of the links between data of every source and source specific import methods.
c. **Manual :** Another way to create knowledgebase is create the content by hand from scratch which is one of the expensive methods, this also tend to be too small to be of any use in NLP applications or to lend support for theories of lexical organization [58]. Despite the limitations this method has some major advantages over the others. One of the main advantage is one can build the schema the way one deems it. It also allows creating sophisticated high quality knowledgebase that may be richer than the information one can extract automated way. WordNet is an example of this kind of acquisition. Success and build duration of this method depends on the available expert work force whose are involved in building it. Providing user-friendly smart tools can accelerate the construction time and also reduce the cognitive load. For large scale multi domain knowledgebase building smart tool that supports distributed processing and knowledge synthesize so that expert from different part of the world can contribute can be a plus point. However there doesn't exist any system that could help to manage and build large scale knowledge system, though in small scale some exists but they are for knowledge management are far from being perfect and inadequate. Knowledge management, however, is complex, and therefore should minimize the user's cognitive load so that it is usable by average users.

As underlined above creating a knowledgebase from scratch by hand is not just expensive but also huge difficult though it has many advantages. It is fundamental to take advantage of available expert linguistic and domain knowledge (which is typically encoded respectively in linguistic resources such as (Multi-)WordNet, and domain specific KOS, e.g. thesauri such as AGROVOC, NALT, AOD, and HBS) . On the other hand, automatic acquisition is faster but has many restrictions for knowledge modeling. Our strategy is to adopt both by taking the advantage of available expert linguistic and domain knowledge which is typically encoded respectively in linguistic resources such as (Multi-)WordNet, and domain specific KOS also by providing the smart easy accessible interface which can be used by different domain expert to contribute manually to synthesizing new knowledge. As it is known, a fundamental step in knowledge integration is alignment, namely finding points of overlap between the two representations and asks domain experts to validate and complete these alignments [61]. Semantic and syntactic matching techniques have been proposed as a possible solution to the semantic heterogeneity problem. In the recent years many of them has been offered. Their output is typically a set of (semantic) correspondences to be further refined manually. A good survey is represented by[63]. Therefore, bootstrapping the UK and reaching a critical mass of knowledge is done by collecting, adapting and integrating knowledge from a vast variety of different sources about different domains in different languages.

To be useful the knowledge base should be kept always up-to-date, and in the future this will be possible only collaboratively with the involvement of the users. All dynamics connected to the maintenance, diagnostics (quality checks and measures, inconsistencies, missing terms and links, redundancy, coverage etc) evolution of knowledge and the design of effective collaborative tools and user interfaces (for example which provide different view modes on the same data) must to be considered. This will be a central issue both at UK level, mainly involving community of experts, and at BK level, involving generic final users. However, this should be done by hiding sophistications to them which is still an open research issue. This should be accomplished by creating very smart user interfaces, which ideally interact only using natural language and intuitive graphical representations. Too many systems currently available pretend their users to know logics formalisms.

3.3. Knowledge Management System Description

As underlined in introduction chapter, building large scale knowledge base with maximum coverage is not possible for a single person or a small group of people without collaborative support. It is essential to provide distributed access support for distributed collaborative. On the other hand Web 2.0 has the potential to support information sharing, interoperability and collaboration [64] on the Web. Therefore we decided to build an advanced web based system to support Knowledge management. The system employed an integrated environment of a collection of sub systems namely Linguistics, Concept, Domain and Entity type management system. The system is rather unique in its kind as it combines linguistic, concept, domain and entity type with capabilities for collaborative development. It is a system that has an easy to use browsing interface for novice users, yet at the same time allows expert users to exploit the full power of a knowledge base management system. This system facilitates search and navigation and also the knowledge building and maintenance by allowing new knowledge creation, removing the obsolete and updating the existing knowledge to make it up-to-date. The system is particularly suited for the expert users (i.e., domain experts) devoted in knowledge creation process. However normal users can also get benefited by the search and navigation while minimizing cognitive load.

3.4. UK Management System

In designing the system to the universal knowledge management, we wanted to make an integrated system which would be simple for a novice to search, navigate and use yet is powerful enough to support experienced users to create and maintain the knowledge base. The system is divided into three different parts:

- ***Linguistic and Ontological Management System***: Modeling the mental representation of human linguistic knowledge i.e., language rules or grammar, for computational use rely on lexicon, therefore need well-structured lexicon with wide coverage. However, this attempt always suffers from the lack of well-structured lexicon and low coverage, which could allow for a seamless and scalable modeling. Linguistic part attempt to solve these problems by defining well-structured lexicon and providing tool to support building large scale linguistic knowledge base with maximum coverage collaboratively. In particular, linguistic part contains terms, senses and lexical relations between them in multiple languages; it provides all the natural language information about the concepts (and relationships) contained in the ontological part. The linguistic management tool provides all the functionalities that are needed to synthesize new linguistics knowledge, managing the existing and also search and navigation on them. Modeling the mental representation of human linguistic knowledge problem becomes more complicated when the question of interoperability among people having different languages is taken into account. Moreover, greater complexities are brought about by different viewpoints entailed by different cultures. To address these problems, we introduce in our system a language independent part., i.e., the ontological part (also called concept part). In particular concept part contains concept and semantic hierarchical and associative relations between concepts in a completely language independent way; The concept management tool basically integrated inside the linguistic management tool, can provides all the functionalities that are needed to manage and conceptualize the linguistic knowledge in a language independent way.
- ***Domain knowledge***: if we provide the users with a basic simple knowledge organization infrastructure as background knowledge, the quality of the users generated domain knowledge will improve and it will improve the retrieval scenario as well. For organizing domain knowledge our group proposed a logically sound, decidable, extensible and reusable generic faceted knowledge organization framework called DERA for creating domain specific knowledge base. In particular, the domain knowledge part includes concepts and relations between them arranged in facet hierarchies codifying the knowledge about a specific domain; A domain is something more than a hierarchy of concepts. It is enriched with role relations with various type of facets and each facet has facet may belong to different category (e.g., entity, relation and attribute). The domain management tool meant to manage domains and their facets.

- ***Entity part***: Entities are instances of the concept(s) associated with the *entity types* (etype) of these entities. An *entity type* describes instances of a particular class by defining the attributes that can be used to describe common properties of these instances through instantiated attributes. It provides a schema and a set of rules for the creation of a digital representation of a real world object. Entity part contains the instances of defined concepts and *etypes*; in this thesis we consider only the management of entity type.

3.5. System Architecture

We have developed the system on 3-tier architecture paradigm. Where we separate concerns by placing the views to present data into Presentation Layer, all business logic and interface to the data access are placed into Business Logic Layer, all data and code that handle data was placed into Data Layer. The web architecture allows easy access of the system. The server component is implemented in Java and provides methods for accessing the ontology content and manages the changes the users make in different clients.

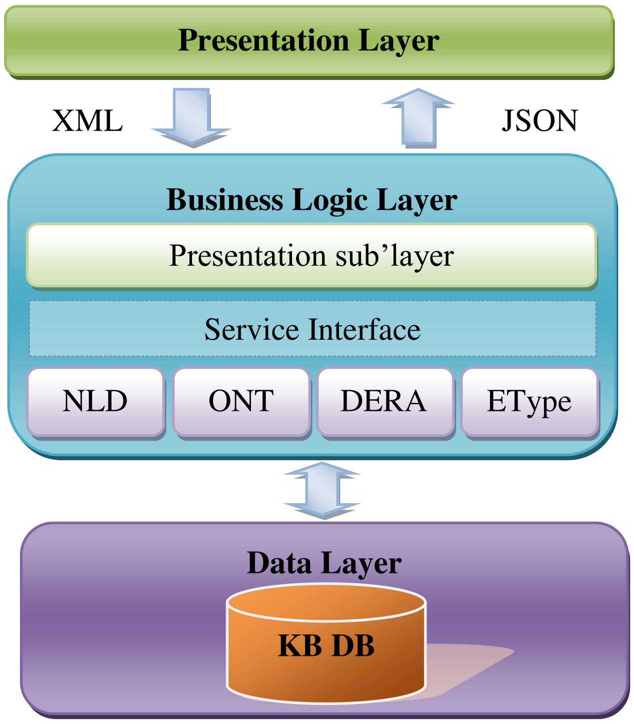


Figure 3 High level architecture of knowledge management system

Presentation layer: The top most level of the system is the presentation layer consists of user interface. Interface is provided by means of a web browser. The presentation layer is implemented using DHTML, AJAX, JavaScript, Dojo[10] (an open source modular JavaScript library designed to ease the quick development of cross-platform Ajax-based web applications), CSS (*Cascading Style Sheets*) and XML (*Extended Mark-up Language*). This makes the knowledge base available on the web and people can use it without the need to install any software also provide the user a more dynamic experience Technologies like *JavaScript, Dojo* is used at this layer so that the user can work more rapidly and thus relieve the server of the burden of user validations. This gives easy and quick form validation and error handling.

Logic layer: The second layer provides the business logic**.** In fact, this layer consists of two other sub-layer: *presentation sub-layer* and *logic- sub-layer.* The presentation sub-layer aimed at handling user requests from the client by means of a controller and forwards them to the appropriate handler. It is also re-

[10] http://dojotoolkit.org/

sponsible for generating the content to be presented in the user's browser prepared by the handler. *Logic sub-layer* provides direct access to the knowledgebase database by means of a well-defined service interfaces to unify data access developed by our group which makes application development and integration very easy. Technologies such as Java, *servlets* were used for this purpose and Maven[11] is used for the project build, reporting and documentation from our group central repository.

Data layer: In this layer data is stored and retrieved from the database. For our knowledgebase, we use an open source relational database *(PostgresSQL[12])* to store all the data about Linguistics, Concept, Domain, Entity and Entity types. The data is then passed back to the logic layer for processing and then eventually back to the user.

3.6. Evaluating the platform in real world scenarios

In order to measure the effectiveness of the proposed solution, we also conduct a series of experiments in real world scenarios, involving expert's users (for UK). This valuable feedback helped us in progressively improve the infrastructure which we are confident will be more and more used in the near future.

[11] http://maven.apache.org/

[12] http://www.postgresql.org/

Chapter 4

In this chapter we briefly describe the linguistic and concept knowledge part of the Universal Knowledge base. Linguistic part is inspired by the well-developed lexical database WordNet. To address the interoperability problem, we introduce in our system a language independent part., i.e., the concept part. In the subsequent sections we briefly describe linguistic and concept knowledge part of the *UK*.

4. Linguistic and Concept Knowledge

The most complex and unique feature of all living species and, more in particular, the most challenging manifestation of the complexities of the human mind is human language. Its study is considered as one of the most interesting research activities since ever and it grounds a proper field of scientific research called linguistics. Linguistics is mainly concerned with the form and structure of linguistic knowledge of human language and it aims at modeling the mental representation of such knowledge. Traditionally, linguistic knowledge is slice up into grammar, morphology, syntax, semantics, and the lexicon. However, the field of linguistics is not limited to grammatical theory but it articulates into a large number of subfields. A great turnover in this field of research has been brought by the revolutionary impact of computer and information technology. The advancement of these technologies has allowed accessing a large amount of information thus opening a new field of interest i.e., Computational linguistic. These new research branches beyond the study of form and structure thus moving towards the understanding of information meaning. In this context, the research outputs of the linguistic theories that have been achieved over the centuries are becoming the input of Natural Language Processing. However most of the approaches use WordNet or gazetteers [65, 66] as their background knowledge and suffering from the problem of low coverage of the available knowledge resources. The dream of building a full agreed universal knowledge schema, able to descrbe reality in an unquestionable way, has been cultivating in fields spanning from Philosophy, Library Information Science (LIS), Artificial Intelligence (AI), and more recently in the Semantic Web. This in turn requires the building of a high quality knowledge base, which codifies concepts and relationships between them. We propose a highly flexible system, which takes into account diversity of knowledge and its evolution in time.

4.1. Organization of the Linguistics Knowledge

Linguistic theories attempt to model human language rules. On the less theoretical point, the combination of computational resources and linguistic theory has evolved into a most important field of research that has initiated a number of natural language processing projects which tends to model the mental representation of human linguistic knowledge. Modeling the mental representation of human linguistic knowledge i.e., language rules or grammar, for computational use rely on lexicon, therefore need well-structured lexicon with wide coverage. However, this attempt always suffers from the lack of well-structured lexicon and low coverage, which could allow for a seamless and scalable modeling. Various attempts have been conducted and a number of projects have been developed over the last decades to fill in this gap. One example is given by WordNet hierarchical structure, one of the principal state-of-the-art projects motivated by theories of human language organization that began in 1985. Linguistic part of the Universal Knowledge Base is inspired by the well-developed lexical database WordNet. Atomic building elements in WordNet are words and synsets. Synsets are sets of words, which express the same meaning in a given context. Words' senses are represented as synsets. WordNet system entails two kinds of relations: lexical and semantic. Semantic relations are defined among synsets (i.e., between concepts in the system), whereas lexical relations are defined among words.

While acknowledging merits and the well-articulated database contained in the WordNet, the starting point for our proposed system is a preliminary and clear distinction between Linguistic knowledge and the Concept that is not acknowledged in the WordNet structure. The goal of the Linguistic knowledge is to provide information on natural language, whereas the Concept Knowledge part is responsible for maintaining *concepts* of the *synsets* and their *relations* in a language independent manner. The objective behind this separation is to make easier the maintenance and to map different languages. The linguistic knowledge part is language dependent, which is composed of several languages (e.g. English , Italian), divided into four lexical categories: nouns (n), adjectives (a) verbs (v), and adverbs (r). Similarly to WordNet, within our system a set of words organized around logical grouping is called synset. Each synset consists of a list of synonymous words i.e. groups of words with the same meaning, and describes the senses of each word with natural language descriptions (glosses). Each synset is attached with the information about the part of speech it represents, i.e. noun, verb, adjective and adverb. One word may have one or more senses thus containing one or more synsets and in more than one part of speech. One synset may contain one or more words having the same meaning in a given context. The words in a synset are logically grouped such that they are interchangeable in some context, for example the word "java" has three senses:

1. "Java" -- (an island in Indonesia south of Borneo; one of the world's most densely populated regions)
2. "Coffee, Java" -- (a beverage consisting of an infusion of ground coffee beans)
3. "Java" -- (a simple platform-independent object-oriented programming language used for writing applets that are downloaded from the World Wide Web by a client and run on the client's machine)

We encode these ideas as follows:

4.1.1 Word

Word represents the morphological root of a word. The exceptional form represents a derived form of the morphological root of a word. For example, the form "better" is a derived form (also called exceptional forms) of the words "well" and "good".

Definition 1: Word

We formally define the word as follows: Let be S the set of strings in natural language with a length greater than zero. A word w is defined as a triple <N^W, L^W , E^W >, where N^W is the lemma of the word, L^W is the language to which the word w belongs to, and E^W is the set of Exceptional forms if any, such that:

(a) $N^W \in$ S, i.e. N^W is a string;
(b) $L^W \in$ {English, Italian}, i.e. L^W specifies the language of the word;
(c) $E^W \subseteq N^W$, i.e. E^W is a set of exceptional forms where $E^W \in$ S, i.e. E^W is a string;

4.1.2 Synset

Synset represents the set of words with same meaning in a language, for instance, when the words *test, trial* and *run* used to described *the act of testing something* , "in the experimental runs the amount of carbon was measured separately"; A particular word can denote different meanings in different contexts (polysemy), for example the word *run* in another context - "take a run into town" , in this context the word *run* used to described a particular concept *a short trip;* and it's also likely for a single word to be used as different parts of speech (noun, verb, adjective etc.) For example, *run* as verb, when it used to refer to carry out a process or program, as on a computer or a machine; "Run the dishwasher"; "Run a new program on the Mac" whereas, *run* as noun "she broke the record for the half-mile run". So, words set belongs to a synset also depend upon the part of speech associated with the sense the synset represents.

Therefore, we can define the synset as a set of synonymous word that in the same context assume the same meaning, for instance, in the first example "test, trial, run", used to describe a particular concept thus belong to a synset.

Definition 2: Synset

We can formally define the Synset as follows: A Synset s is defined as a triple $<W^S, G^S, P^S>$, where W^S the non zero set of words in the same context assume the same meaning in a language. G^S is the natural language description of the synset, and P^S is the parts of speech of the of the synset, lang(w_i) is the language for word w_i, such that:

(a) $W^S \in \{W\}$ s.t. $\forall\ w_i, w_j \in W^S$ where $i \neq j$ and $lang(w_i) = lang(w_j)$

(b) G^S is the natural language description of the synset in a language specified in W;

(c) $P^S \in$ {Noun, Adjective, Verb , Adverb }, i.e. P^S is the Part of speech of the synset;

4.2. Organization of the Conceptual Knowledge

Modeling the mental representation of human linguistic knowledge problem becomes more complicated when the question of interoperability among people having different languages is taken into account. Moreover, greater complexities are brought about by different viewpoints entailed by different cultures. To address these problems, we introduce in our system a language independent part., i.e., the concept part. By concept we mean the language independent representation of a class of real world objects. The Concept Knowledge part (also called ontological part) of our system maintains *concepts* and *relations* between *concepts*. This will form the *upper ontology or foundation ontology.* Upper ontology is ontology where concepts are described very generically. The concepts descriptions in upper ontology are the same across all the domains. It delineates top-level classes, such as, physical objects, activities, mereological and topological relations from which more specific classes and relations can be defined. IEEE defined upper ontology as, "An upper ontology is limited to concepts that are meta, generic, abstract and philosophical, and therefore are general enough to address (at a high level) a broad range of domain areas. Concepts specific to given domains will not be included; however, this standard will provide a structure and a set of general concepts upon which domain ontologies (e.g. medical, financial, engineering, etc.) could be constructed" [67]. Since the concept descriptions are generic in upper ontology, it will allow the semantic interoperability between a large numbers of ontologies accessible "under" this.

4.2.1 Concept

It is the language independent representation of a synset. Each concept has a unique identifier that uniquely describes it so that the problem of polysemy and synonymy of natural language does not arise. For example, "coffee", used to describe a particular concept. Synsets with the same sense from the different languages are grouped together under a single concept to give a language independent representation. Notice that, in one specific language one synset can have one and only one concept. Conversely, a concept can link to many synsets with the same sense from different languages thus making sure that it links just only one synset in a language. For example, in the English linguistic part, the word flower has three noun senses /synset, one of them is "a plant cultivated for its blooms or blossoms". On the other side the Italian linguistic part one of the noun sense/synsets of fiore is "*in una pianta, la parte che serve alla riproduzione; in genere è la più appariscente, colorata o profumata*". The meanings of these two senses/synsets are same therefore we can attach these two synsets under a single concept represented by a unique identifier. The natural language name of the concept would be the word's lemma N^W of the corresponding synset in the given language L^W.

Definition 3 : Concept

We can now proceed to the definition of concept. A concept c is defined as follows:
A concept C is a pair $<id , S^C>$, where:

(a) id is the unique identifier of the concept;

(b) $S^C = \{s_i\}$, is the set of synsets s.t. $\forall\ s_i, s_j \in S^C$ where $i \neq j$ and
   a. $lang(s_i) \neq lang(s_j)$
   b. $lang(s_i) = lang(w_k)\ \ \forall_k : w_k \in w_i^s$

4.2.2 Concept hierarchy

Objects managed by concept knowledge are assumed to be heterogeneous and indexed in concepts hierarchy, also called ontology. Concepts hierarchy is manually made and provided by expert users. A concept hierarchy can be associative or purely hierarchical or both structure of nodes. Users build concept hierarchy node by node, starting from the root node. Each node in the classification can have an arbitrary number of children. Concept hierarchies are dynamic structures.

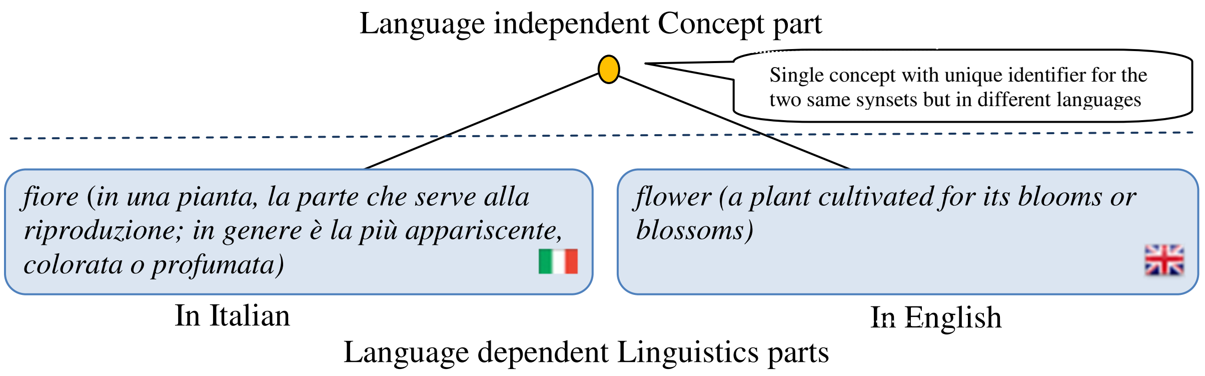

Language independent Concept part
Single concept with unique identifier for the two same synsets but in different languages
fiore (in una pianta, la parte che serve alla riproduzione; in genere è la più appariscente, colorata o profumata)
flower (a plant cultivated for its blooms or blossoms)
In Italian
In English
Language dependent Linguistics parts

Figure 4 Language independent concept representation of the synset of Italian word fiore and the English word flower

Relation: Relations play an important role for effective knowledge discovery. As we have mentioned before, the concept part of our system maintains concepts and the relations between concepts. To put it simply, relations are links between two concepts. These relations can be abstractions or characteristic of two concepts. Each relational builds a semantic link between two concepts. Relations are binary. As a matter of fact, prepositions and transitive verbs constitute the primary type of relations (e.g., car *is a* motor vehicle). We classify the relation between concepts are in two categories:

- **Hierarchical relations**: Transitive relations are considered as hierarchical relations. Hyponym, instance hyponym, part meronym, and their inverses, are transitive relations.
- **Associative relations**: Except transitive, any arbitrary relations are considered as a associative relation

Definition 3: Relation

We can define the hierarchical relation R_H as follows: $R_H \in$ {is-a , part-of }, i.e., R_H is the hyponym, part meronym and the associative relation as follows $R_A : \{ r \mid r \in C \times C \ \& \ r \notin R_H \}$
Thus, we can define the set of all the relations R as follows: $R = R_H \cup R_A$

A relation between concepts can be represented as a triple $< S^C ,R^C , T^C>$ where S^C is the source concept of the relation , T^C is the target concept of the relation, and R^C is the relation that holds between source and the target, where $R^C \in R$:

Relation kind	Relation	Concept
Hierarchical	Is-a	Hyponym, subordinate, subordinate word
	Part Meronym	Portion , component part, component, constitute
Associative	Antonym	Antonym , opposite word, opposite
	Related	Related, Related to
	Verb	Verb
	Original	Original
	Similar	Similar
	Substance Meronym	Substance component part

	Member	Member
	Cause	Consequence, effect, outcome, result, event, issue, upshot
	Deduction	Deduction, entailment, implication
	Group	Group, aggroup
	Consider	Consider, Take, deal, look at
	Property	Property, Attribute, dimension

Table 1 List of possible relations maintained by the concept part of our system thus classifying it according to their hierarchical and associative feature.

Atomic building elements in the Linguistic part of our system are words and synsets and atomic building elements in Concept part are *concepts* and *relations*. Below is an example of two words sharing the same synset (sense # 2 of the noun "java" and sense # 1 of the noun "coffee"), which contains two words ("java" and "coffee"):

Three noun senses of java in English:

1. Java -- (an island in Indonesia south of Borneo; one of the world's most densely populated regions)
2. coffee, java -- (a beverage consisting of an infusion of ground coffee beans; "he ordered a cup of coffee")
3. Java -- (a simple platform-independent object-oriented programming language used for writing applets that are downloaded from the World Wide Web by a client and run on the client's machine)

One Italian sense for "coffé":

1. Caffé – (il caffe e un bevanda ottenuta con i semi torretti e macinati della pianta del caffé)

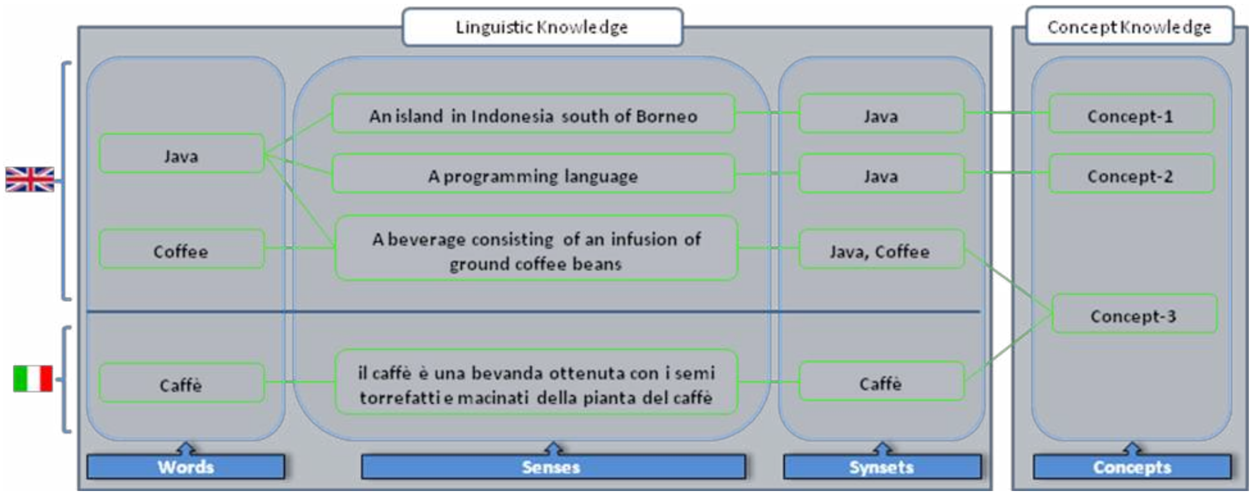

Linguistic Knowledge
Concept Knowledge
Java
Coffee
Caffè
An island in Indonesia south of Borneo
A programming language
A beverage consisting of an infusion of ground coffee beans
il caffè è una bevanda ottenuta con i semi torrefatti e macinati della pianta del caffè
Java
Java
Java, Coffee
Caffè
Concept-1
Concept-2
Concept-3
Words
Senses
Synsets
Concepts

Figure 5 Conceptual model of Linguistic and Concept Knowledge

Figure 1 depicts the conceptual model of Linguistic and Concept Knowledge in terms of Words, Senses, Synsets and Concepts and their relations.

4.3. Conclusion

In this chapter we have provided a brief description of the basic building blocks of linguistic knowledge and concept knowledge. We introduce the concept part as a solution to overcome the interoperability problem having different languages. We have shown how senses/synsets from lexical part are linked with language independent concepts. In chapter 8 we will present our proposed system that can be used to search, browse and manage the linguistic knowledge and concept knowledge collaboratively to overcome the coverage limitation.

Chapter 5

5. Domain Knowledge

The amount of knowledge and information in today's society is growing enormously. According to the definitions provided by Russell Ackoff [68] data is symbols, information is data that has been given meaning by way of relational connection and knowledge is the appropriate collection of information of data and information such that it's intent is to be useful. The knowledge is reproducible and sharable. The reproducible nature of knowledge makes it very dynamic. Something we know as true today, tomorrow appear as false. It keeps on changing every moment of time, in the same time growing as well. To get the proper benefit of this ever growing knowledge it becomes fundamental to develop efficient and easy to use systems to access and navigate information [9] [14] [31][69]. Knowledge diversity because of interoperability among people and cultural viewpoints managing and organizing in an efficient and useful way is not a simple task.

In traditional library system, the documents are organized in the shelf in a manner that they are easily retrieval. In organizing the library materials into the library shelf is done by the librarians. For easy retrieval of documents in a traditional library, there are in general several access points made available to search and retrieve the documents from the shelves. And majority of time, the library users become satisfied as they retrieve the required information they look for. However, this is not the case in Web environment. In most of the cases users are dissatisfied as the retrieved results sets lacks precision, as well as recall. The reasons behind these two principal problems could be traced as the lacks in organizing and describing the Web documents. In traditional library environment, the knowledge organization systems, such as, term lists (e.g., Dictionary, Glossary, etc.), subject categorization tools (e.g., Dewey Decimal Classification, Universal Decimal Classification, Colon Classification, etc.) and relationships lists (e.g., thesauri, etc.) are used for organizing the knowledge. These traditional knowledge organization systems are found efficient and effective in organizing the knowledge in the shelves for easy retrieval. However in context of Web, there are many such initiatives, including Yahoo! Directory, DMOZ, etc. are in place in organizing the Web documents and manual intervention is involved in these. But it is understandable that it is impossible for any manual effort to organize and describe the entire World Wide Web documents. It is mainly the problem of scalability. So, in order to reduce the human efforts and to solve the problem of relevant retrieval, the idea of Web 2.0 is in place.

It is appropriate to state that there is no finality of knowledge, as it is ever growing and changing. Therefore it is not that we design one classification system once for all; rather it is a continuous process. In response to these needs, in the last years classifications and in particular faceted systems are becoming very popular[9][69]. They tend to naturally group objects with respect to their distinctive properties, i.e. their attributes, categorizing them in different hierarchies; each hierarchy corresponds to a different facet[70]. Among various suggestions in improving the quality of the users generated catalogues one of them is to provide the users with a background knowledge organization infrastructure (i.e., classification ontology) to classify and to describe the documents. In order to be intuitive for the user in this process, facets are typically taxonomic structures, i.e. IS-A hierarchies[70]. Unfortunately, these kinds of semantically "pure" classifications are difficult to construct and maintain [71] because they are data-dependant and the construction process is clearly subjective. In fact, in [71] it is argued that human efforts to represent knowledge in classification systems obey to the Second Law of Thermodynamics (in a closed system, entropy or the measure of disorder, always increases). So, it is simply not worth trying to construct an a-priori comprehensive information organizational structure. This task cannot be done only by automatic systems, first because generally users do not trust automatic tools and also because the summary is always up to date and never just a "snapshot" of time[71]. It has been demonstrated that these systems are inadequate to support users because they do not enhance their browsing experience [71]. At each user action the system should react in order to consequently and transparently update the knowledge base. By evolving mentioned techniques, our system aim to support heterogeneity in data and to naturally adapt to the increasing data complexity and user experience.

Our understanding is, if we provide the users with a basic simple knowledge organization infrastructure as background knowledge, the quality of the users generated classification will improve and it will improve the retrieval scenario as well. In this context we advocate here for a less complex ontology to be provided to the end users, which we call as faceted lightweight ontology [72]. We call it faceted lightweight ontology, because it is developed following the noble "faceted approach" as discussed below. The goal of this chapter is to introduce a management system of a logically sound, semantic knowledge organization framework DERA designed following the faceted approach and epistemological [73] theories in order to facilitate large scale knowledge integration.

5.1. DERA

DERA is logically sound, decidable, extensible and reusable generic faceted knowledge organization framework for creating domain specific background knowledge base. The data model allows representing concepts, and named entities (objects), relation between objects and the properties of objects. DERA framework consists of 4-tuples, such as, domain, entity, relation, and attributes which offer a powerful knowledge-structuring mechanism, whose meaning was intended to reflect important cognitive assumptions. The designed independent nature of DERA framework makes it flexible to fit any domain in it. Our conviction is that ontology framework should be designed independent of any particular domain. It is because we cannot develop ontologies just to meet everyone's perception about a domain. In the framework, each domain consists of a set of entities, relations, and the attributes providing the contents of attention and qualities, which are interestingly significant in context of a domain in hand. In addition, each domain will have an identifier and name.

$D = < \{E\}, \{R\}, \{A\}>$
Where, D = Domain
E = Entity
R = Relation
A = Attribute

The details descriptions for each of these elements are provided in the following sections.

5.1.1 Domain

In classification, domain is a field of study or interest whose boundaries are defined considering the interest and purpose of intended audience/user groups. It is an elementary category representing a branch of knowledge dealing with specific kinds of subjects considered from specific points of view. It includes both the conventional fields of study (i.e., branches of learning) as well as day-to-day domains of interest. For example, the conventional domains are, like, history, philosophy, physics, mathematics, biology, computer science, etc. and the day-to-day domains of interests are, like, travel, sports, recipes, weather, etc. We are conscious that librarians since provides a theoretical and historicized background in information categorization, especially in the work of Ranganathan[74] and Battacharyya[] [16].

The division of universe of knowledge into several knowledge areas offers comparatively better solution regard to the problems of knowledge organization and retrieval, and also serves better to the knowledge users. The exposition of this method is found in the Book II of De Dignitate (expanded version of the Advancement) by a famous English lawyer, essayist, historian, intellectual reformer, philosopher, and champion of modern science Francis Bacon [28]. He deeply examined the prevailing state of knowledge and means of its progress. Any knowledge area consists of a set of individuals, relevant concepts, and properties of individuals. These are three fundamental components of any domain discussed in details in the succeeding sections. These are fundamental as using these three basic components we can express any domain knowledge.

5.1.2 Entity

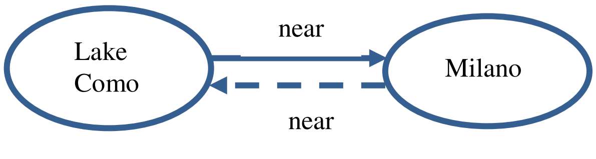
WordNet defines entity as “which is perceived or known or inferred to have its own distinct existence living or nonliving” [46]. While, Wikipedia defines entity as, “it is something that has a distinct, separate existence, though it need not be a material existence. According to Prof. G. Bhattacharyya [75], entity is “an elementary category that includes manifestations having perceptual correlates or only conceptual existence, …”. The former two definitions are defined linguistically, whereas, the latter one is defined in context of subject classification.

In our case, an entity is an elementary category consisting of core concepts and the entities (objects), having either perceptual correlates or only conceptual existence in a domain in context. According to this definition, category “entity” consists of two parts: “entity class” and the “entities” or “objects”. *Entity class(e)* holds the core concepts representing a domain in context, while “entity” or “object” represent the real world entities or objects. Note that, from now onwards wherever we use the term “entity” or “object”, we mean the second one. In the above, by mentioning “*core*”, we mean, the candidate terms (ideas) representing a domain in hand, and does not includes the terms representing properties of entities under a domain in context. For example, in context of a domain, "mysticism" (“*A religion based on mystical communion with an ultimate reality”*), the concepts like *hindu, jain, buddhistic, mahayana, judaic, christain, islamic*, or concepts, like, *god, nature, plant, time, space*, etc. are the *core* concepts use to represent the domain, while the concepts like, *nomenclature, symbolism, vision, tradition, manifestation, telepathy, transformation, omen, divination, “magic and witchcraft”*, etc. are not. The latter sets of concepts are to be considered as properties in context of domain “mysticism”. In linking to natural language, the core “concepts” can be thought of as “common nouns”, while the “named entities” (object) can be thought of as proper nouns, acronyms, nickname, or abbreviation

5.1.3 Relation

According to Carol Stockdale and Carol Possin [76], the relation can be between oneself and the environment or between two or more objects outside of oneself. In a simplest way it can be said that, a relation is a link between two entities (objects). Each relation will have a source entity (object) belonging to a class and target entity (object) belonging to a class. It is possible that the source and target entities for a relation belong to the same class or they are from two different classes. However, the relations can also be established between two classes, or between an entity (object) and a class. In the following paragraphs we discussed three such different situations when relations can be defined. These are:

- **Relation in general (E – E relation)**: Consider the following fact. “Lake Como is near Milano”. Here, “Lake Como” and “Milano” are two locations, where Lake Como is a kind of Lake, while, Milano is a kind of Administrative division. Now, according to the stated fact the former one is geographically is “near” (near in time or place or relationship) to the latter one. So, here the relation “near” links the objects “Lake Como” with “Milano”. Notice that, the fact “Milano” is near “Lake Como” is also true for the relation “near”. The above facts are represented graphically as follows. Here, source and target classes are “Lake” and “Administrative body” respectively for the relation “near” and vis-à-vis.

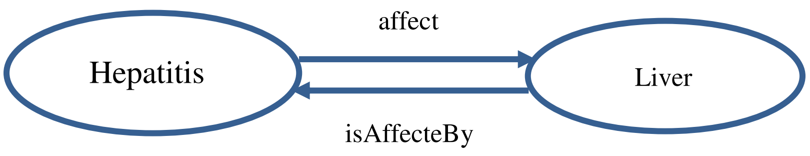


- **When e as an entity (E – e relation)**: There are some cases when we do not have the target entity (E) or we do not exactly know the name of the target entity for a relation r. In these cases, our system allows a relation to link between an entity (E) and an “entity class” or concept (e). Note that, this is not the case of “class and its instance” scenario, where, an entity (E) is an instance of a class (e). Let consider another example, “Aosta Valley, located in the Alps of Northwestern Italy, surrounded by some of the highest mountains in Europe is a mountain lover's dream." In this example "Aosta Valley" which is a valley and it is surrounded by some mountains (i.e., a mountain range or a chain of mountains), but we do not know the name of the mountain range. In this case, our system allows storing the partial information, instead of just forcing a user to store some meaningless or wrong information. By mentioning meaningless or wrong information, we mean that, the user could give any random name to this entity, which is an alternative solution in this situation but not a correct solution. So, our system allows the user to link an entity with the entity class. The above fact can be represented like the following.

< Aosta_Valley surroundedBy MountainRange>

Here, "Aosta Valley" is a kind of "Valley", "Mountain range" is an entity class, and "surroundedBy" is the relation (r) linking these two

- **When relation between e and e (e – e relation)**: Like the above scenario, i.e, relation between E and e, we may also have the situation, when we don't have any named entities, instead, we will only have the classes. For example, consider a domain, "Medicine". Body and its organs, such as, Lower extremity (toe, foot, leg, etc.), Upper extremity (shoulder, axial, hand, etc.), Digestive system (mouth, pharynx, intestine, etc.), Circulator system (heart, artery, capillary, etc.), Respiratory system (nose, larynx, bronchi, etc.), etc. and the classes like Obstetrics, Disease (fever, tuberculosis, inset, deficiency, etc.), Physical fitness, etc. represent the domain "Medicine". We do not have the named entities as instances of those classes. In this type of situation, our system allows to link between two classes or sub-classes. Therefore, from the above discussion, it can be said that there are some cases, when we will have both source and target entities as e, not E'. For example, consider the fact "hepatitis affects the liver". After analyzing the fact, we can say that, liver, a body part of human being gets affected by hepatitis, a kind of infectious disease. This fact can be represented as shown below. Notice that, here relation "isAffectedBy" is an inverse relation of relation "affect".



5.1.4 Attribute

An elementary category includes manifestations referring the characteristics or properties of entities (objects). These include qualitative or quantitative or descriptive characteristics of the entities. For example, depth, area (such as, catchment area, surface area), water volume, etc. are the attributes for the entity like, Lake Garda. While, the attributes like, latitude, longitude, elevation, climbing history, etc. are attributes for the entity like, Mount Everest. Based upon following observation we derive two kinds of attributes, such as, datatype attribute (A') and descriptive attribute (e') discussed in the succeeding subsections

- datatype attribute: Datatype attributes are mainly the qualitative, and/ or quantitative. For example, red car, i.e., redness of a red car, or we can further say that the redness all red cars share. Here, being red is a datatype attribute of a car. Consider another example, deep lakes, here, deepness is a datatype attribute cab be shared by all deep lakes. On the other hand we could also quantify the exact depth of the lake.
- descriptive attribute: descriptive attribute (e') describes distinctive characteristics or essential features the of an entitie. The values of it could be atomic concepts or a sequence of strings. For example, consider a world famous monument, "Taj Mahal", located in India. A possible attribute of this could be "history", as for this kind of historical entities, "history" is one of the important characteristics. It is worthwhile underlining that in natural language the descriptive attributes and their value(s) correspond to the common nouns, and/ or noun phrases..

5.2. Facet Hierarchy

Facet is a hierarchy of homogenous terms, where each term in the hierarchy denotes a primitive atomic concept, i.e., a primitive class of real world objects. Facets have the following two key properties:

- They are organized as a set of independent domains that are completely modular and can be developed independently. A facet hierarchy generally is an IS-A hierarchy in which attribute values corresponds to leaf node labels. At each level these values are grouped in a more general concept, i.e. a set of real world objects, represented by the parent node label usually by aggregation (IS-A) or more rarely by composition (PART-OF). Therefore, the concept expressed by the parent node is as expected more general than the concepts expressed by its children. These hierarchies are usually manually made using a bottom-up approach by grouping step by step values in more general chunks
- For each domain, facets are grouped into specific elementary categories according to the DERA categories.

Figure 6 : Space domain (partial view) shows part of "Space" domain modeled according to the theory of DERA. As an example one group of entity class is "Body of water". Water body is any significant accumulation of water, usually covering the Earth and other planet. We consider here only water bodies on the Earth. The term body of water generally refers to large accumulations of water, such as oceans, seas, and lakes. It also includes smaller pools of water such as ponds, puddles or wetlands, Rivers, streams, canals, etc. It also shows some relations and attributes. However the example (Figure 6) shows only a partial view of the Space domain.

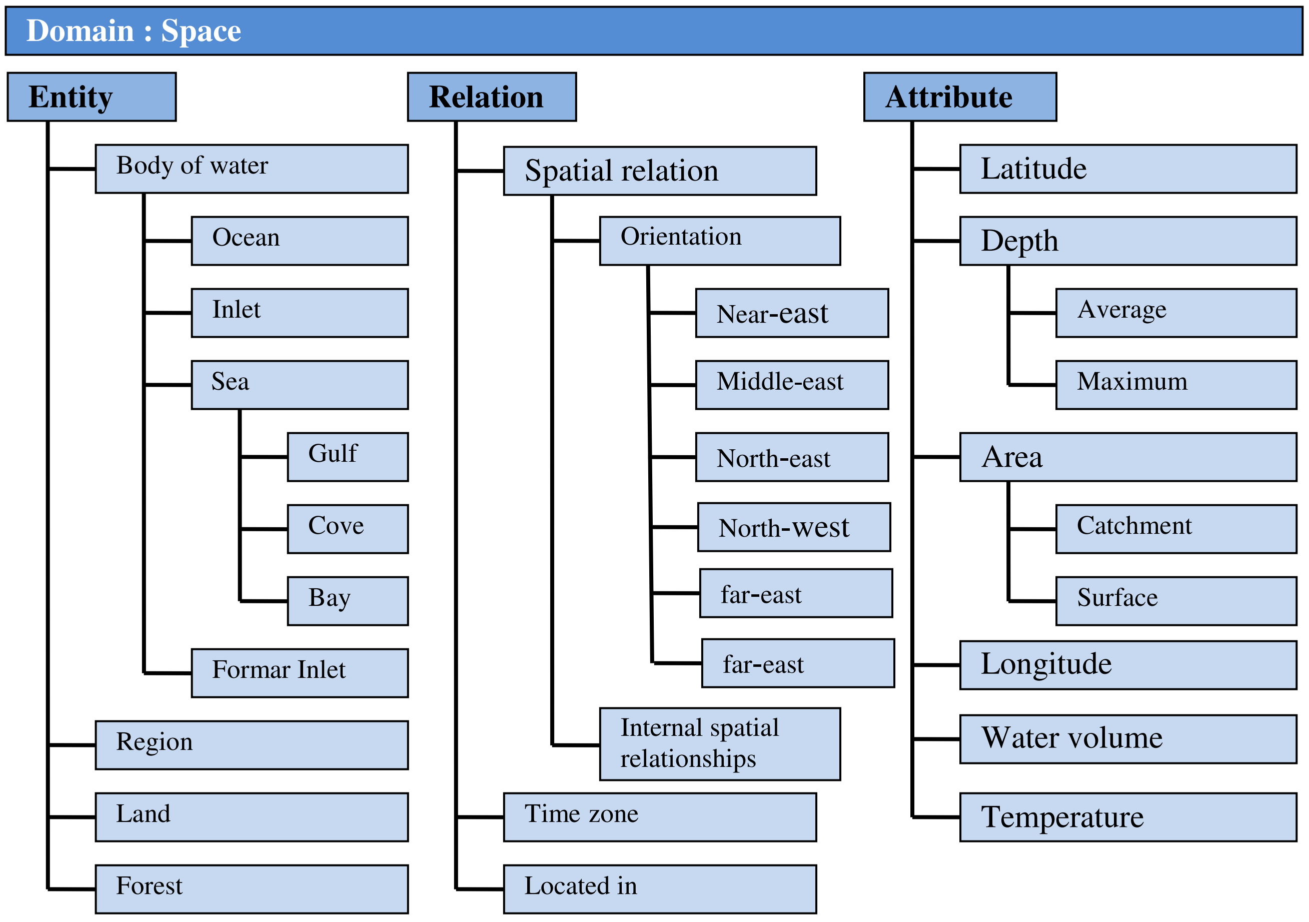


Figure 6 : Space domain (partial view)

5.3. Conclusion

In this chapter we have introduced and demonstrated the logically sound, semantically enriched faceted knowledge organization framework DERA to facilitate large scale knowledge integration designed following the faceted approach and epistemological theories. In the chapter 9 we will demonstrate the implementation of a web system designed for developing and maintaining the domain knowledge in DERA structure.

Chapter 6

6. Entity Type

This chapter describes a model for attribute definitions, entity types, entities and for other related objects.

6.1. Fundamental Basis of Entity and Etype

The fundamentals for the definition of entity types and entities are defined in terms of some meta-properties: essence, rigidity, identity, unity, and dependency base on the state-of-the-art work in the field [77,78,79,80] . Meta-properties are properties that characterize the properties used in ontological modeling.

6.2. Primitive notion of Entity Type

We already have introduced the notion of entity in our Domain Knowledge chapter. Entities are generally the objects of interest. It represents the real world entities or objects. People can be entities; rivers can be entities. Both people and lands can be entities. In general we can say that an entity type or Etype provides a template to define Entities based on the primitive notions of *concept* and *system data types.*

- A ***Concept*** is defined as a set of individuals (also called the extension of the concept). Each concept has a unique identifier and a natural language name that describes the concept.
- ***System data types*** are data types supported by state of the art data base systems and programming languages. They include (but are not limited to): integers, strings, dates, floats, Booleans. It also includes the URL data type that is defined on the range of HTTP URLs. Table 1 shows the list of system data types.

6.3. Overall Aspects of Etype Model

We assume that a large portion of the data described in the model will be generated by the user in the bottom-up fashion. Therefore, the model should not be (too) expressive or (too) restrictive. While giving a relative freedom to the user, the model should take into account system-oriented requirements such as the availability of constructs for basic operations. The data structures, necessary to represent the model, as well as model-enabled operations should not be (too) space and time demanding and should optimally guarantee real-time performance.

6.3.1 Data Types

A data type defines all the possible values for that type, its semantics and the set of operations that are allowed on those values; it defines the upper bound on the domain of possible values. The platform comes with a set of system data types that provide the basic blocks around which an *Etype* can be defined. Each data type includes a special value "null" that indicates an uncertain state, literally it means either "not applicable" or "not known", note how it is different from the zero value since it does not encode any semantic. In the following table we provide the list of the system data types supported by the platform

Name	Description	Value Example
Boolean	Allows the assignments of the "true" and "false" values.	"True" or "False"
Integer	Allows the assignment of a value in the domain of the integer numbers.	"100"

Float	Allows the assignment of a value in the domain of the real numbers.	“3.14159265”
String	Allows the assignment of a string value encoded as a simple sequence of characters, where the semantic is not extracted from its content but is codified in the business logic of the application. It could be a name without any semantic attached, for example the name of a star, a passport number, an ISBN number, the hash code of a password, a URI etc.	ISBN: “88-515-2159”

Besides these state of the art data base systems and programming languages data types there also some other predefined custom data type , as an example, *Duration* data type. The *Duration* data type allows the assignment of a value which codifies a duration in time. It is encoded as two values that specify the minimum (dtm) and the maximum (dtM) amount of time using standard duration representation. Two values are used in order to encode the variation of a duration (E.g. A meeting may last about one hour, more or less 10 minutes). Note that a precise (with a precision of 1ms) amount of time can be encoded when dtm = dtM. Another data type is *URL***,** it allows the assignment of a value which pattern follows the URL schema[13]. For example a SURL or a WURL. One interesting data type is *Entity***,** in our model entity is also used as a data type. The entity data type allows the assignment of a value in the domain of all the possible entities (See Section 6.3.4) of any *Entity Type* (See Section 6.3.5) , for example “Fausto Giunchiglia”, as an entity of type “Person”.

Here we present a subset of data type that support by the system. Note that each data type is used within the definition of an attribute (See section 6.3.3) to define its co domain. Each attribute may allow the instantiation of multiple values, in this case each attribute value must be picked from the defined domain of the data type, for the entity data type there is a special treatment, in fact each value can be instantiated as a set of entities.

6.3.2 Domain Restrictions

A data type defines all the possible values and their operations; the it is also possible to define a specific sub set of values redefining the domain of the data type, putting a set of restriction on the domain of possible values. The domain of an attribute definition is defined on the range of system data types or on the range of entities of a particular etype. Note that entities belonging to more specific etypes are also valid members of the domain. For instance, attribute definition PLACE can be defined on the range of entities of etype LOCATION and on the range of entities of more specific etypes (e.g., REGION). In the following table we provide one example domain restrictions:

Name	Description	Example

Enumeration<N>	Allows the assignment of a value taken from a given list of possible values. The enumeration defines an exact listing of all of its elements without repetitions which value is of any datatype defined in section 6.3.1. N (from <NAME>) is the name of a datatype of the enumeration elements. It also allows defining a ***default value*** from the list of elements of the enumeration.	Enumeration<Integer>:{1,**2**,3, 4}
		Enumeration<Entity>:{***Fausto Giunchiglia***, Ilya Zaihrayeu}

Note that an enumeration can also be used to give a quantitative value for those qualities for which there is no metric space in the qualitative value of the attribute, for example the ranking of an entity (e.g. the five star ranks can be defined as an enumeration of integers {1, 2, 3, 4, 5}.

[13] See http://www.ietf.org/rfc/rfc1738.txt.

6.3.3 Attribute

An attribute is defined as the whole composed by its definition and its instance, in the following sections we first report the description of an attribute definition with the related extensions and implications and we further analyze the different cases of attribute instances.

6.3.3.1.Attribute Definition

An attribute definition (*AD*) provides a template for the creation of instance of a given data type.

<u>Definition 1: Attribute Definition</u>

An Attribute Definition is a tuple $AD = < C^{AD}, D^{AD}, isSet^{AD} >$, where

a. C^{AD} is the concept of the attribute definition
b. D^{AD} is the domain of the attribute definition and
c. $isSet^{AD}$ is a boolean value which indicates whether *AD* can be instantiated to a set of values (in this case $isSet^{AD} = true$) or to a single value (in this case $isSet^{AD} = false$).

In this definition C^{AD} is a concept that identifies the attribute name, D^{AD} defines the domain of the data type as a set of domains of values that provide the boundaries (by a set of data types and/or the domain restrictions) the attribute instances are limited to. In other words D^{AD} is a list of data types plus eventual domain restrictions on each single data type. Finally *isSet* is a boolean value that indicates whether the attribute can be instantiated to a set of values or not. As a constraint within an Etype different attribute definitions cannot share the same name denoted by the concept C^{AD} of the attribute definition, note that the concept is used to identify attributes among different Etypes, in fact an attribute defined inside an Etype could be redefined inside another Etype using the same concept. This allows the system to compute the domain of an attribute as the set of Etypes composed by the Etypes of the attributes that share the same concept. Note also that an attribute definition allows providing a set of domains of values (D^{AD}), in other words polymorphism is allowed.

An attribute definition is called *relational attribute definition* when the domain of its data type is defined on sets of entities. In this case the attribute definition encodes a relation between two entities or between sets of entities

A context is information that is relevant to an entity with respect to its relation with another entity. A *context attribute definition* applies therefore to a *relational attribute definition* and it's defined as an attribute definition attached to another (relational) attribute definition.

It is also possible to define attribute that provide a schema for the description of logically indivisible pieces of information. This kind of attribute definition is called *structured attribute definitions*. Note that it allows for recursive structured attribute definitions, e.g. a structured attribute defined within another structured attribute definition.

6.3.3.2.Instantiated Attribute

<u>Definition 2: Instantiated Attribute</u>

An Instantiated Attribute is defined as the tuple $A = <AD, V, T>$, where

a. *AD* is an attribute definition;
b. *V* is a set of elements from the domain of the attribute definition ($V \subseteq D^{AD}$) if $isSet^{AD} = true$, and it is an element from the domain of the attribute definition ($V \in D^{AD}$) if $isSet^{AD} = false$; and
c. *T* is a time interval during which the assignment of *V* to *AD* is valid.

Here *AD* is an attribute definition or a structured attribute definition and *V* is the set of values composed by single value according to the *isSet* value of *AD* belonging to the domain of the data types D^{AD} of *AD*. Note that in the case of a structured attribute definition *STD*, D^{AD} is not defined at its level but it is defined for each of the *AD* in *STD*. Notice that, an instantiated attributes are defined for a certain object which defines the context for the temporal validity of instantiated attributes. For instance, attribute isFather is instantiated to false from the moment when a male person is born up to the moment when his first child is born.

6.3.4 Entity Types

Intuitively, an entity type describes instances of a particular class by defining the attributes that can be used to describe common properties of these instances through instantiated attributes. It provides a schema and a set of rules for the creation of a (digital) representation of a real world object (e.g. a person, a building, an event, a document etc.).

Definition 3: Entity Type (Etype)

An Entity Type (Etype) is a tuple $ET = <C^{ET}, AD^{ET}>$, where

a. C^{ET} is the concept of the etype, and
b. $AD^{ET} = \{AD\}$ is a non-empty set of attribute definitions.

Here C^{ET} is a concept that denotes the name of the Etype and AD^{ET} is a non empty set {AD} of attribute definitions including structured attribute definitions and context attribute definitions.

6.3.5 Entity

Entities are instances of the concept(s) associated with the etype(s) of these entities. For example, instances of class PERSON can be described with attributes like NAME, BIRTHDATE, GENDER, and so on. Thus, there can be etype Person with the above-mentioned attribute definitions and entities of that etype with instantiated attributes.

Definition 4: Entity

An Entity is a tuple $E = <ET^E, AE>$, where

a. $ET^E = \{ET\}$ is a non-empty set of entity types, and
b. $A^E = \{A\}$ is the set of instantiated attributes, such that for any attribute definition of any etype in ET^E, there is an instantiated attribute in A^E.

We say that etype ET_i is more specific than etype ET_j iff C_i^{ET} *more specific than* C_j^{ET} . We also say that entity E belongs to etype ET_i $ET \in ET^E$.

An entity can be considered as complex when it is composed by different entities. In particular a complex entity creates a structured piece of information organized in a graph where each node is an entity linked with other entities through the "part of" relation. Therefore we can say that when an entity instantiates an attribute which concept correspond to "is part of" and/or "has parts" it becomes a complex entity.

6.4. Types of Attributes

According to the data type of an attribute definition we can categorize an attribute as follows:

- **Descriptive (Textual)**: It's an attribute in which its values cannot be codified as a single entity and/or atomic concept. They are given in input from a user as a simple sequence of characters on which a process of entity/concept recognition can be applied. Descriptive attributes are those whose data type is semantic string
- **Entity (Relational)**: It's an attribute in which its value points to an entity (or an entity set) and that creates a (semantic) link between the entity owning the attribute and the target entity (the value of the attribute). It encodes a semantic relation where the name of the attribute definition encodes the relation existing between the source (the entity instantiating the attribute) and the tar-

get (the value). Relational attributes are those whose values are of data type Entity, examples can be “Birth Place” and “Father”

- **Concept (Attributive)**: It’s an attribute in which its value carries an implicit semantic (E.g. the date of birth of a person), basically all the attributes whose data type is not semantic (less) strings or entity can be considered as attributive

6.5. Meta Attributes

A meta attribute is a special attribute that describes additional properties and semantics of an object in a given context. Meta attributes are used to enable specific and advanced services on entities. A Meta attribute can be assigned to an attribute definition within a given Etype and/or Role, in the following we list those meta attributes grouped by category and sorted within a category from the strongest to the weaker:

6.5.1 Necessity Properties:

- ***Strictly Mandatory***: An attribute is strictly mandatory when each entity must instantiate that attribute and provide a (not null) value.
- ***Mandatory***: An attribute is mandatory when each entity must instantiate (by default) that attribute even if the value is allowed to be null (e.g. unknown value).
- ***Suggested***: An attribute is suggested when an instance of that attribute is not mandatory in the set of attributes of an entity.

6.5.2 Permanence Properties:

- ***Permanent***: An attribute is permanent when its value(s) is stable in time. Once a permanent attribute is instantiated it cannot be changed.
- ***Temporary***: When it’s not permanent an attribute is considered to be temporary, e.g. when its value(s) may change in time.

6.6. Etype Attributes Categories

Each attribute definition within a an Etype can be grouped inside a category. A category exists in the context of an Etype only and it is defined as the tuple EC = <C, ET, ADS> where C is a concept denoting the name of the category, ET is an Etype and ADS is a (non empty) set {AD} of attribute definitions. Note that different categories within an Etype cannot share the same attribute definition and concept. A category differs from a structured attribute definition because while the latter codifies logically indivisible pieces of information that can be shared between different Etypes, the former is used to group pieces of information within a single Etype.

6.7. Etype System Design Requirements

So far we have described the theoretical background of Entity type and attribute definition. Therefore, starting from the above discussion we plan to investigate the feasibility to develop a system that is able to:

- Provide efficient and effective way for browsing and searching the Entity type;
- Provide efficient and effective way for browsing and searching the Attribute;
- Support unskilled users in building their own etypes and attribute in a flexible and effective way.
- Support user in entity type evolution and customization;
- Support user in managing multilingual etype;

- Import and manage existing etypes provided by experts;

6.8. Conclusion

In this chapter we have described a model for attribute definitions, entity types, entities and for other related objects. We also have described the initial requirement specification of the etype management system. In chapter 10 we described the entity type management system.

Chapter 7

7. Visualization Theory

7.1. Information Visualization

Visualizations have a crucial and expanding role in cognitive systems as we acquire more information through vision than through all of the other senses combine[82]. A wide variety of uses for the term visualization is stated in [81] . "The purpose of visualization is insight, not picture" the main goals of this insight are *discovery*, *decision making* and *explanation* [83] . In K. Stuart et al.[83] visualization is defined as "*the use of computer-supported, interactive, visual representations of data to amplify cognition*". Thus they define the information visualization as follows: "the use of "*the use of computer-supported, interactive, visual representations of abstract data to amplify cognition*". These definitions mainly focus on the purpose of the visualization as the means. Visualization is an activity in which human beings are engaged as an internal construction in the mind [84,81] Considering visualization is something that cannot be printed on paper or displayed on a computer screen [85] summarized the visualization as *a cognitive activity, facilitated by external visual representations from which people build an internal mental representation of the world.* Indeed [85] Tried to make the definition independent from computers as an activity that occurs in the mind where computers with some visualization tools may facilitate the visualization process.

7.2. From Data to Information Visualization

The transformation process and the relationships between Data, Information, Knowledge and Wisdom has been analyzed by [68 ,87 , 88, 89] as intimately connected which gives a formation of other as from data to wisdom . Nathan Shedroff, in his article in *Information Design* [86], analyzes the process of understanding data and the generation of information from data, from the information design point of view. We summarize the key features of the process as follows:

7.2.1 Data

Data are symbols, atomistic, lowest abstract or distinct pieces of information without context. In computer terms, characters, symbols, numbers or images are data. *Information Science* defines data as *unprocessed information*. They are not adequate for communicating as they lack any meaning, though sometime it is also possible that data itself carries information. It comes about through research, creation or other ways. However, they are the basic building blocks for construct information and our communicative processes.

7.2.2 Information

Data as distinct pieces of information are not adequate to establish a communicative process. To make them meaningful, it is necessary to process, organize by adding context through relationships between data, and present them in such a way so that it becomes meaningful information. It is generally the processed outcome that derived from the data. This information can be about facts, things, or anything relevant to the topic concerned or even concept that can be used in many domains.

7.2.3 Knowledge

Unlike data and information, knowledge is generally personal, subjective and inherently local [87]. When information is integrated with experience, it creates knowledge, which enables us to understand things

[85]. T. D. Wilson [88] defined 'Knowledge' as what we know: knowledge involves the mental processes of comprehension, understanding and learning that go on in the mind and only in the mind, however, much of them involve interaction with the world outside the mind, and interaction with others. Whenever we wish to express our knowledge, we express it in terms of information, which a knowing mind may understand, comprehend and incorporate into its own knowledge structures. For example, during our student life we memorize many theories and during the exams we apply these theoretical concepts to solve the problems in real-life situations.

7.2.4 Wisdom

Wisdom is the top level of comprehension. According to Shedroff, it is much more abstract and philosophical than other levels and less is known about how to make or effect it. It can be defined as the phase in which a person has acquired such an sophisticated level of knowledge of processes and relationships that it is then possible to express qualified judgment on data [85]. A fundamental philosophical definition of wisdom is to make the paramount use of knowledge. Unlike knowledge, wisdom cannot be directly transmitted or taught. It is also the understanding of what is true or right together with optimum judgment as to action.

7.2.5 Information visualization

Information visualization is located between data and information [85]. As we have mentioned before data requires the creation of relationships between data, organizing and presenting them in a proper way to make meaningful information, information visualization play role in this phase. Information visualization provides the means with which we can organize the data into a meaningful shape, present it in a meaningful and appropriate ways, and finally allow communicating the context around it. Card et al. [90] defined this process as “the use of computer-supported, interactive, visual representations of data to amplify cognition.” Information visualization aimed at presenting the data in such a way so that our cognitive processes can easily create information from it.

Therefore our challenge is presenting these data through information visualizations in an effective ways to take proper advantage of our most powerful human perceptual system and to improve the cognitive process accuracy. However, designing an effective presentation is challenging, as it requires a theoretical understanding of how we perceive.

7.3. Perception

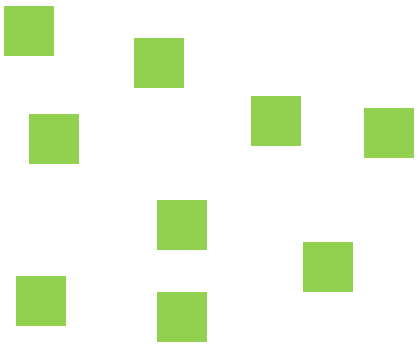
A design is an arrangement, a way of organizing the things that can be seen. The elements of design refer to “*what*” are used and the principles of design refer to “*how*” they are used [92], which describe basic ideas about the practice of good visual design. The elements of design are the building blocks of design, includes line, color , shape etc which are processed by our visual perception. Principles are problem specific while elements are use according to the principle to solve the problem, making these visual choices is design [92].

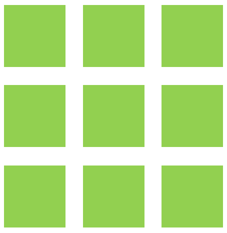
7.3.1 Gestalt Principals

In the twentieth century, a group of German psychologists developed theories that attempt to give explanation how human visual perception works. The German word "Gestalt" roughly translates to "unified whole" or "form," and the Gestalt psychologist's believed that the whole is greater than the sum of its parts. One of their basic findings was people have a tendency to organize visual elements into groups or unified wholes when certain principles are applied. Our visual system automatically imposes structure on visual input and is wired to perceive whole shapes, figures, and objects rather than disconnected edges, lines, and areas. In order to understand what we receive through our senses, they theorized that we try to organize information into definite groups. This allows us to construe the information entirely without unnecessary repetition. These theories are known as the Gestalt principles [93,94,95] of visual perception. The most widely accepted results from psychological studies based on human perception of visual elements are the Gestalt principles of visual perception. Each of these principles can be applied to any element of design such as line, shape, and value. In the subsequent sections we will demonstrate the relationship between the elements and principles of design using Gestalt theory.

7.3.1.1.Principle of Proximity

This principle states that elements or objects that are close to one another are tend to be perceived more related or as group than elements that are spaced farther apart.


(a)


(b)

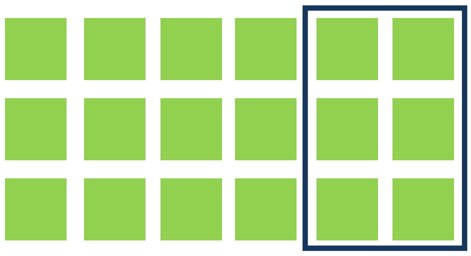
(c)

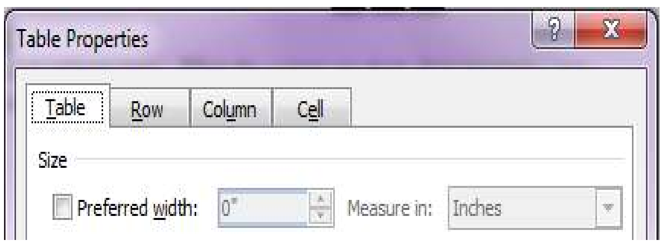
Figure 7 Principale of Proximity

In figure 7 (a), the nine squares above are positioned without proximity. They are perceived as separate shapes. In figure 7 (b) the same squares when we placed them closer we are indeed giving close proximity, unity occurs; we perceive this collection of square close to each other as forming a group. While they continue to be separate shapes before, they are now perceived as one group. Figure 7 (c) depicts that even if the color, sizes, shapes, and objects are very dissimilar, they come into sight as a group if they are close together, the relative distance between objects affects our perception of how the objects are organized.

7.3.1.2.Principle of Uniform Connectedness

The principle of uniform connectedness refers to the fact that objects that are connected by uniform visual properties are perceived as being more related than objects that are not connected. This principle generally overpower the other principles (e.g., proximity, similarity)


(a)



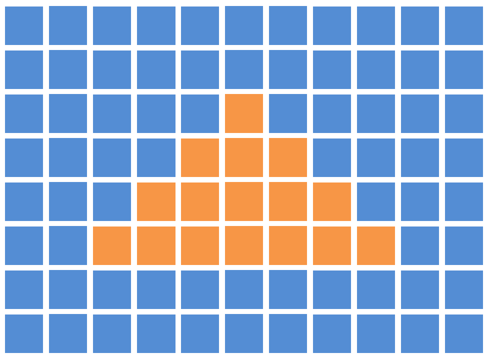
(b)

Figure 8 Principle of Uniform Connectedness

In figure 8 (a) even though the color of the squares and the spacing between square is consistent within this collection of green objects , those inside of the connecting lines are perceived to be more connected than the rest. In graphical user interface design, it is common to employ uniform connectedness to show context. Tabbed navigation is a common example of this principle (see figure 8(b)).

7.3.1.3.Principle of Similarity

It states that objects that are similar are perceived to be more related and often perceive them as a group or pattern than objects that are dissimilar. It is a powerful mechanism for communication. There many ways that objects can be perceived as being similar, therefore, related. For instance, some of the factors are, similarity of color, size, shape, dimension, texture, and orientation, to name just a few.


(a)

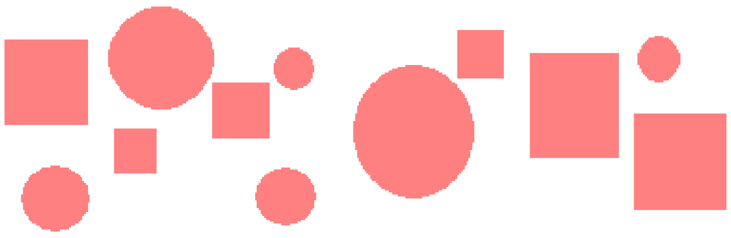

(b)

Figure 9 Principle of Similarity

In the graphic figure 9(a) , the viewer is likely to perceive a triangle shape in the middle, because we group the yellow rectangles in our perception, though each individual object is the similar shape. If it is ask to classify the objects above figure 9 (b), almost anyone would say that the strongest communication toward classification is dependent upon shape. Based on shape, it seems that the squares are related to one another and the circles are related to one another. It is important to note that in this example shape, not proximity or size provides the strongest communication. In designing it is essential to supply visual clues as to which interface objects are related to one another so that the end user of the application can promptly identify organization and make sense of how to use or interact with it.

7.3.1.4.Principle of Continuity

Our visual perceptions tend to continue shapes beyond their ending points and tend to continuous forms rather than disconnected segments.

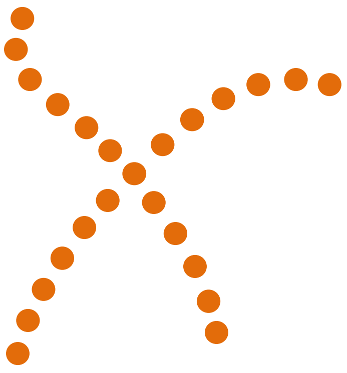


Figure 10 Principle of Continuity

We have a tendency for continuous figures. In the example above, we perceive the figure as two crossed lines intersecting and forming an X shape, instead of seeing four separate line segments meeting at the center.

7.3.1.5.Principle of Closure

It is related to Continuity, which states that our visual system automatically tries to close open figures so that they are perceived as whole objects rather than separate pieces. Closure means that we "close" objects that are themselves not complete; not only completing the figure in our perception, but perceiving the figure as having an extra constituent of aesthetic design; This happen because of our brains often ignore contradictory or incomplete information and fill in gaps in information.

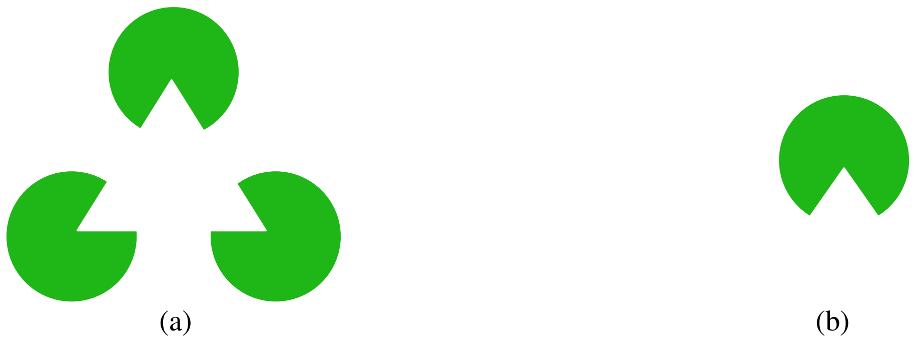


Figure 11 Principle of clouser

Our visual system is so strongly biased to see objects that it can even interpret an absolutely blank area as an object. Despite the fact that the information is incomplete, our perceptual system can close the gaps and perceive the whole. In the Figure 11(a) we see the combination of shapes as a white triangle overlapping three green circles, even though the figure actually contains simply three green Pac-Men (an old video game character). It is a form that we see because of our perceptual system tends to close open parts, and it acts like an independent whole. One interesting thing is if we cover everything except one circle(see figure 11-b), it no more looks like a white corner over a green circle rather it looks more like a Pac-Man. This is the type of phenomenon German psychologist Kurt Koffka was talking about when he made the famous statement “the whole is other than the sum of the parts”.

7.3.1.6.Principle of Symmetry

A third fact about our tendency to see objects is captured in the Gestalt principle of Symmetry. It states that we tend to parse complex scenes in a way that reduces the complexity. The data in our visual field usually has more than one possible interpretation, but our vision automatically organizes and interprets the data so as to simplify it and give it symmetry.

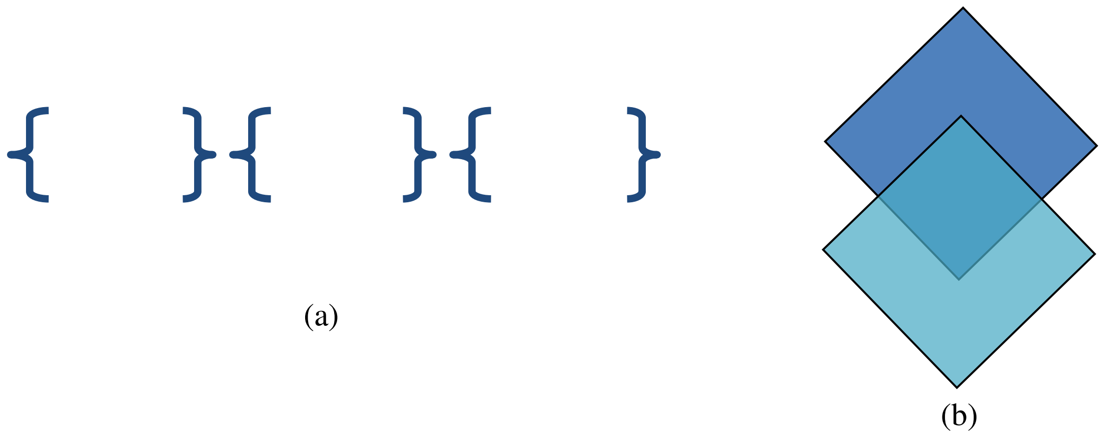


Figure 12 Principle of Symmetry

The human visual system tries to resolve complex scenes into combinations of simple, symmetrical shapes. In figure 12 (a) we likely see three sets of opening and closing brackets in the above image. Here symmetrical balance is stronger than proximity. If we look at figure 12 (b), if we perceiving according to the principle of symmetry, we most likely see two diamonds overlapping each other, rather than three objects, a small diamond and two irregular objects above and below it.

7.3.1.7.Principle of Figure and Ground

This principle states that our mind separates the visual field into the figure (the foreground) and ground (the background) based on one or more of a number of possible variables, such as color, contrast, size, etc. The foreground consists of those elements of a scene that are the object of our primary attention, and the background is everything else. The Figure/Ground principle also specifies that the visual system’s parsing of scenes into figure and ground is influenced by characteristics of the scene. For example, when a small object or color patch overlaps a larger one, we tend to perceive the smaller object as the figure and the larger object as the ground (see Figure 13 Principle of Figure and Ground).

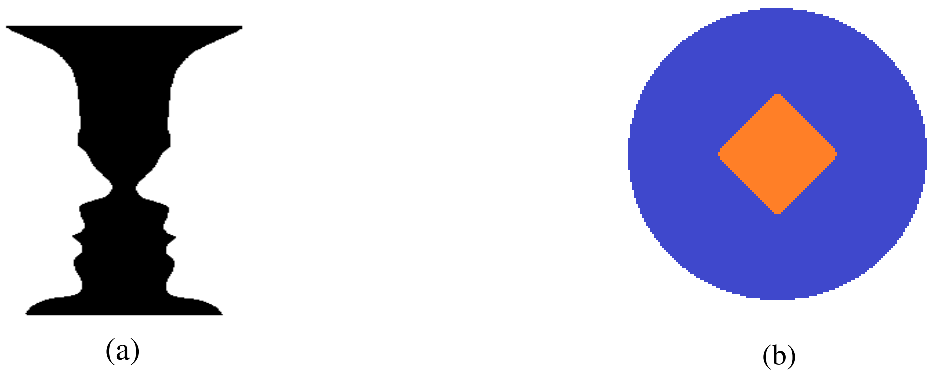

(a) (b)

Figure 13 Principle of Figure and Ground

The classic example of figure and ground is vase/face figure 13 (a) . Indeed we seem to have a natural tendency to perceive one phase of an image as the figure or fore-ground and the other as the ground or back-ground. If we look at the picture Figure 13 (a) indeed there is only one image, and yet, by altering nothing but our mind-set, we can see two different stuff i.e., vase/face. It doesn't even seem to be possible to see them both at the same time! When objects overlap, we see the smaller as figure on ground Figure 13(b). Simple composition may have only one figure. In a complex composition there will be several things to perceive. As we look from one to another they each become figure in turn.

7.4. Memory

Human memory has strengths and weaknesses. Therefore it is important for the designers involve in interactive systems design to have some understanding of human memory. This understanding is needed to support and augment human memory rather than burdening or confusing it. Indeed it does not matter exactly where in the brain something processed and perceived, rather it is essential for effective design to know how and what kinds of visual information the brain usually can process efficiently. The memory is a function of the brain, which process, store and reason information, this is also responsible for constitutes the common ground for perception. In fact, we receive light through the eye, which generates a visual stimulus. This stimulus is translated into neural signals by the retina and passed to the brain, where it is processed and perceived. In cognitive psychology, there is one memory system, but it is normally divided into three functions for storage [96].

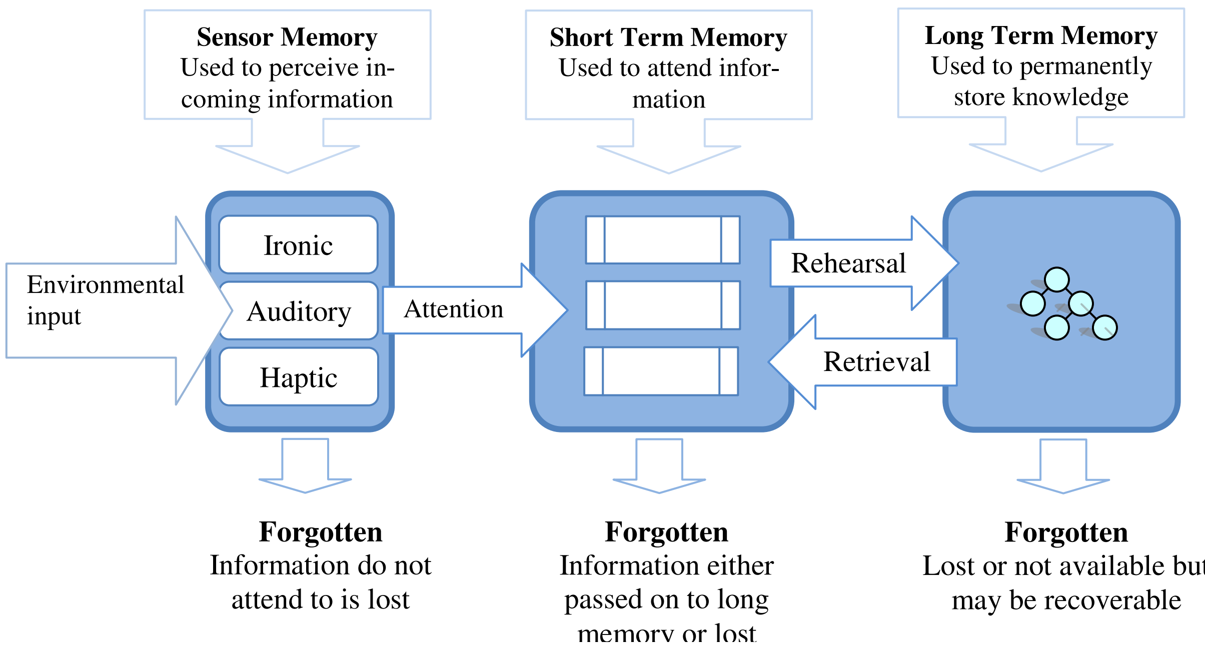

Sensor Memory
Used to perceive in-coming information
Short Term Memory
Used to attend infor-mation
Long Term Memory
Used to permanently store knowledge
Environmental input
Ironic
Auditory
Haptic
Attention
Rehearsal
Retrieval
Forgotten
Information do not attend to is lost
Forgotten
Information either passed on to long memory or lost
Forgotten
Lost or not available but may be recoverable

Figure 14 Memory model explaining some steps involved in sensory, long-term and short-term memory

7.4.1 Sensory Memory

The sensory memory stores information about the world coming from sense organs in a fairly unprocessed way for less than a second. It holds an exact duplicate copy of what is seen (visual) or heard (auditory). It only lasts between 250 and 500 milliseconds or less. It has unlimited capacity. For each sensory channel there is a sensory memory:

- *Iconic memory* for visual stimuli, it holds visual images.
- *Auditory memory* for aural stimuli, it holds sounds. This memory can retain longer than iconic memory.
- *Haptic memory* for touch.

The processing that takes place in iconic memory is called preattentive processing [97]. During preattentive processing, only a limited set of visual attributes is detected. Some of these basic features are colors, contrast, shape, size, end of lines, and curvature [97]. However the content of the sensory memory is still abstract with no meaning attached to its input. Meaning is generated when the data is transfer to the central cognitive short-term memory for interpretation. object identification is done by the cognitive processor which is associated with short-term memory used for storage of temporary working information. From sensory memory, some of the information is transferred into short term memory by attention, thereby filtering the information to only those which are of interest at a given time. The amount of information transferred is limited. In the design of a visual representation the properties of sensory, short-term, and long-term memory have important implications. For example, if we want people to understand and get the information quickly, we should present it in such a way so that it could easily be detected by their iconic memory. In particular, designer should pay attention to preattentive visual processing, as preattentive visual attributes are perceived instantaneously without any consciousness. Indeed it is fundamental for creating visual representations.

7.4.2 Short-Term Memory (STM)

As mentioned above, selective attention determines what information would pass from sensory memory to short-term memory. This memory performs the intellectual tasks associated with consciousness [16]. Unless people rehearse the material, information remains here approximately a few seconds to a minute. Depending on periodic rehearse it can remain for few hours. STM store images, sounds, especially in recalling words. It works basically the same as a computer's RAM in that it provides a working space for short computations and then transfers it to other parts of the memory system or if not use then discards it. That's why this memory also called *working memory*. This memory has extremely limited capacity. The length of this thought to be about seven bits, because of this we normally remember seven items, while Miller [98] has found that STM has a limited capacity of around 7±2 'chunks' of information. These chunks of information can be simple characters or numerals or even more complex abstracts and images. This characteristic of STM says that it is possible to improve the capacity or efficiency of short-term memory if the presented information is organized in chunks. Considering this phenomena of STM, Cognitive load refers to the total amount of mental activity imposed on working memory at an instance in time[99]. The fundamental principle of cognitive load theory argued that it is possible to increase the quality of instructional design by giving greater consideration to the role and limitations of working memory [99]. The same principle can be also apply in visualization. Mapping information in visual representations is usually maintained in the SMT. As we have seen from the above discussions that the capacity and duration of holding information of this memory is limited, therefore designers of visual representations shouldn't constrain users to remember more than nine chunks of information. Information can be made easier to inspect if they are structured. One example is telephone numbers. Usually, such number is broken into parts to make them easier to scrutinize and remember. Sometime this principle is also miss used in website navigation design. Based on this principle some designer argue that website navigation cannot have any more than 7 items in it though it is simply not true. Indeed there is no demand on the user to actual remember the navigation which user can easily do by scanning again to refresh our memory of what was there. A common tendency of the people is they tend to inspect rapidly for relevant information while navigating software or sites instead analyzing screen cautiously and read every word in it. Therefore, it is important to presented the information in a concise, structured way so that it becomes easier for people to scan and understand. They should also follow to the rules of graphic design, some of which were presented in previous section. The important thing is the capacity of SMT can be expanded by abstracting qualities from the basic information and store the abstraction instead.

7.4.3 Long-Term Memory (LTM)

This is relatively permanent storage. It is intended for storage of information over a long time which could be recollected for lifetime. The information could last for 30 seconds or for decades. Information from the working memory is passed to the LTM after a few seconds and is stored on the basis of meaning and importance. Long term memory can further divide in to two basic types: episodic and semantic memory. Episodic memory helps us recollect various experiences and events that happened in our lives in a serial form. On the other hand semantic memory records concepts, facts, and our skills that we acquired throughout our lives and helps us to learn from our various experiences more facts and new concepts.

7.5. A Reference Model

Riccardo Mazza in his book Introduction to Information Visualization [85] described a reference model that shows the process of generating an artifact of visual representation. In his model a visual artifact creation process has been modeled through a sequence of successive stages, which we present here with some modifications:

1. Raw Data:
2. Creating Data structure:
3. Visual mapping:
4. View creation:

7.5.1 Raw Data

This step involves accruing the raw data, which we have to visualize. Raw data, also known as source data or atomic data, is data that has not been processed in order to be displayed in any sort of presentable form [100]. Unless processed it this raw data may look nearly meaningless, but it may also be in a form that someone can interpret, depending on the situation. This raw data is the data that have been supplied by the world around us[85]. They can be any unprocessed computer data stored in a file. They can also be data generate by any sensors or tools that has been collected but not formatted or analyzed yet. Notice that, not necessarily they have to be always unprocessed; data stored in database in electronic format and with a well-defined structure are already ready to use. In this case preprocessing phase involves just fetching it from the database and converting them into the structured format so that the visualization system can visualize them. For an example, let us assume we want to present student grade sheet for the course Physic. Exam is divided into three part written, assignment and oral. Let assume we got a file of the student grade with this format as a raw data:

A02LucaPHYW34A35V15
A23AndrePHYW35A35V16
A24CarloPHYW38A38V17
A23MattiaPHYW39A38V19

7.5.2 Data Structure

This process involves preprocessing the raw data if the data are unprocessed, to a suitable format so that visualization logic can utilize them to visualize. However if the data is already well formed in database in electronic format in this case it will just fetch the data and convert them into the structured format to use by the visualization system. Raw data can be formatted in many ways, for example by removing unnecessary data which are generated by the system but not interesting for display. It could also add additional information or calculations for obtaining new data. In the above example we can obtain new data by calculating the total number for each course thus represent in the visual system. It may also need to add attributes [85] or also called metadata [90] to the data that may be used to logically organize the data structure that would use by the visualization system. The intermediate data structure, of the example we are processing, could therefore look like the following:

Student id	Name	Written Exam	Assignment	Total
A23	**Mattias**	**74**	**18**	**92**
A24	**Carlo**	**68**	**18**	**86**
A23	**Andrea**	**55**	**15**	**70**
A02	**Luca**	**64**	**15**	**79**

In this structure in particular, we have filtered out some information, such as the tag added by the system, for example an exam type identification character before each grade. We also remove the repeated course code. We attribute *Total* obtain from the calculation by calculating the total number for each student

7.5.3 Visual Mapping

In this stage visual structure and their location are determined for the data. The critical thing is how best to transform the processed data into visual mapping that people can understand for optimal decision making. According to [85] this step involves defining the visual structures that will use to map the processed data structure from the raw data and their location in the display area. Visual mapping requires identifying three structures [90] that correspond to the data that we want to represent visually. These are spatial substrate , graphical elements and graphical properties.

- **Spatial substrate**: The spatial substrate defines the position or the dimensions in physical space where the visual representation is created. It can be defined in term of axes (e.g., x-axes and y-axes). Depending on the type of data that would map on the axis the type of axis varies. In particular an axis can be *quantitative*, *ordinal* or *nominal.* [83] defines the quantitative, ordinal or nominal as follows(they called it variable type):
    - N = Nominal (are only = or ≠ to other values),
    - O = Ordinal(obeys a< relation) or
    - Q = Quantitative (can do arithmetic on them)

A *nominal* variable N is an unordered set, such as name list {*Mattias, Carlo, Andrea, Luca*}. This is basically the labeling function. Sometimes numbers are also used in such a way there is no sense in which the number can be placed in an ordered sequence. For example the bus number to indicate the route on which the bus travels. The transport authority put a number on the front of a bus, this number generally doesn't indicate any ordered sequence but identifies the route, thus it has a purely nominal value.

An *ordinal* variable O is a ordered set, [85] defines ordinal as follows "data of a non numeric nature, but which have their own intrinsic order, such as the days of the week". The position of a in a list is an ordinal quality. Similarly it is possible to say from a set of items that a certain item comes before or after another item. When we create merit list of a group of students in order of their achieved mark, we indeed create an ordinal scale.

A *quantitative* variable Q is a list of integers, real numbers or a numeric range, such as total mark [0,100] . However it is also possible to transform one type to another type. For example the nominal name list becomes ordinal when we sort them lexicographically, such as name list {*Andrea, Carlo, Luca, Mattias*}, here the names are sorted lexicographically. In the reference model,
an axis is quantitative, when there is a metric associated to the values reported on the axis; An axis is ordinal, when the values are reported on the axis in an order that corresponds to the order of the data; An axis is nominal, when the region of an axis is divided into a collection of sub regions without any intrinsic order.

- **Graphical elements**: everything visible that appears in the space. There are four possible types of visual elements: *points, lines, surfaces, and volumes*

- **Graphical properties**: The graphical properties (also called retinal variables) are properties of the graphical elements which are perceived immediately, the retina of the human eye is very sensitive to them. It significantly impact on interface design the way in which these properties are used. Notice that human visual perception doesn't perceive all the properties in the same way. The most common graphical properties are size, orientation, color, texture, and shape. These are applied to the graphical elements and determine the properties of the visual layout that will be presented in the view.

To continue with the process of generation, we have to associate a visual structure with which to map the data that we wish to represent, to the data structures. Attributes of a visual structure are determined by one or more attributes of a data structure. For example, the height of a bar could be determined by a student's total score on the final exam for a course, whereas the bar's color might indicate if the student is a male or female. In the specified case, we have five attributes to represent:

Attribute	***Data Type***
Student Id	Ordinal
Name	Nominal
Written Exam	Quantitative
Assignment	Quantitative
Total	Quantitative

7.5.4 Views

A system's visual design is usually the first thing that users notice and the visual impacts heavily influence the user's perception. However as they become more familiar with the system they tend to focus less on visuals and the system functionalities start to getting important. The views are the final result of the generation process. They are the result of the mapping of data structures to the visual structures, generating a visual representation in the physical space represented by the computer. They are what we see displayed on the computer screen.

7.6. Conclusion

Visualization provides not only an ability to comprehend large amounts of data but also allows the perception of emergent properties that were not anticipated. In this chapter, we have illustrated some important principles of visual perception. We have presented classics Gestalt principles, a designer can get interesting insights into the design of groups of objects to conform unity thus make the visual representations more effective. We also have seen how short-term memory and preattentive processing play a very significant role in the design of effective visual representations.

Chapter 8

8. Linguistic and Concept Knowledge Management Consol

The capability to systematize and manage an emerging Linguistic and Ontology/Concept Knowledge is key to an editor's usability. Convenient and intuitive presentations and manipulations of a Word and its Concept, Concept's interlinking concepts and relations are essential. Concept hierarchies also have multiple inheritances, convenient and insightful presentations while keeping the associations straight is a challenge. Therefore, our aim is to create a visualization that will effectively display all this elements and at the same time let the user perform various operations easily on the Linguistic and Ontology/Concept Knowledge. Universal Knowledge Base (UK) consists of four different modules namely Linguistic Knowledge, Ontology/Concept Core, Domain Knowledge and Entity Core. This document describes the management console for the Linguistic and Conceptual part.

8.1. Functional overview and requirement description

In the chapter 4 we have described the theoretical background of Linguistic and Concept Knowledge. As explained in the introduction chapter the process of synthesizing new Linguistics knowledge is very tough and demanding. There is neither an easy way nor a straightforward way to achieve this goal. As underlined in the State of the art section, there are many tools for search and navigation Linguistic databases but not enough that allows creating Linguistic knowledge collaboratively. Therefore, starting from the above discussion we plan to investigate the feasibility to develop a system that is able to:

- Provide efficient and effective way for browsing and searching the linguistic knowledge;
- Provide efficient and effective way for browsing and searching the concept knowledge;
- Support unskilled users in building their own taxonomies in a flexible and effective way.
- Support user in linguistic and concept knowledge base evolution and customization;
- Support user in managing multilingual linguistic knowledge
- Import and manage existing linguistic and concept knowledge provided by experts;

According to this view we can identify the following sub-tasks that would support by the system:

- **Building language specific linguistic knowledge base**: taxonomies representing linguistic knowledge in different languages will be manually made according to the theoretical approach described above.
- **Building language independent concept knowledge base**: language independent concepts and their relations representing language independent representation of the linguistic knowledge will be manually made according to the theoretical approach described above.
- **Finding a suitable visualization technique to represent Linguistic and Concept knowledge**: We need to investigate how to deal with the enhanced complexity due to the multilingual linguistic knowledge and their concepts and relations, which facilitates the comparison of the lexicon of the aligned languages
- **Exploring possible relations between linguistic knowledge and related concepts:** As a first step we would put in the Linguistic Knowledge only the natural language information e.g., words, senses and group of words as synset. Subsequently, we can then link each synset to a language independent concept in the concept knowledge. This entails that Linguistic knowledge will be partitioned according to languages whereas the concepts remain language independent.

- **Managing concept knowledge:** While editing concept knowledge, the system should takes in to account different uses of concepts in the different languages.
- **Managing universal knowledge:** The system takes in to account while editing Linguistic knowledge with particular emphasis to their uses. For example, when we delete a synset from the linguistic knowledge we have to take into account if the concept corresponds to that synset is used by domain or etype part (see chapter 5 and Chapter 6 for a more detailed description of domains and etypes). For example, if it is the last word, in the last sense, in the last language for a concept used somewhere, its deletion should be forbidden.
- **Performing search and navigation:** Developing an efficient multilingual search and navigation environment to realize the potential benefits of Linguistic and Concept Knowledgebase.

From an implementation point of view, the abovementioned tasks can be categorized into two different parts:

- **Search and navigation part:** It includes Search, browsing words and its components such as synsets and concepts hierarchy. More in details:
    a. **Search and navigate words**: The list (in alphabetical order) of already defined words must be available on request to the user. A text search facility to search among them must be available.
    b. **Search and navigate synsets of a word**: By selecting a word, it must be possible to list the corresponding synsets thus emphasizing the corresponding part of speech (i.e., Noun, Adjective, Verb and Adverb).
    c. **Search and navigate concepts**: By selecting a synset of a word, it must be possible to navigate the concept hierarchical of that synset. For each node, the corresponding concept details must be visible.
- **Maintenance part:** e.g., creation, updating and deletion operations. This part is, in turn, articulated into operations on linguistics and operation on conepts:
    d. **Operations on linguistics**: This is the list of operations that the user interface must support on linguistic and concepts:
        i. *Create a word*: A word is created by assigning a lemma and any eventual exceptional forms. The user interface must provide a specific facility to assigning the lemma and adding exceptional forms.
        ii. *Delete a word*: When selecting a word, it must be possible to delete it. The system takes in to account its use while deleting word.
        iii. *Create a synset*: A synset is created by assigning each word the sense of information conveyed (e.g., gloss and part of speech). Creating a synset should also create a concept for the synset when the concept is not already present in the system.
        iv. *Update a synset*: It must be possible to change the gloss, the part of speech or the rank of a synset.
        v. *Add a word to a synset*: It must be possible to add a word to a synset. In case the word doesn't exist it must be possible to create it and add it thus specifying the new word through its corresponding part of speech.
        vi. *Remove a word from a synset*: It must be possible to select a word from the synset and remove it. This does not result in the deletion of the word because this word might belong to another synset in different context.
        vii. *Delete a synset*: It must be possible to select a synset and delete it. This does not result in the deletion of the words associated with it unless any word is the last word in the knowledge base. The system takes in to account to the use of its concept while deleting synset.

e. **Operations on concepts** : This is the list of operations that the user interface must support on a facet (not necessarily associated to a domain):
    i. *Create a relation*: It must be possible to create a new relation between two existing concepts.
    ii. *Update a relation*: When selecting a relation between two concepts, it must be possible to change the relation
    iii. *Remove a relation*: when selecting a relation between two concepts, it must be possible to remove the relation.

All operations of creation of new knowledge, updating, deletions search and navigation will be pursued in the Knowledge Base through an Inference engine. In subsequent sections, we will make more precise design specifications for an effective and user-friendly management system to perform CRUD operations including search and navigation on the linguistic and concept part for the *UK.*

8.2. Search and navigation

Search and navigation services offer the means to unlocking the wealth of data that exists in digital form. Developing an efficient search and navigation environment requires leveraged cognitive study to effectively realizing its potential benefits. Therefore one of the basic goals is to reduce the cognitive load requested to the user. In other words, the number of items that at each step the users must track and process should be reduced, so to allow them to concentrate on specific parts of the task. This can be achieved by distinguish amongst objects (e.g., word, sense, concept) those that are conceptually similar and group them together. Visualization and navigation structures should determine by analyzing these object characteristics and the pattern. For example, words can be shown in alphabetic ordered lists , synsets in structured format whereas for concept (i.e., ontology) the most suitable visualization would be hierarchical. These visualization solutions will allow users to focus only on the relevant part they are interested in. Search and navigation can be divided into three different parts depending on the type of objects of interest. In this way, the number of visualized information will be minimized thus reducing the information overload on users. We are also interested in the interrelationship and the multilingual relationship involving both in synsets and concepts. Depending on the type of objects each part would be responsible for a set of related functionalities:

8.3. Search and navigation of Words

Providing the facility for word search is one of the basic operations in any thesaurus. The goal of word visualization is to provide a meaningful context in which the user can explore the semantics of word from different points of view. The idea is to show the search result as an alphabetically ranked list of words, as shown in Table 2, thus combining the use of keyboard and a scrolling list to make navigation more efficient. Basically, a specific word can be singled out easily from the alphabetically ranked list.

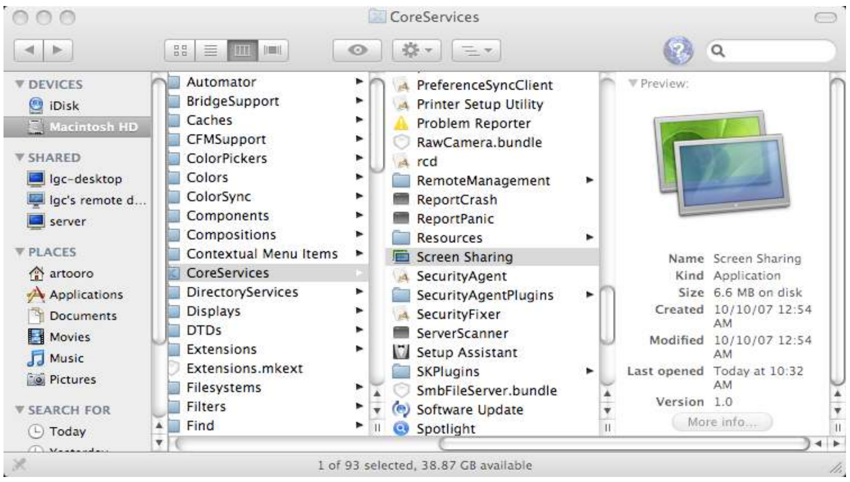
Words
computable
computation
computational
computational linguistics
computationally
compute
computed axial tomography
computed tomography
computer

Table 2 List of result for the search token "comput"

However, the length of lists can jeopardize the efficacy of this solution. When lists tend to be very long, filters can become useful elements. In particular, information visualization systems appear to be most useful when they allow users to refine search results. More in particular, pressing any key will automatically filter the words whose first character matches the pressed key itself and any subsequent keypress will further refine the search considering also following characters. This mechanism is an obvious and widely adopted solution to support search and navigation but a proper strategy for dealing with language mode is needed. Auto-suggestion features are a recently common search trend. Search result list would used to navigate through the words and provide a means for selecting query scope

8.4. Search and navigation of Synsets

Every word has a synset. Each synset organizes a set of words with same meaning according to logical groupings. If we look at the conceptual model of Linguistic and Concept Knowledge (see Figure 5 Conceptual model of Linguistic and Concept Knowledge) we see there is a list of words. In this words list each word has one or more synsets. Each synset corresponds to a concept in a language and word has no parent. This model we can easily map with a directed graph because the edge relation is asymmetric. The most common example of such visualization is the directory structure of a hierarchical file system. Figure 3 shows an example of a horizontal layout. Clicking on an item switches it from a 'collapsed' to an 'exploded' state. The path that has been followed through the selection is also highlighted. This characteristic we can use to maintain the context. Notice that, the 'child' branches of only one item can be displayed within each level; this approach does allow the user to explore the navigation in a very 'space efficient' manner.



Name	Screen Sharing
Kind	Application
Size	6.6 MB on disk
Created	10/10/07 12:54 AM
Modified	10/10/07 12:54 AM
Last opened	Today at 10:32 AM
Version	1.0

1 of 93 selected, 38.87 GB available

Figure 15 Hierarchical tree browser with a horizontal layout for a file system

In our case, to present word, synsets and their concepts we can use a three columns panel, where the first column is for word list, the second column is for synsets corresponding to the word and the third column is for concepts corresponding to the synsets. Each word from the first column can be treated as a parent or starting node. Clicking on a word would initiate the query and show a list of synsets for the word that has been clicked, therefore passing to an 'exploded' state in the second column. Clicking on a synset would show the corresponding concept, therefore passing to an 'exploded' state in the third column.

A common tendency is, during navigation people don’t inspect screens watchfully and read every word instead they scrutinize swiftly for relevant information according to their goal. Therefore when information is offered in a brief, structured way, it is easier and convenient for people to search and comprehend. The bottom-line is, the more structured and briefs the presentation of information, the more swiftly and effortlessly people can examine and understand it. To make structured we can present a synset as a block of information. Each block represents a sense/synset , contains the associated word together with its synonyms and corresponding gloss. A set of synsets of a word can be presented as a blocks of information in a vertical linear list in the second column.

- **Synsets category:** Synsets are further categorized by parts of speech (POS). We assume users will start their search with words and they will get in to the synset through the words navigation. Therefore, the scope of the synset visualization depends on the search word. In some cases, this list could be quite long, as it will include the synsets from all the part of speech (POS). We can use POS to further filter the query result and grouping the synsets into consistent categories. There are many alternative ways of showing POS groups. One possible way is to show them in an expandable vertical list. However showing this way has a drawback. If we expand all the categories, then the result will be a long list. Furthermore the user would have to scroll up and down to find an item from the list. An alternative way is to show them in a Tab view. A tab view provides a convenient way to present information in a multipane format. The tab control is displayed horizontally centered across the top edge of a content area. Each tab would contain one group representing a part of speech. Users can easily switch between tabs so to see the search results associated with each of them. As a management tool it is also intuitive there should have some way to edit these existing information. For example, editing gloss of a synset, changing part of speech, reordering the ranks.

- **Synsets Rank**: Conceptually senses/synsets associated to words are arranged according to their rank in ascending order. This ordering is determined by frequency of use in semantically tagged corpora [46], thus the synset in rakn 1 is the most frequently used. We can present this by vertically ordering synsets in a list, wherein the synset with rank 1 would be in the first position of the synset list, the synset with rank 2 would be in the second position and so on. In WordNet data senses that have not occurred in the tagged text are presented in random order.

- **Words Rank**: There are some words that people use more frequenlty and there are some words those people use more rarely and this is an important psycholinguistic fact about the mental lexicon. There are many ways to determine the familiarity of a word. One good way is the frequency of use. Another alternative way that has been used by WordNet to indicate the familiarity is the frequency of occurrence and polysemy. According to them, the more frequently a word is used, the more different meanings it will have in a dictionary. [46] Showing polysemy seems to predict lexical access time as well as frequency does. In WordNet polysemy uses as an index of familiarity. We can present this by horizontally ordering words inside the synset block, which means that the word with rank 1 would be in the first position of the word list, the word with rank 2 would be in the second position and so on within the synset.

8.5. Search and navigation of Concepts

Every synset has one concept. Concepts are linked between them according to their semantic relationship and build a hierarchy of concepts. A concept hierarchy generally consists of associative and hierarchical relations. The lemma of the representing word of a synset corresponds to a concept represent the leaf node label of the hierarchy. We can visualize the concept hierarchy in two different ways, from more general to more specific or from more specific to more general. If we consider the former, the concept expressed by the parent node is, as expected, more general than the concepts expressed by its children. In the other case, the direction would be opposite.

The more intuitive way to represent a hierarchy of concepts is through a rooted tree where the root label is the name of the concept and internal and leaf nodes' labels represent either its child or parents depending on the view direction of the concept hierarchy. Such tree view mechanism is very well-situated and familiar for the user. However large trees create some difficult problems. If the number of nodes is large it can compromise performance or even arrive at the limits of the viewing platform. Even if it is possible to show all the nodes, the question of viewability or usability arises, because it will become impossible to differentiate between nodes and edges. …

One can easily expand and collapse the nodes and is capable to get a quick outline over the hierarchy. Notice that, if we consider parent child relationship, a concept can have multiple parents and multiple children. For example, if we consider *Bird* as root and view it from more general to more specific, the concept "*Bird*" has 5 parent concepts and 38 child concepts. Yet, for the wide number of relations and concepts, the illustrative potential of a tree might not be enough. A concept with multiple parents is not so easy to represent in combination with an effective representation of the relations. It is desirable for the visualization to indicate concepts with multiple parents and provide efficient means to view all direct ancestors of a concept. A common technique followed by many ontology editors is visualizing multiple inheritances by replicating child nodes under all their parents and many of the Hierarchical visualizations even do not support this feature. However our intuition is DAG (directed acyclic graph) visualization instead of a tree might, in this case, help to overcome this visualization problem. This example also brings back the usefulness of filtering amongst results. The concept “Bird” has 38 child concepts and these, in turn, have their own children too. As a result, the resulting tree would be very wide. In cases like these, filtering out filter out unintended relations and reduce the information overload becomes important.

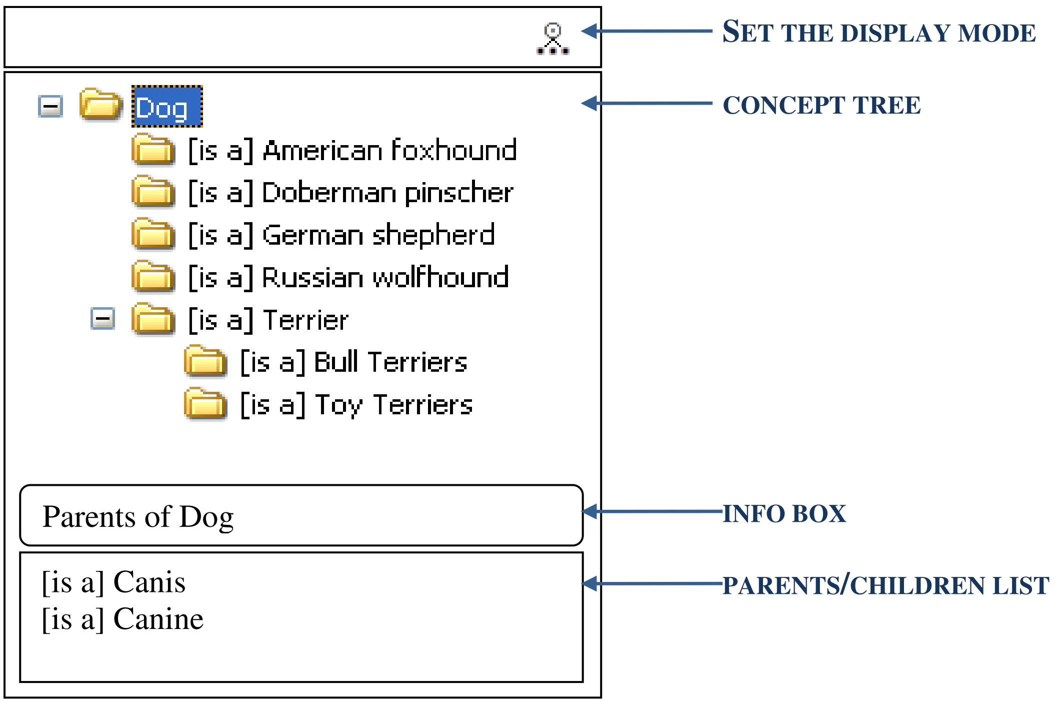
From the above analysis we can identify some essential elements that will guide the way of displaying the DAG:

- We should have a way to change the overall display mode of the concepts visualization. There are two display modes: “Show from parents to children” and “Show from children to parents”
- There should be a relation filter to reduce the complexity of the visualization, e.g. by selecting a specific relation and filtering out the others representing the DAG.
- There should be a Concept tree that allows the navigation through the concepts. If the display mode “Show from parents to children” is selected, then the root will be treated as a parent concept and its children will be shown subsequently. In the same way, if “Show from children to parents” is selected, then the root will be treated as a child concept and its parent(s) will be shown as subsequent children. Also, the kind of each relation (e.g., is-a, part-of) existing between concepts should be displayed in some way.
- As the concept hierarchy is a DAG, there should have a way to visualize the multiple parents and children thus allowing the DAGs navigation in a natural way.
- The properties associated with a relation are very essential and an effective visualization should include their representation.
- Often tree nodes label is not descriptive enough. Whenever it is needed, then concept descriptions should be displayed. Additional information can be given via tool-tips.

There are many ways to show and interact with the hierarchy of concepts. In the model below(see Figure 2), multiple parents are visualized in the **“Parent/Children Box”**. This box will always visualize the alternative parents/children (those not in the tree view) for the selected concept depending on the *display mode*. Selecting any parent from the list will add it to the tree as a parent of the currently selected node and show its parents in the Parent box. For instance, if “Change” is selected, it will become the new root of the tree (i.e., the user goes one level up following the desired path). As a consequence, the parents of “Change” will appear in the list, i.e. “focus”. The added value of our visualization lies in its expressivity. The concepts and their relationships between their child and between their parents are easy to detect.

Scenario. Consider the following scenario:

1) The user types the word “Dog”
2) The system gives all those senses in which “Dog” is present in the system
3) The user selects the desired sense (assume it is “a member of the genus Canis”)
4) The concept panel shows the hierarchy of concepts for “dog”. In the example (see Figure 16 Concepts Panel) the hierarchy is build “Parents to Children”
5) The “Parent/Children List” shows the parents list according to the selected concept from the hierarchy. By default it would show the parents/children of the root node. In the example (see Figure 16 Concepts Panel) selected node is “dog” thus the “Parent/Children List” showing the parents of “dog”.



Dog
[is a] American foxhound
[is a] Doberman pinscher
[is a] German shepherd
[is a] Russian wolfhound
[is a] Terrier
[is a] Bull Terriers
[is a] Toy Terriers

Parents of Dog

[is a] Canis
[is a] Canine

Figure 16 Concepts Panel

Notice that here are **different kinds of relations** - including hierarchical (is-a, part-of) and associative ones (e.g. member, see also). Different kinds of relations can be emphasized using the standard icons[14] also codifying the direction of the relation. The idea is we show the relation in textual form followed by a concept label. The interesting aspect of the visualization is that direction in the DAG is related to semantic relations. For example to represent the concept "Dog" with the relation "is-a" we use: >> [is a] Dog".

In the same way, assuming dog has multiple parents, to represent the DAG, we can use : ⊞ >> [is a] Dog. Notice that either "<<" or ">>" is used according to the selection mode.

8.6. Management functionalities (create/update/delete)

We clearly separate the CUD (create/update/delete) operations part from the search and navigation to avoid possible confusion. In the following sections, we describe in detail the CUD operations on words, senses ,synsets, concepts and relations between them. We also present the pseudo-code for all operations. Likewise search and navigation, the type of operation also varies from object to object. "CUD" operations can be divided into three different parts where each part is responsible for a set of related functionalities:

- **Components of Word**: We identified the following components those are directly related with a word:

    a. A string that represents the word

    b. One or more derived or exceptional form of the morphological root of a word.

- **Components of Sense/Synset**: We identified the following components those are directly related with sense/synset:

    a. A natural language description of the sense/synset

    b. Part of speech of the sense/synset; possible values are "Noun", "Adjective", "Verb" and "Adverb".

[14] http://www.destin.be/ASKOSI/Wiki.jsp?page=Icons%20for%20SKOS

c. A set of words associated to the sense/synset;

- **Components of Relation**: We identified the following components those are directly related with relation:

  a. A list of predefined relations (relation kind). Possible relations can be hierarchical (e.g., is-a, part-of) and associative (See section 4.2.2).

  b. A source Concept

  c. A target Concept.

8.7. Managing Words, Synsets and Concepts

It is important to underline that for us the notion of synset is central. The user can manage words and concepts through a synset. In particular, the user can perform the following operations on words and synsets:

1. ***Create a Synset***: There are three different scenarios in which a user can create a synset:

   a. Create a synset as a new root concept

   b. Create a synset (and corresponding concept) as child of an existing concept

   c. Create a synset to be associated with a concept that has no synset in that language

   In all the cases above the synset is created by specifying its first word.
2. ***Update a Synset***: Updating a synset can be done in the following ways:

   a. By adding a word to the synset

   b. By removing a word from the synset

   c. By updating the rank of a word in the synset

   d. By updating the rank of a synset among the synsets

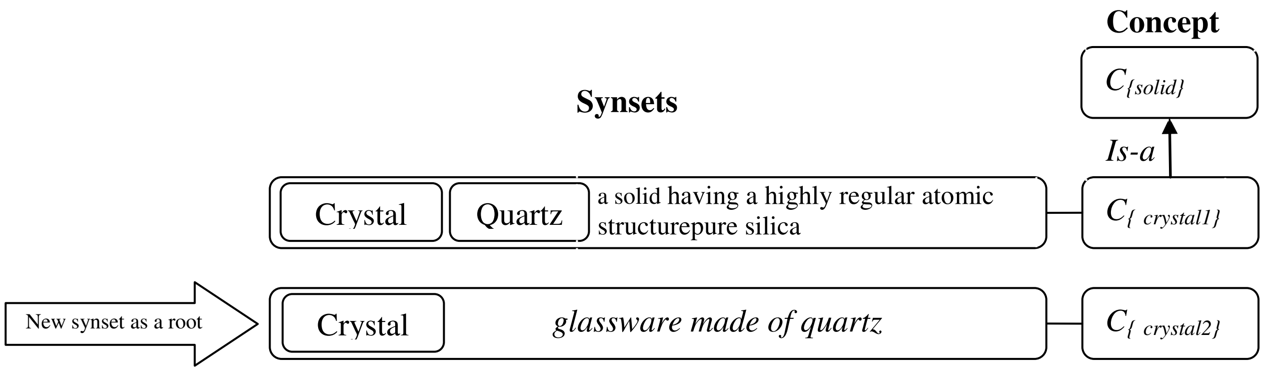
   e. By changing gloss and POS of the synset
3. ***Delete a Synset***: Deleting a synset can be done in two ways:

   a. Deleting the synset and keeping the concept

   b. Deleting the synset together with the corresponding concept
4. ***Create a relation***: Linking the (concept corresponding to the) synset with an existing concept
5. ***Delete a Relation***: Remove an existing relation among two concepts
6. ***Update a Relation***: Update the by changing the relation's type.

In the subsequent sections we will describe all these possible operations in details together with pseudo code when necessary.

8.7.1.1.Create a Synset as a new root concept



Add a new synset "*crystal*" as a new root concept

By the term "root concept" we mean here a concept that is not linked with any other concept present in the system. This situation can arise when one does no't know exactly where in the concept hierarchy this new concept could be placed. The creation of a synset corresponding to a root concept entails several subsequent steps. First we need to provide the necessary information that are required to build the synset, namely: a word for a given lemma; the description of the word in terms of sense gloss that would describe the word meaning in natural language; the Part Of Speech of the word sense (i.e., noun, adjective, adverb or verb); and the language of the word that would say which language this word belongs to. Finally, there is the need to build a concept that would be the language independent representation of that specific synset.

All operations that are required to create a synset can be merged into a single function with the following parameters *CreateSynset(lemma : String; pos : String ; gloss : String; String; lang: Locale)*. This procedure creates a word together with a new *synset* and *concept* thus it links the sense with the *synset* and *concept*. The *glosss* belongs to the synset as it represents the set of words with the same sense in a language. GetWord(lemma) is a select procedure which finds the corresponding word of a given lemma in the KB. If any word corresponding to that lemma than it returns the word, *null* otherwise. CreateX() procedure is used to create the persistence instance in the database.

CreateSynset

It creates a synset and a new root concept

lemma	= is a string;
gloss	= the natural language description of the synset;
pos	= the part of speech
L	= the language of the new synset;
w	= the word for the new synset;
s	= the new synset for the word;
c	= the new root concept

```
CreateSynset(lemma, gloss, pos , L ) {
 w :=  GetWord(lemma, L)
     if (w = null) {
       w := CreateNewWord(lemma, L); }
       s := CreateNewSynset(w, pos, gloss);
       c := CreateNewConcept(s);
 }
```

Notice that, it will create a new word in case a word for the given lemma and in the given language doesn't exist.

8.7.1.2.Create a Synset (and corresponding concept) as child of an existing concept

A concept can be created as child of an existing concept when one knows exactly where to place the new concept, for example creating a concept "puppy" as a child concept of Dog. Notice that this operation demands to specify a relation where relation R ∈ {⊥, ≡, ⊑, ⊒} (see the relation table 1 for details) that would hold between the new concept and the parent. This operation can be performed modifying the procedure just illustrated by adding two extra parameters, parent concept and the relation type. Therefore, we can define our create synset procedure as follows: CreateSynset(lemma : String; pos : String ; gloss : String; String; parentconcept:Concepr; relation: Relation ; lang: Locale). The steps are the same as those illustrated above except for the CreateRelation step that will create the relation between parent and the new child concept

CreateSynset

It creates a synset and corresponding concept) as child of an existing concept

lemma	= is a string;
gloss	= the natural language description of the synset;
pos	= the part of speech
L	= the language of the new synset;
pc	= the parent concept
r	= the given relation
w	= the word for the new synset;
s	= the new synset for the word;
c	= the new root concept

```
1.   CreateSynset(lemma, gloss, pos ,pc, r, L ) {
2.    w :=  GetWord(lemma, L)
3.         if (w = null) {
4.           w := CreateNewWord(lemma, L); }
5.           s := CreateNewSynset(w, pos, gloss);
6.           c := CreateNewConcept(s);
7.           r := CreateRelation(pc,r);
7.    }
```

8.7.1.3.Create a Synset to be associated with a concept with no sense in the language

Due to gaps in languages or because of incomplete information, some concepts in the hierarchy might not have a corresponding *synset* in a certain language. In these cases, the *concept* will have no label associated to it for that language. For example in Italian there exist a concept "Monastero" which has two child concept "Monastero" and "Convent", similar concept exists in English "Monastery" and "Convent" but they do not have any parent concept.

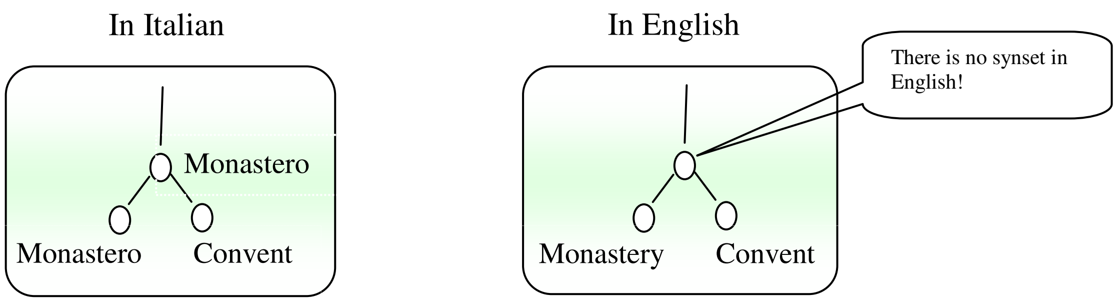


Figure 17 Gap in the language

This gap can be filled by associating a sense to the concept. More precisely, in this case we have to provide a lemma for the new word and associate to it the corresponding natural language description. We modified the abovementioned procedure as follows: CreateSense(lemma : String; gloss : String; srcconcept:Concepr; lang : Locale) It takes a reference of a concept as a parameter which is an reference to the concept that didn't have the sense in a given language but exist in another language. The pseudo-code then proceeds as follows:

CreateSynset

It creates a synset to the concept that didn't have the sense in a given language but exist in another language

lemma	= is a string;
gloss	= the natural language description of the synset;
sc	= the source concept with whom the synset will associate;
L	= the language of the new synset;
w	= the word for the new synset;
s	= the new synset which will associate with the concept;

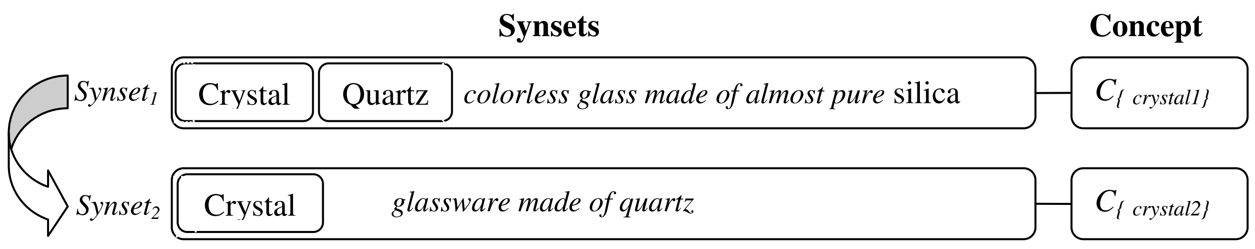
```
CreateSynset(lemma, gloss, sc, L ) {
 w :=  GetWord(lemma, L)
     if (w = null) {
       w := CreateNewWord(lemma, L); }
       s := CreateNewSynset(w ,GetPOS(sc), gloss);
       AddSenseToConcept(sc , s);
}
```

The procedure AddSynsetToConcept(concept , synset) attached the given synset to the given source concept. The parts of speech of the new synset would be the same as existing synset of the given concept. Notice that, it will create a new word in case a word for the given lemma and in the given language doesn't exist.

8.7.1.4.Update a Synset

Errors belong to life. Therefore, it is necessary to provide edit/update facilities. A synset can be updated in various ways. For example, updating the gloss of a synset or adding a word to an existing synset. In the subsequent sections we will describe the various ways a synset can be updated.

8.7.1.5.Update Synset Rank



Changes the position of the synsets thus update the synsets ranks

Conceptually, synset are associated to a word and are ordered according to their rank in ascending order, which means that the synset with rank 1 would be in the first position of the synset list, the synset with rank 2 would be in the second position and so on. Updating the rank of a synset means changing its position among the synsets associated to a word. Reducing the rank of a synset will bring it one step higher in the list and, therefore, will increase its rank value. Clearly, this operation will affect two synsets because the change will touch upon consecutive synsets. It will affect (*n-m*) synsets if the change has done between synset1 with the position *n* and synset2 with the position *m* in the list. The SetSynsetRang() procedure takes two parameters a synset and the intended rank. The given pseudo code shows how we can perform this operation:

`UpdateSynsetRank`

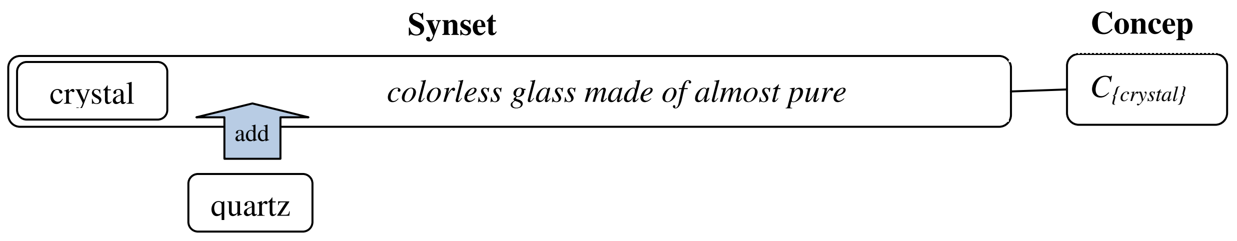
It updates synsets rank by swapping two associated synset
synset1 = the synset whose rank will be update;
synset2 = the associated synset of synset1 ;
temprank = temporary variable;

```
 UpdateSynsetRank(synset1,synset2) {
  temprank :=  GetSynsetRank(synset1)

SetSynsetRank(synset1, GetSynsetRank(synset2));
SetSynsetRank(synset1,temprank);
  }
```

The procedure GetSynsetRank(synset) returns the current rank of the given synset as parameter. The temporary variable *temprank* save the intermieadet value to avoid the overwrite during the rank swaping.

8.7.1.6. Add a word to an existing synset



A word cannot remain without links to any synset (thus concept). Therefore, to add a word the user must relate it with any existing synset. The word will be added to the list of words corresponding to that particular synset. since a synset consists of a set of words that are ranked according to their significance within a specific context, whenever adding a new word, it will display after other existing words associated to that synset.

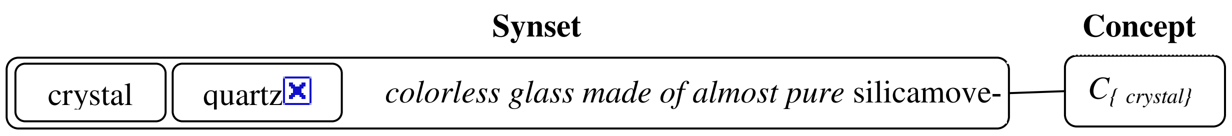
AddWord

It adds a word to a given synset

synset = the given synset;
word = the word which will be added to the given synset;

```
AddWord( word, synset) {
AddWordToSynset( word, synset)
}
```

8.7.1.7.Remove a word from a synset



Remove the word *quartz* from the synset

It is possible to remove a word from a synset. When removing a word from a synset, it will be also automatically removed from the KB if no other synsets use it. Moreover, it is necessary to control that the word's concept is not used by any domain or Entity. In case the word was the only one in the list of a synset, this latter will also be deleted. In fact, it is nonsense to have an empty synset.

RemoveWord

It removes a word from a given synset

synset = the given synset;
word = the word which will be added to the given synset;

```
RemoveWord( word, synset) {
RemoveWordFromSynset( word, synset)
}
```

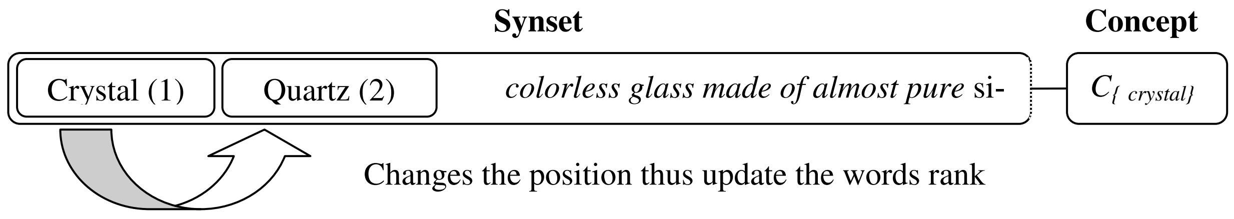
8.7.1.8.Update a Word of a Synset

The properties of a word associated to a synset are its exceptional forms and its rank. Updating any of these properties is also considered tantamount to the update of the corresponding synset:

- Exceptional forms: if changed, the changes will affect all the synsets that use the word.

Rank (within the sense): if changed, the changes will affect only the associated synset from which the word has been selected.

8.7.1.9.Update Word Rank


Synset
Concept
Crystal (1)
Quartz (2)
colorless glass made of almost pure si-
C{ crystal}
Changes the position thus update the words rank

Same as Synset rank words associated to a synset are arranged according to their rank in ascending order, which means that the word with rank 1 would be in the first position of the word list, the word with rank 2 would be in the second position and so on within the synset. Notice that, if a word is placed at the beginning of other words of that synset this word would treat as the preferred word for that synset. Decreasing a rank of a word will bring the corresponding word one step ahead in the list and will increase its rank value, while the next consecutive word will take its initial place. Clearly it will affect tow words while the change is among the consecutive words. It will affect (*n*-*m*) words if the change has done between word1 with the position *n* and word2 with the position *m* in the list. Notice that the unlike synset rank the SetWordRank() procedure takes one more parameter which is the synset associated with the word.

UpdateWordRank

It updates word rank within a synset
synset = the synset in which the word belongs to;
word1 = the word which rank will be updated;
word2 = the associated word of word1 ;
temprank = temporary variable;

```
UpdateWordRank(word1, word2,synset) {
temprank :=  GetWordRank(word2, synset)3.
SetWordRank(word2,synset, GetWordRank(word1,synset));
SetWordRank(word1,synset,temprank);
}
```

The procedure GetWordRank(word, synset) returns the current rank of the given word within the given synset. The temporary variable *temprank* save the intermieadet value to avoid the overwrite during the rank swaping.

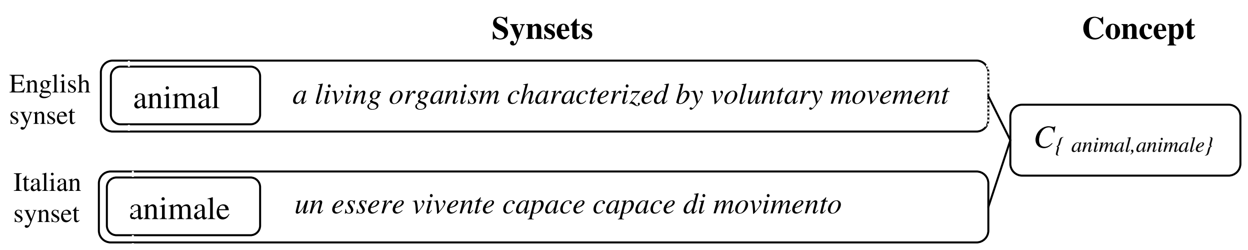
8.7.1.10. Update Gloss and POS of a Synset

Procedure for updating Gloss and POS of a synset is same as update a word of a synset, only the difference is the passing information.

8.7.1.11. Delete a Synset

Deleting a synset can be done in two ways:

(a) Deleting the synset only
(b) Deleting a synset together with its corresponding concept


Synsets
Concept
English synset
animal
a living organism characterized by voluntary movement
Italian synset
animale
un essere vivente capace capace di movimento
$C_{\{ animal,animale\}}$

The synsets of *animal* in Italian and in English

It is possible to delete a synset for a given language. However the correspondence between a synset and a concept is one to one for every specific language although for one single concept there can be multiple synsets in different languages. When deleting a synset, if it is the only one synset for a concept (considering all available languages), the deletion of that synset will also cause the deletion of the corresponding concept. However, it is impossible to delete both the synset and the concept when the concept is also used by others (i.e., domain or Entity). Note that, deleting a synset is a language dependent operation, which means it will remove only that synset of the given language. There might exist other synsets in different languages for the same concept and those will not be removed. To keep the pseudo code simple and readable we avoid the check that determine if the concept is related with any domain, entity or not.

`DeleteSynset`

It deletes a synset
synsets [] = the list of synsets (from different languages) of a given concept;
concept = the given concept;

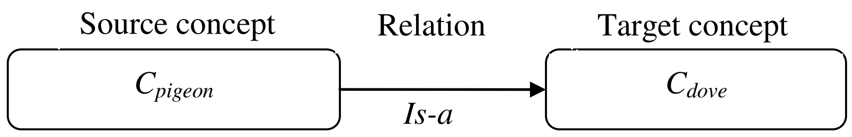
```
DeleteSynset(synset , concept , L) {
synsets[] :=  GetSynsets(concept)
if(synsets.size = 1){
    DeleteConcept(concept);
    DeleteSynset(synset, L );
    } else {
    DeleteSynset(synset, L );
   }
}
```

Notice that, the procedure GetSynsets(concept) returns all the synsets of a given concept from different languages. With the condition (*synsets*.*size=1)* we are actually checking if the given concept contains only one synset. If it contain only one synset in this case we are also deleting the concept but if the concept contain synset from other language then we delete just the synset of the given language.

8.7.2 Managing Relations

Relations are established among the concepts. Therefore, to work with relations we have to deal with concept. In the subsequent sections we describe the various operations on relation.

8.7.2.1.Linking the (concept corresponding to the) synset with an existing concept



Creating a relation *“Pigeon is a dove”*

To link a synset, or better its corresponding concept, to another concept we need to know first the target and the source concepts from the concept hierarchy and then the intended relation among them. The procedure CreateRelation would establish the given relation among the source and target concept.

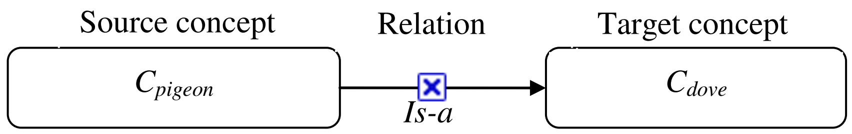
CreateRelation

It creates a relation among two concepts given as source and target

sc	= the source concept;
tc	= the target concept;
r	= the given relation;

```
CreateRelation(sc , tc,  r ) {
CreateNewRelation(sc , tc,  r );
}
```

8.7.2.2.Delete a relation



Delete the relation *“Pigeon is a dove”*

Unlike linking concepts, removing a relation also need to know just the relation object, we assume that the relation object holds the target and the source concepts from the concept hierarchy. The procedure DeleteRelation would remove the given relation among the source and target concept.

8.7.2.3.Update a relation

Update a relation means update the relation’s type only (see type of relations).Thus to update a relation we need to pass the existing relation that we want to update together with the intended relation. The UpdateRelation() procedure would replace the old relation with the new given one.

8.8. Dealing with Multiple languages

One synset can have corresponding synset(s) in other languages. Therefore, there should be a way to visualize the senses associated to a word in different languages while also maintaining the context. One intuitive way of performing this operation is expand the synset block and showing the words associated to the same senses in different languages as a separate block under that corresponding sense. To give a visual impression, we can add flags to synset blocks so to indicate its corresponding language (e.g, for Italian 🇮🇹). However, in this way, we can show only synset’s information at a time, but to get whole information in another language we need a different way to change the language for the whole interface. There are two different ways to dealing with multiple language (1) general view, in which case the change of the language would affect the whole interface (2) specific view, in which case the language change would affect only the specific part of the interface.

8.8.1 General View

One way to allow the user to select the language is to show a set of flags where each flag represent a language on the top of the system and allow the user to select any of them. However, this approach does not scale enough when languages are too many. One alternative way is to put all the supported languages by the system in a dropdown list and allow the user to select one from there. Moreover, we have to consider also one more thing that is the CUD operations. When the language will be change by the user what will be the active language for the operations. A user can change the language at any point of time during the navigation or search processes. To deal with this point we introduce the concept of "working language". If the user sets a language and initiates search for a word then the chosen language setting is considered as the *working language* of the system, and it will apply also for the subsequent CRUD operation. To indicate the system's current working language a small flag on the top of the system can be used. The effect of working language on different objects relates to several aspects:

(1) **Words**: word search results would always displayed according to the selected language. If the working language setting is "Italian" and the user searches with a keyword "come" then the system will show only those Italian words that match with "come" but not the English word "come" which means "move toward" or "arrived".

(2) **Senses**: Subsequent navigations will also be affected by the language choice. Selecting any word would show the Italian senses of that word. However, it is not guarantee that all senses will have full information, completeness of sense information as these details depend on how rich the vocabulary of that language is. In case of missing information, for example sense description we could show "undefined" or "null". Notice that not necessarily the number of senses/synsets of a word for a given language would be the same for all the languages as people that use different languages also belong to different cultures and communities, have different purposes, opinions and levels of expertise. Note also that to bootstrapping the UK and to reach a critical mass the knowledge has collected, adapted and integrated from a vast variety of different sources about different domains in different languages.

(3) **Concepts**: Selecting any sense within a certain language would show the complete concept hierarchy of that sense even though not all the languages might have the sense in the language. This is can be due to the well-known problem of gaps in languages or because of incomplete information. The concept part is language independent and thus it is always the same for all the languages. Concept with missing lexicalization would show "?" as concept label.

(4) **CUD** : the label of POS and the relation concepts would change according to the working language. Any kind of CUD operation will be based on the working language.

8.8.2 Spotlight View

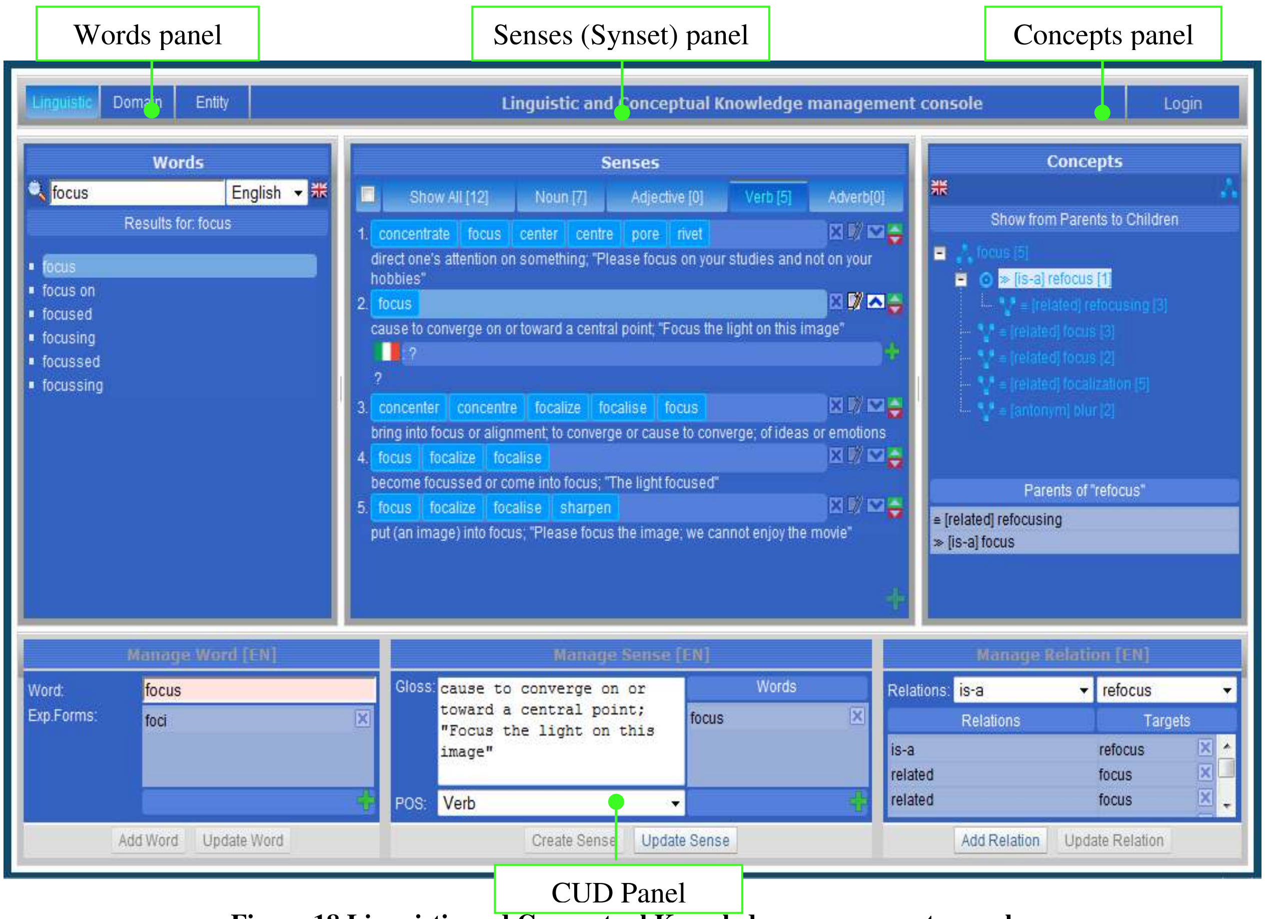
At any point of time during navigation, the user might want to see a specific part of the information in another language. For example, an Italian user who also speaks English might want to see the English description of an Italian sense. We introduced the idea of *Spotlight* view to visualize the specific part of a information (e.g., a sense in another language) in other language or languages while maintaining the context. We can do this in two different ways:

1. Selecting any specific part (e.g., synset or concept) and changing the working language. This will affect the selected object as well as those information that are linked with it. For example, if a synset is selected and we change the language, then the concept of this synset and the concept hierarchy will change according to the chosen language.

2. Visualizing the information in different languages simultaneously by expanding the sense block. In this case, the block will be expanded and the senses/synset in other languages will appear as a separate block under that corresponding sense. On the right side of each sense/synset block there can be a flag indicating the language (e.g, for Italian 🇮🇹) of the corresponding block.

8.9. Design Principle and Implementation of the console

The console is divided into two different parts , this grouping we made according to the kind of operations , precisely : 1) "Search and navigation" and 2) "CUD (create/update/delete)" part. Further, the "Search and navigation" part is divided into three different panels namely: (a) Words panel (b) Synsets panel and (c) Concepts panel. Each part is responsible for a set of related functionalities. In visualization theory chapter we have illustrated how parts of a whole working together to achieve a specific goal. From operational point of view "Search and navigation" and CUD operations are different. Therefore it is essential to supply clear visual clues to identify the separation among them and understand which interface elements are related to one another so that the end user of the application can promptly identify organization and make sense of how to use or interact with it. The principal of *Proximity* and *Similarity* have obvious relevance to the layout design. According to *Proximity* principal objects that are close to one another are tend to be perceived as group , we arranged all the navigation panels closer together horizontally and all the CUD related panels closer together horizontally on the bottom. This particular arrangement of panels gives us an impression as to which interface panels are related to one another. To make the user visual communication even stronger towards the grouping we applied the principle of *similarity*. According to the *Similarity* principle there are many ways (e.g., color, size, shape, dimension, texture, and orientation etc) to make objects to be perceived as being related. In our design to make the search and navigation parts similar therefore related, and so the CUD parts, we applied the *size* property, in particular the height of the panels. We make the height of the entire search and navigation panels the same. In the same way we make the height of all the panels dedicated to CUD are same. The combination of these two properties enables us to make a clear separation between search and navigation. The peculiarity of this prototype is that it is very "space efficient" while maintaining the navigation context. In addition to structural clarity, this design provides simple navigational structure to explore the challenging manifestation of the complexities of linguistics and concept knowledge.



Words: focus — English

Results for: focus

- focus
- focus on
- focused
- focusing
- focussed
- focussing

1. concentrate | focus | center | centre | pore | rivet
direct one's attention on something; "Please focus on your studies and not on your hobbies"
2. focus
cause to converge on or toward a central point; "Focus the light on this image"
?
?
3. concenter | concentre | focalize | focalise | focus
bring into focus or alignment; to converge or cause to converge; of ideas or emotions
4. focus | focalize | focalise
become focussed or come into focus; "The light focused"
5. focus | focalize | focalise | sharpen
put (an image) into focus; "Please focus the image; we cannot enjoy the movie"

- focus [5]
  - » [is-a] refocus [1]
    - ≡ [related] refocusing [3]
  - ≡ [related] focus [3]
  - ≡ [related] focus [2]
  - ≡ [related] focalization [5]
  - ≡ [antonym] blur [2]

Parents of "refocus"

≡ [related] refocusing
» [is-a] focus

Word: focus
Exp.Forms: foci

Gloss: cause to converge on or toward a central point; "Focus the light on this image"
POS: Verb
Words: focus

Relations: is-a — refocus

Relations	Targets
is-a	refocus
related	focus
related	focus

Figure 18 Linguistic and Conceptual Knowledge management console

(1) **Search and navigation**: The upper part of the console is dedicated to search and navigation, and consists of three different panels. They are grouped by headings (i.e., “Words”, “Sense” and “Concepts”). All of this reinforces the relationships of the three groups of information, group by object (i.e., word, sense, and concept). *Proximity* is used to indicate grouping.

   a. *Words panel*: this panel provides the facilities for word search, language selection, autosuggestions, and a search result list based on the searched word. Search result list is used to navigate through the words and provide a means for selecting query scope.

   b. *Sense (Synsets) panel*[15]: The result of this panel depends on the search word in *word panel*. In general, it will show a list of synsets for the word that has been searched. Results are shown grouped by POS.

   c. *Concept panel*: result of this panel depends on the synset selection from the synset panel. Concepts and corresponding relations are shown as a DAG based on the selected generalities.

(2) **CUD-Create/Update/Delete**: the lower part of the console is dedicated to CUD operations on words, senses ,synsets, concepts and relations between them. They are grouped by headings (i.e., “Manage Word”, “Manage Sense” and “Manage Relation”), and the fields themselves are arranged vertically, with the left sides of the field aligned with one another. All of this reinforces the relationships of the three groups of information, group by object (i.e., word, sense, relation). *Proximity* is used to indicate grouping. By aligning three operational parts to a common axis we also ensure the *visual unity* that would support each other and all work together toward a common goal.

In the following sections, we describe in detail the single functionalities.

8.10. Detailed description of the Search and Navigation Panels

8.10.1 Words panel

This panel provides the facilities for word search and filtering based on language selection, autosuggestions, and a sorted search result list.

[15] In this document we will use the term Synset panel instead of Sense panel. But in the UI we sued Sense because we believe it is more intuitive for the general users.

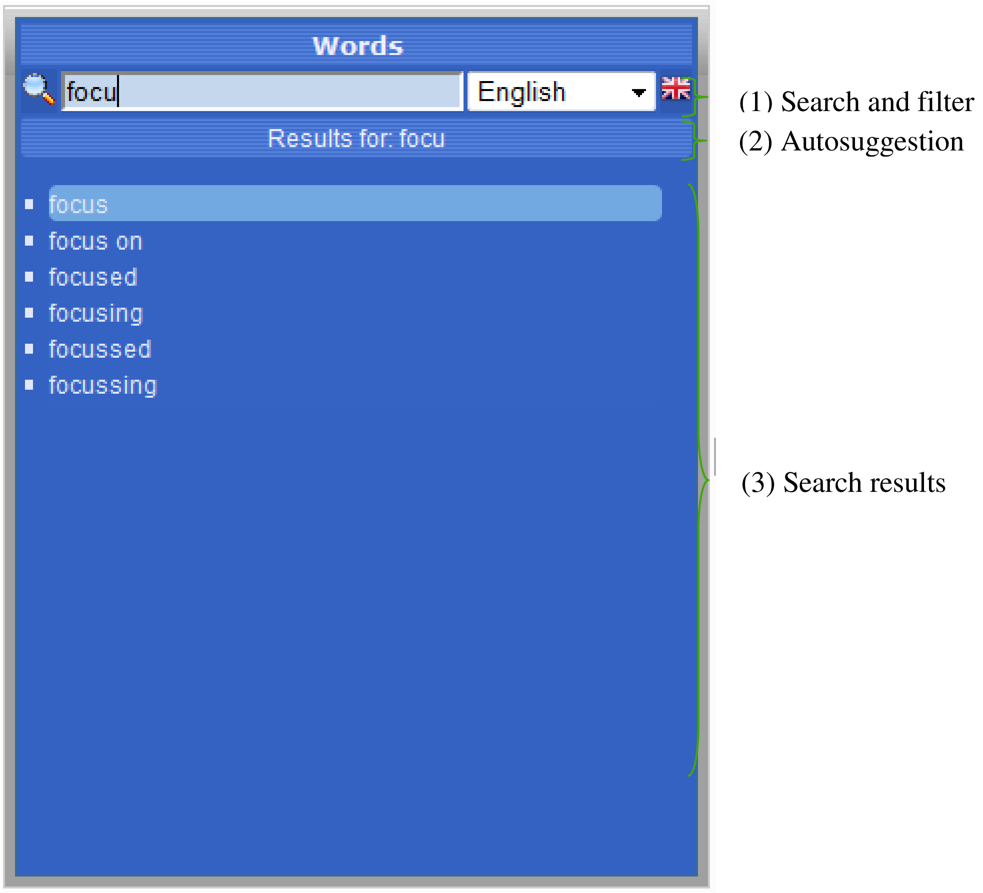


Figure 19 Words Panel

(1) **Search and filtering** input panel consists of the following components:

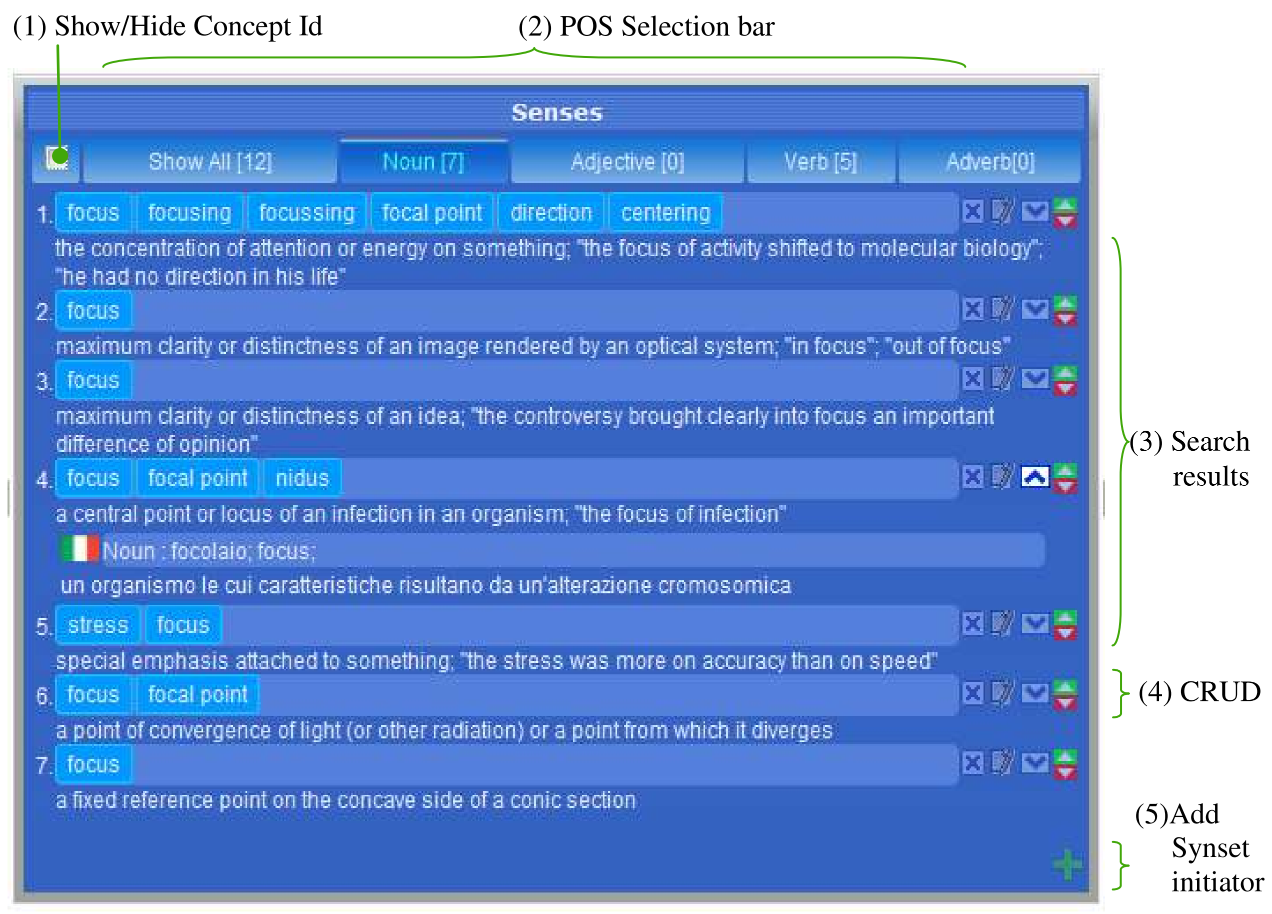
1. *Language selection combo*: A dropdown list box for selecting preferred language. When the user selects a different language, all information in the UI remains the same but it is shown in the selected language.
2. *A text box for free text word input*: figure 19 shows a situation in which a user is typing a word in the word input text box. After typing a word and the user presses the enter key, two possible scenarios could take place:
    - The word exists: a list of results will appear in the search result list box in alphabetical order together with the searched word; The user can select a word from the list. Synset(s) for that selected word would appear as a list in the synset panel for that word.
    - The word does not exist: it can occur due to misspelling or language setting or simply because the word does not exist in the KB. In this case, autosuggestion panel will show a suggestion.

(2) **Autosuggestion** shows a suggestion in the form of *"Try this? xxx"*. For example, if the user typed "architecher" which was misspelled, thus system could suggest *Try this? "architecture"*.

(3) **Search result**: Search result is shown as a list in the search result list box in alphabetical order together with the searched Word.

8.10.2 Senses panel

Displaying information to be easy to search and navigate, it is not enough simply to make them brief, structured, and no repetitious. It is also necessary to conform to the rules of graphic design, some of which were presented in visualization theory chapter. Senses panel consists of a set of ***POS selection buttons*** (e.g., noun, adjective, verb, adverb). This presentation in turns employs the tabbed navigation that conform the *principle of uniform connectedness* of the synsets of a word to show context. The list for presenting search results consists of a set of blocks, each block representing a sense/synset. Notice that, by presenting synset as a block we are actually giving a structure to the information. Each block contains a set of buttons to delete or update the sense/synset and to expand the block in order to visualize complete information. Finally, at the bottom of the result list, a plus button can be used to add a new sense/synset.



1. focus | focusing | focussing | focal point | direction | centering
the concentration of attention or energy on something; "the focus of activity shifted to molecular biology"; "he had no direction in his life"

2. focus
maximum clarity or distinctness of an image rendered by an optical system; "in focus"; "out of focus"

3. focus
maximum clarity or distinctness of an idea; "the controversy brought clearly into focus an important difference of opinion"

4. focus | focal point | nidus
a central point or locus of an infection in an organism; "the focus of infection"
Noun : focolaio; focus;
un organismo le cui caratteristiche risultano da un'alterazione cromosomica

5. stress | focus
special emphasis attached to something; "the stress was more on accuracy than on speed"

6. focus | focal point
a point of convergence of light (or other radiation) or a point from which it diverges

7. focus
a fixed reference point on the concave side of a conic section

Figure 20 Senses Panel

(1) Show/Hide Concept Id: Selecting and deselecting this checkbox would show and hide the UK concept id of the corresponding sense.

(2) **POS Selection**: The user can filter the result based on parts of speech by clicking on any tab for POS. A tab view provides a convenient way to present information in a multipane format. The tab control is displayed horizontally centered across the top edge of a content area. Users click a tab to see the content associated with it. Each tab also shows the number of synsets for every POS inside the brackets.

(3) **Sense/Synset search result**: Displaying search outcome in a concise, structured way and avoiding recurring clatter can get better people's capability to search swiftly and find what they look for. In our case we present search result as a list representing the grouped of synsets (grouped according to the selected POS). Results are presented as blocks of information in a linear list. Each block represents a sense/synset, Initially each block contains the word together with its synonyms and corresponding gloss. In this design we imposed the unity rule. *Unity* in design can be achieved through many ways in layout that can also be achieved through applying margin and padding to the elements. One more way to use unity in design is to divide text into groups using headlines. The headline with some visual contrast and by grouping it with the description below it is clearly related to that content. In our case headlines contain the set of words of a synset together with some control buttons. Presenting gloss of the synset below the headline clearly relate it with the synset. User can further navigate to visualize it in different languages by clicking on the down-

arrow button attached with the sense/synset block. Clicking on the down arrow will expand the block and the words associated to the same senses in different languages (see Figure 21 Sense block with multi language) will appear as a separate block under that corresponding sense. On the right side of each sense block there is a flag that indicate the language (e.g, for Italian) of the corresponding sense/synset block.

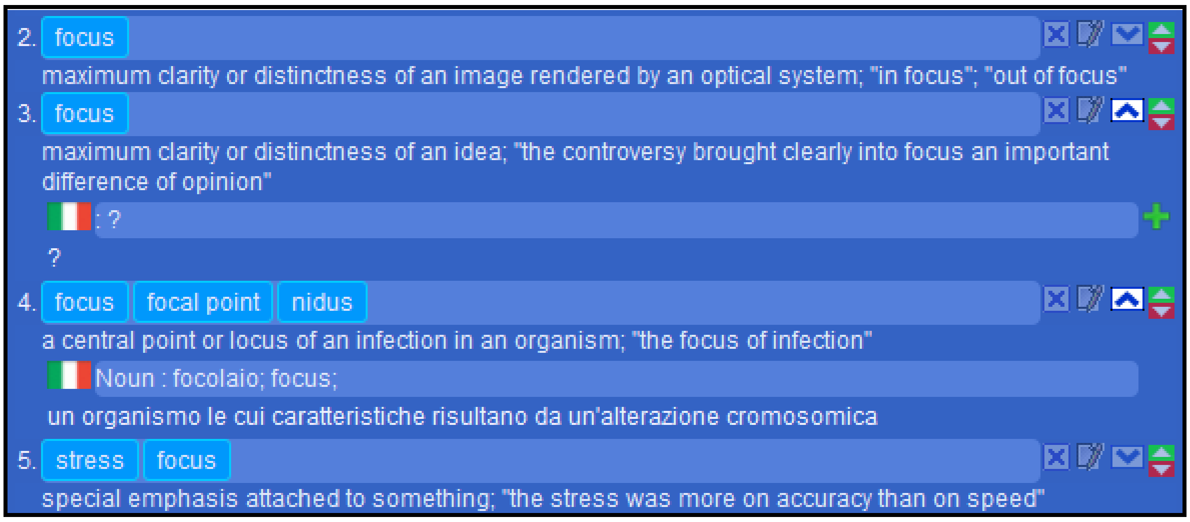

2. focus
maximum clarity or distinctness of an image rendered by an optical system; "in focus"; "out of focus"
3. focus
maximum clarity or distinctness of an idea; "the controversy brought clearly into focus an important difference of opinion"
: ?
?
4. focus | focal point | nidus
a central point or locus of an infection in an organism; "the focus of infection"
Noun : focolaio; focus;
un organismo le cui caratteristiche risultano da un'alterazione cromosomica
5. stress | focus
special emphasis attached to something; "the stress was more on accuracy than on speed"

Figure 21 Sense block with multi language

(4) **CUD Operations on a Sense block**: user can delete or update a sense/synset by clicking on and respectively. Note that, the operations will be applicable only on the "selected" sense/synset from the Sense panel. Each block also has a down-arrow button and a up-arrow button on the right corner. These two buttons are used to change the sense rank of a sense/synset w.r.t the word selected in the word panel. The position of a word within a sense/synset (from right to left) represents the rank of the word within the sense/synset. Each word within a sense/synset is moveable; user can modify the rank of a word within the sense/synset by modifying the position (for rank details please see the section 8.14, Manage Rank).

(5) **Adding a new Sense/Synset**: In the bottom of the synsets panel there is an add button which is for adding a new synset. This functionality is described in detail in Section 6. Note that, if a multilingua block is open and the corresponding sense doesn't exist in any of the languages then the block will appear with a button (Figure 21, 3rd Sense block), In this case the user can add the missing sense by clicking on that the button of the corresponding language.

8.10.3 Concepts panel

Provide a visual hierarchy is one of the main goals in structuring information presentations. Presenting data as in graph is one way of structuring information. However it is well known that understanding and comprehensive analysis of data in graph structures is easiest if the size of the presented graph is small. Therefore we employ an incremental browsing method of the information space structure and visualizes in every moment only specific part of the information where its content depend on previous action. The Concepts panel (figure 22) displays the resulting concept hierarchy based on the selection of a sense/synset from the Senses panel instead of showing the full concept hierarchy. This panel consists of a **Tree** and a **Parent/Children List** to visualize and navigate the resulting concept hierarchy as a DAG (directed acyclic graph) based on the sense/synset selection from the Senses panel. To keep the presented graph size small tree is generated for different levels of depth. Initially it will show the Level ”1” which means that only direct children/parents concepts that are directly related to the root concept are represented. In this visualization a user can navigate through the graph and expand each concept by clicking on them to reach the second level and so on.

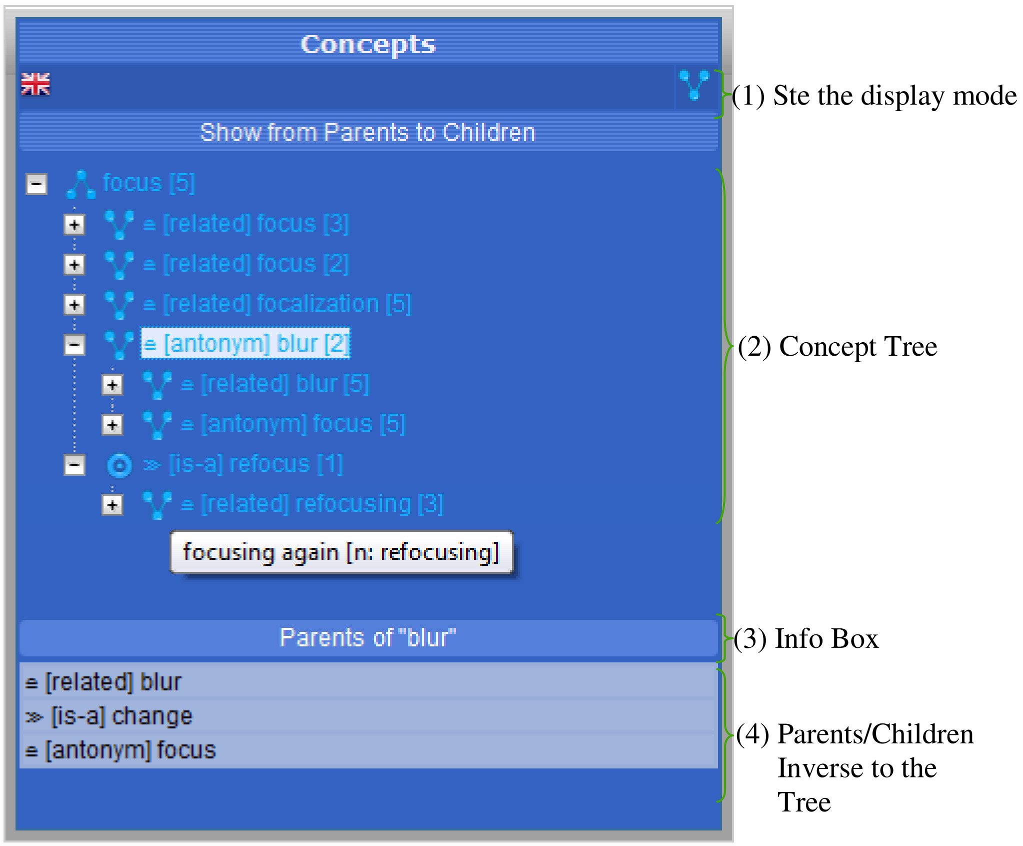


Figure 22 Concepts Panel

- **Display mode:** The button on the right of the top most panel allows setting the overall display mode of the concepts visualization. There are two display modes:

  : Show from parents to children
  : Show from children to parents

- **The concept hierarchy**: The panel used to navigate the concept hierarchy as a DAG. Concept tree allows navigating through the concepts. If "Show from parents to children" is selected by clicking on , then the root will be treated as a parent concept which is also the "selected" concept and its children will be shown as subsequent child(s). In the same way, if "Show from children to parents" is selected by clicking on, then the root will be treated as a child concept which is also the "selected" concept; its parent(s) will be shown as subsequent child(s). Notice that the kind of each relation (from source to target) is shown in squared brackets (e.g. [is a], [part of], [see also]).
- **Parent/Children concept(s):** This list box will show the parent/children concept(s) of the selected concept. By clicking on any concept from the concept tree, the panel will show the list of parents or children (if any) of the selected concept depending on the *Display mode* selection(inverse of Tree). The heading of this list will be changed dynamically depending on the mode selection (e.g, either "Parents" or "Children").Clicking on any parent/children from this list of concepts will change the concept hierarchy of the concept panel and will show the sub tree of that selected parent/children. This allows navigating DAGs in a natural way, always showing a (partial) tree view.
- **Concept description:** A tooltip shows the description/gloss of the selected concept. Moving the mouse over any concept will show the gloss of that selected concept.

8.10.3.1. Navigating Concept Hierarchy using keyboard

Action	*Key*
Navigate to the next sibling	Down arrow
Navigate to the previous sibling	Up arrow
Open a subtree	Right arrow
Close a subtree	Left arrow
Navigate to open subtree	Right arrow
Navigate to parent	Left arrow
Activate a tree item	Enter
Navigate to first tree node	Home
Navigate to last visible tree node	End

Table 3 Keyboard Navigation

8.10.3.2. Visualization details of concept hierarchy

The concept hierarchy is represented as a Tree[see figure.22]. Any concept is visually represented as a tree node. *Nodes* are visually represented with an open/collapse icon, for opening and collapsing the node. It also has a “has multiple parents”/”has multiple children” symbol used to show if the concept has more than one parents or children depending on display mode setting, as relation kind symbol informs about the relation between them visually followed by textual label, a concept label and ended with a number saying the number of children/parents if any. Furthermore placing mouse cursor on any concept node will show the description of that concept together with its synonyms and parts of speech. In general, data in the tree node are represented as follows:

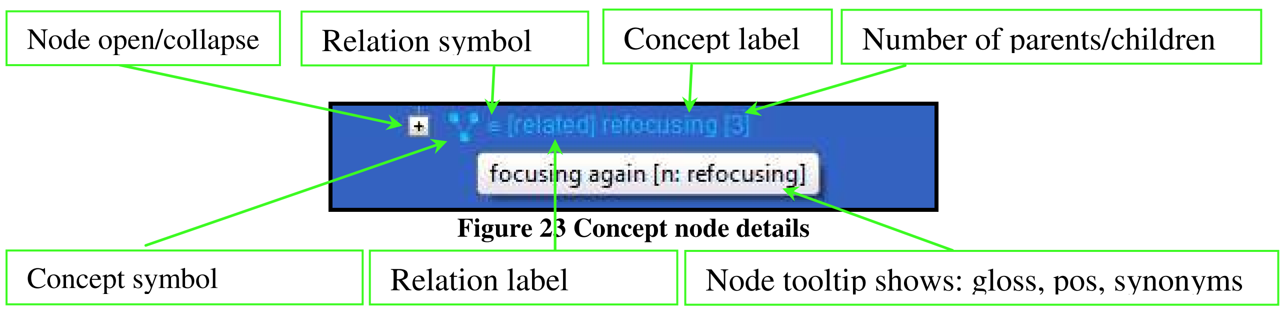


Figure 23 Concept node details

We show the relation in textual form followed by a concept label. For example to represent the concept “Dog” with the relation “is-a” we use: >> [is a] Dog”. In the same way, assuming dog has multiple parents, to represent the DAG, we can use : ⊞ >> [is a] Dog. Notice that either “<<” or “>>” is used according to the selection mode.

Concept node description: The following symbols are use to describe a concept node:

1. **Concept symbol**: a concept node represent using the following symbols:
   a. ⊙ : represent a concept with single parent/child depending on the *display mode* setting. A concept node will display ⊙ icon in the following situations:
      i. if the display mode is “Show from parents to children” and the intended node has just one parent
      ii. if the display mode is “Show from children to parents” and the intended node has just one child

b. : represent a concept with multiple children, when the display mode is “Show from children to parents” and the concept node has more than one child

c. : represent a concept with multiple parents, when the display mode is “Show from parents to children” and the concept node has more than one parents

2. **Concept Label**: The label of the concept or question mark (?) if the concept doesn’t have any label in the given language.
3. **Relation symbol:** relation between two nodes are represent using the following symbols:
   a. ≘ : represent concept’s associative relation with its parent node(see section 4.2.2)
   b. << , >>, <, > : represent concept hierarchical relation together with direction with its parent node(see section 8.8.3).
   c. [+]/[-]: node open/close icon
4. **Relation Label**: The natural language label of the relation (e.g., is-a, part-of, related Chapter 4 for complete list of relations).
5. **Number of Parents/Children:** At the end of each node there is a number which represent the number of Parents/Children (if any) of that node depending on the display mode. For example if the display mode is “Show from parents to children” and the concept node contain 3 child, then this ending number would be [3] and vice versa according to the display mode setting.
6. **Open/Close icon:** each node precede with a open/close **[+]/[-]** icon that shows if the node is open or close.
7. **Node Tooltip:** placing cursor on any node would show a tooltip which present then following information of that node concept:
   a. Gloss: Natural language description of that concept.
   b. Pos: parts of speech of that concept.
   c. Synonyms : synonymous words of that that concept.

8.10.3.3. Visualization of concept relations

There are **different kinds of relations** - including hierarchical (is-a, part-of) and associative ones (e.g. member, see also)._Different kinds of relations can be emphasized using the following icons[16] symbols (codifying also the direction of the relation):

	Show from parents to children		**Show from children to parents**	
	Relation Kind	Symbol	Relation Kind	Symbol
Hierarchical relations	Is a	>>	Is a	<<
	part of	>>	part of	<<
Associative relations	member-of	>	member-of	<
	substance-of	>	substance-of	<
	for all the others relations	≘	for all the others relations	≘

[16] http://www.destin.be/ASKOSI/Wiki.jsp?page=Icons%20for%20SKOS

(4) Notice that the symbols above are the standard Unicode characters for brother and narrower semantic relations, frequently used to display relations in SKOS.

8.11. Dealing with Multiple languages

There are two different ways to dealing with multiple language (1) general view, in which case the change of the language would affect the whole interface (2) specific view, in which case the language change would affect only the specific part of the interface.

8.11.1 General View:

The words panel (see Figure 19 Words Panel) contains a dropdown list that contains a list of all the supported languages by the system. The default setting is English. If the user sets a language and initiates search in the words panel then this setting of the language is considered as the *working language* of the system, because it is applied also for the CRUD operation, In other words we can say that any kind of CUD operation in this language. A small flag just beside the language dropdown list indicates the current working language. However, at any time a user can change the working language by selecting a language from the list followed by a "New Search". The effect of working language in different panel would be as follows:

a. **Words panel**: word search results are always displayed according to the selected language. If the working language setting is "Italian" and the user searches with a keyword "come" then the system will show only those Italian words that match with "come" but not the English word "come" which means "move toward" or "arrived".

b. **Senses panel**: Subsequent navigations will also get affected in the same way as search. Clicking on any word would show the Italian senses of that word. However, it is not guarantee that all senses will have full information, completeness of sense information depends on how much rich the vocabulary of that language. In case of missing information, for example sense description the panel will show "undefined" or "null". Notice that not necessarily the number of senses/synsets of a word for a given language would be the same for all the languages as people use different languages belong to different cultures and communities, have different purposes, opinions and levels of expertise. Note also that to bootstrapping the UK and to reach a critical mass the knowledge has collected, adapted and integrated from a vast variety of different sources about different domains in different languages

c. **Concepts panel**: Selecting any sense from the sense panel would show the complete concept hierarchy of that sense even though not all the languages might have the sense in the language. This is can be due to the well-known problem of gaps in languages or because of incomplete information. The concept part is language independent and thus it is always the same for all the languages. Concept with missing lexicalization would show "?" as concept label.

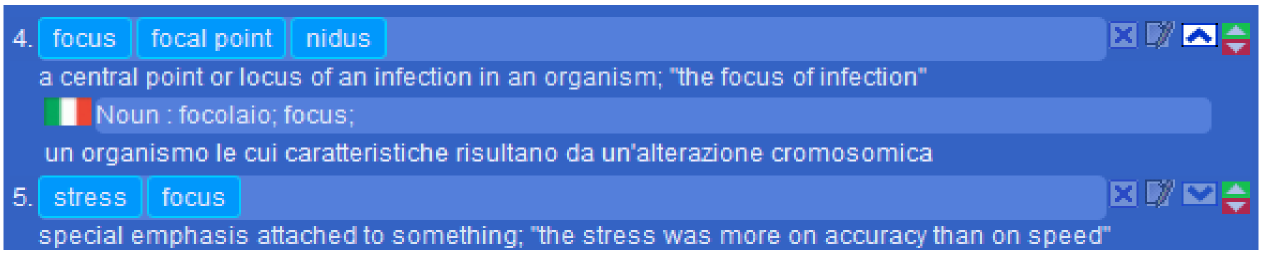
d. CUD panel: the label of POS and the relation concepts would change according to the working language. Any kind of CUD operation will be based on the working language.

8.11.2 Spotlight View:

Spotlight view provides a way to view the specific part of the information (e.g., a sense in another language) in another language or languages. There are two different ways to do this:

3. At any point of time during navigation, the user can select any sense/synset or concept and change the working language from the language dropdown list. It will show the corresponding senses in the selected language together with its concept hierarchy.

4. The user can also visualize information in different languages simultaneously by clicking on the down-arrow ⌄ button attached with the sense block. In this case, the block will be expanded and the senses/synset in other languages (see Figure 24 Multilanguge Selection) will appear as a separate block under that corresponding sense. On the right side of each sense/synset block there is a flag that indicates the language (e.g, for Italian 🇮🇹) of the corresponding block.



4. focus | focal point | nidus

a central point or locus of an infection in an organism; "the focus of infection"

Noun : focolaio; focus;

un organismo le cui caratteristiche risultano da un'alterazione cromosomica

5. stress | focus

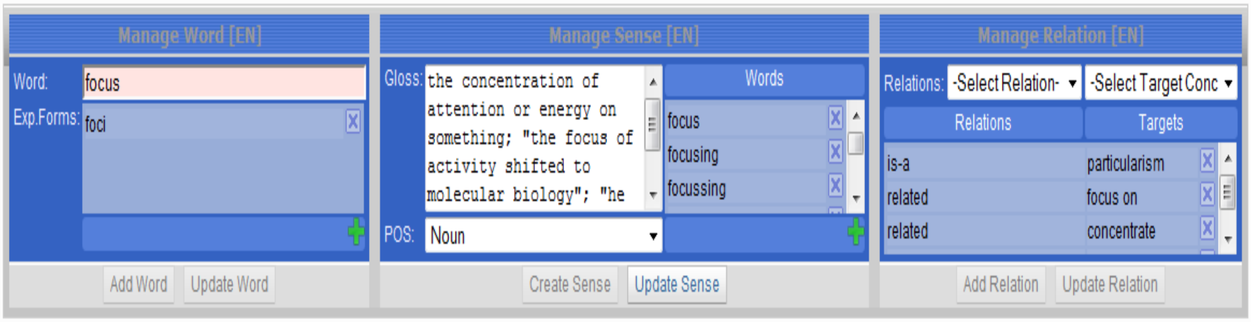
special emphasis attached to something; "the stress was more on accuracy than on speed"

Figure 24 Multilanguge Selection

8.12. Detailed description of the CUD Panel

CUD panel is dedicated for any sort of add, delete or update operations on words, synsets, concepts and relations between them. To work on any existing word or synset the user has to select a synset block from the synsets panel and initiate the desired operation by clicking on the appropriate button provided there.

For example, selecting the 1st sense/synset for "*focus*" from the senses panel (see Figure 24 Multilanguge Selection) and by clicking on the update button, it would show all its information in the CUD panel (see Figure 25 CUD Panel), including the facilities for managing relations for the corresponding concept. Alternatively, the user can also add a new sense/synset using the button on the bottom of the *senses panel*.



Manage Word [EN]

Word: focus

Exp.Forms: foci

Add Word | Update Word

Manage Sense [EN]

Gloss: the concentration of attention or energy on something; "the focus of activity shifted to molecular biology"; "he

Words
focus
focusing
focussing

POS: Noun

Create Sense | Update Sense

Manage Relation [EN]

Relations: -Select Relation- | -Select Target Conc

Relations	Targets
is-a	particularism
related	focus on
related	concentrate

Add Relation | Update Relation

Figure 25 CUD Panel

The overall CUD panel is divided in to three segments, namely "Manage Word", "Manage Sense" and "Manage Relation". As it can be noticed, there is a perfect symmetry between this panel and the navigation panel above.

8.12.1 Manage Word:

The Manage word part of the CUD panel is dedicated to the management of the information related to words. It consists of the following components:

c. **Word**: a text field for entering a word

d. **Exceptional Forms**: a list box that presents the existing exceptional forms associated to the word followed by a button that enables the user to add new exceptional forms to the word. Note that clicking on this plus button will open a new popup window for adding exceptional forms. The user can delete an exceptional form for the word by using the button attached with each exceptional form.

e. Finally two buttons: "Add word" and "Update word"

8.12.2 Manage Sense:

The middle part of the CUD panel is to manage sense/synset. It is dedicated to the management of the information directly related to the sense/synset. It consists of the following components:

d. **Gloss**: a text area for entering the gloss (the natural language description of the sense/synset)

e. **POS**: a dropdown combo box for selecting parts of speech; possible values are "Noun", "Adjective" , "Verb" and "Adverb".

f. **Words**: a list box showing the set of words associated to the sense/synset; the user can add or delete a word from the sense/synset using the ➕ and ☒ button respectively.

g. Finally two buttons: “Update Sense” and “Delete Sense”

8.12.3 Manage Relation:

The right most part of the CUD panel is the manage relation. Relation related operations can be done from here. This part consists of the following components:

d. **Relation**: A dropdown list for selecting a relation kind. Possible relations can be hierarchical (e.g., is-a, part-of) and associative.

e. **Target Concept**: A dropdown list for selecting the *target concept*; This list can have value of the selected concept’s label, if any concept is already selected from the concept hierarchy.

f. **Relation-Target table**: this table contains a list of targets of the selected sense/synset together with corresponding relations.

g. Finally two buttons: “Add Relation” and “Update Relation”

8.13. Managing Words and Senses/Synsets

It is important to underline that for us the notion of synset is central. The user can manage words and concepts only through a synset. In particular, the user can perform the following operations on words and synsets:

7. ***Create a Synset***: There are three different scenarios in which a user can create a synset:

   a. Create a synset (and corresponding concept) as child of an existing concept

   b. Create a synset to be associated with a concept with no synset in the language

   c. Create a synset as a new root concept

   In all the cases above the synset is created by specifying its first word.

8. ***Update a Synset***: Updating a synset can be done in the following ways:

   a. By adding a word to the synset

   b. By removing a word from the synset

   c. By updating the proprieties of a word in the synset

   d. By changing gloss and POS of the synset

9. ***Delete a Synset***: Deleting a synset can be done in two ways:

   a. Deleting the synset and keeping the concept

   b. Deleting the synset together with the corresponding concept

In the subsequent sections we will describe all these possible operations in details.

8.13.1 Create a synset (and corresponding concept) as child of an existing concept

The user can create a synset by adding a word together with a new synset. In fact, it does not make any sense to have a word without any synset and vice versa. In this case the user is asked to provide the information of the word (e.g. its exceptional forms) together with the information needed for the synset (e.g., gloss and POS).

(1) **Information to provide:**

a. Mandatory information for the word:

   - Word: the user types the word he wanted to add

- Is preferred term: the first word of the synset will be automatically selected as preferred term.

b. Optional information for the word:
   - Exceptional forms: The derived forms of the word

c. Mandatory information for the synset:
   - POS: the Part Of Speech of the synset. It can be noun, adjective, adverb or verb.
   - Gloss: the natural language description of the synset
   - Target concept: The target concept to link to (the concept corresponding to) the sense must be selected from the Concept panel. If no target concept is selected it would create an isolated sense/concept.

(2) **Operation description:**

The user first clicks on the add synset ➕ button located on the bottom of the synsets panel. Clicking on add synset ➕ button will open the CUD panel with blank information. Only the **Add word** button of the CUD panel will be enabled. All other buttons will be disabled, while the input fields from the others parts of the CUD panel (e.g., manage synset, manage relations) will remain enabled. The user types the word in the word input box and presses enter. If the word already exists the exceptional forms related to that word (if any) will automatically appear in the CUD panel. If the word does not exist then it is created, by preliminary asking the user for a confirmation (to avoid cases of misspellings). During the operation, the following information must be provided:

- Word: The user types the word in the word input box.
- Exceptional forms (if any): to add exceptional forms, user clicks on the add Exc Form ➕ button located on the bottom of the Manage Word panel. Clicking on add ➕ button will open the pop up dialog to add the Exceptional form.
- Rank: by default the rank is 1 if it's a isolated synset otherwise it would add at the end of the existing synset list.
- Is preferred term: the user can select the word as "preferred term" after creation, by placing the synset at the 1st positing. However there must always be exactly one preferred word in any synset. Therefore, if it's the first synset it will be automatically preferred word.
- Gloss: the natural language description of the synset.
- POS: the part of speech of the words in the synset.
- Relation: to add a relation with an existing synset/concept, the user has to browse the concept hierarchy and select the appropriate target concept. For details on how to manage the relations, please see the section 8.1.
- Finally, Add word button will be pressed for final commit.

(3) **Post state change of the UI:** after pressing the Add button the following changes will take place:
   - CUD panel will be refreshed. The new word will appear as the unique word in the list of words for the synset.
   - Synsets panel will be refreshed. The list of synsets will refer to the preferred word of the synset created. The new synset will appear as the selected synset block in the synsets panel.
   - A new concept will appear as child of node of the target concept in the concept panel. This new concept will be the selected one.

8.13.2 Create a synset to be associated with a concept with no synset in the language

Due to gap in languages or because of incomplete information, some concepts in the hierarchy might not have a corresponding synset for the selected language. The user might want to fill the gap (see section 8.5.1.3) by associating a synset to it. The user should then proceed as follows:

(1) **Operation description:**

- The user selects the concept with no label from the concept panel
- The user clicks on the add synset button ✚ from the bottom of the synset panel. A message box will be appear and will ask the user weather he wants to create a synset for this concept or a new concept as child of this concept. If the first option is chosen then do as described in below, otherwise do as described in 8.11.1 (notice that in the latter case the label of the target concept in the CUD Manage Relation field will be blank).
- Look at the other languages to understand what the concept is about. The user can do this just by changing the language from the Word panel. When the user selects a different language, all the information in the UI remains the same but it is shown in the selected language. If the user finds any concept in any other language then he can get some idea on the synset information in the language he is working with.
- Fill up the information for word and synset as for the create synset operation described in section 8.1.
- User click on add word button of the CUD panel.

(2) **Post state change of the UI:** after pressing Add button the following changes will took place:

- The language setting will be changed back to the previous one to the language that was selected when the ✚ button has been clicked.
- The whole UI will be refreshed as described in section 8.13.1
- The new label of the concept will be the label of the “preferred term” label of the synset. The label will appear for the concept.

8.13.3 Create a synset as a new root concept:

In the case the user wants to create a new synset corresponding to a root concept, he should proceed as follows:

(3) **Operation description:** To add a new synset associated with a new root concept:

- The user clicks on the add synset button ✚ in the synset panel.
- Fill up the information described as create synset (see section 8.13.1)
- The user doesn’t need to selects any target concept from the target concept dropdown list.
- The user clicks on add word button from the CUD panel.

(4) **Post state change of the UI:** after pressing the Add button the following changes will took place:

- The whole UI will be refreshed as described in section 8.13.1
- The new Concept will appear as a root concept in the concept panel.

8.13.4 Update a Synset:

User can update a synset in various ways. For example adding a word to an existing synset also updates the synset. In the subsequent sections we will describe the various ways a synset can be updated.

8.13.4.1. Add a word to an existing synset

Users are not allowed to add a “free word”. Here by the phrase “free word” we mean “a word without linking to any synset (thus concept)”. So to add a word the user must relate the word with any existing synset. The word will be added to the list of words of that particular synset.

(1) **Information to provide:** Before adding a word the user must select a synset from the synsets panel.

a. Mandatory information:

- Word: the user types the word he wanted to add

b. Optional information:

- Exceptional forms: The derived forms of the word

(2) **Operation description:** To add a word to an existing synset:

- The user first selects a synset from the synsets panel in which he wanted to add the word.
- The user clicks on the edit synset button from the bottom of the synsets panel. As a consequence, the CUD panel will be initialized with the information of the selected synset. List of existing word(s) will appear in the words list box of the “manage synset” segment.
- The user clicks on the add word button from the words list box in the “Manage Synset” panel which will activate the manage word panel on the left of the CUD (see Figure 25 CUD Panel) with blank values. Only the **Add Word** button of the CUD panel will remain enabled. The user types the word in the word text box and presses enter. If the word already exists (in any other synset) the exceptional forms related to that word will be automatically shown. If it does not exist then the user is asked to confirm the creation of the new word (as described in Section 8.13.1). The user fills the remaining information. Notice that by default the word would add at the end of other existing words of that synset after creation, the user can change its position(i.e., Rank) by drag and drop to the intended position, if the word is placed at the beginning of other words of that sense the this word would treated as preferred word for that sense
- Finally, the Add button will be pressed to final commit the operation.

(3) **Post state change of the UI:** after pressing the Add button the following changes will took place:

- CUD panel will be refreshed. The new word will appear in the words list. If the word was set as the preferred one, it will appear as the first one in the list
- Synsets panel will be refreshed. The new word will appear in the selected synset block.

8.13.4.2. Remove a word from a synset

It is possible to remove a word from a synset by acting on the words list of the CUD panel. However, to remove a word from a synset the user first has to select the synset from the synsets panel and click on the update synset button to open the CUD panel. The information for the synset will appear. The list of words for that synset will appear in the words list box. Mouse hovering on the words list will show for each word. To remove a word the user can just click on the button of any word.

(1) **Operation description:** Clicking on button will do the following:

- Remove the corresponding word from the selected synset
- The word will be automatically removed from the KB in the case no other synsets use it.
- In case the word was the only one in the list, the synset is deleted. In fact, it does not make any sense having an empty synset.

(2) **Post state change of the UI:** after pressing delete button the following changes will took place:

- CUD panel will be refreshed by removing the word from the words list
- Synsets panel will be refreshed by removing that word from the selected synset block
- In case the synset is deleted, the synset panel will be refreshed by removing the synset block and the CUD panel will be disabled.

8.13.4.3. Update a Word of a Synset

To update the properties of a word associated to a synset, the user first has to select the synset from the synsets panel and click on the update synset button to open the CUD panel. The information for the synset will appear. The list of words for that synset will appear in the words list box. The preferred word will appear in the word input box together with its information. The user can select any other word from the words list that he wants to modify. The user can change the following information of a word:

- Exceptional forms: if changed, the changes will affect all the synsets which use the word.
- Rank (in the sense): user can change its position (i.e., rank) by drag and drop to the intended position.
- Is preferred term (in the synset) : if the word is placed at the beginning of other words of that sense the this word would treated as preferred word for that sense

Notice that the word field will be not editable. Finally, the user presses the Update word button to commit the changes.

8.13.4.4. Update Gloss and POS of a Synset

Procedure for updating Gloss and POS of a synset is same as update a word of a synset, only the difference is user press the Update Synset button after making the changes to gloss and POS.

8.13.5 Delete a Synset

Deleting a synset can be done in two ways: (a) deleting the synset only or (b) Deleting a synset together with its corresponding concept

8.13.5.1. Deleting the synset only

To delete a synset the user first has to select the synset from the synsets panel and click on the delete button ☒ attached with the synset block.

(1) **Operation description:** Clicking on ☒ button will do the following:

- Delete the Synset for the active language. However the correspondence between a synset and a concept is one to one in any particular language. For a concept there can be multiple synsets in different languages but always at most one for each language. When deleting a synset, if it is the only one synset for a concept (considering all available languages), the deletion of that synset will also cause the deletion of the corresponding concept. However, the user is forbidden to delete both the synset and the concept in case the concept is related to other concepts. The user has to delete all the relations before being able to delete it.

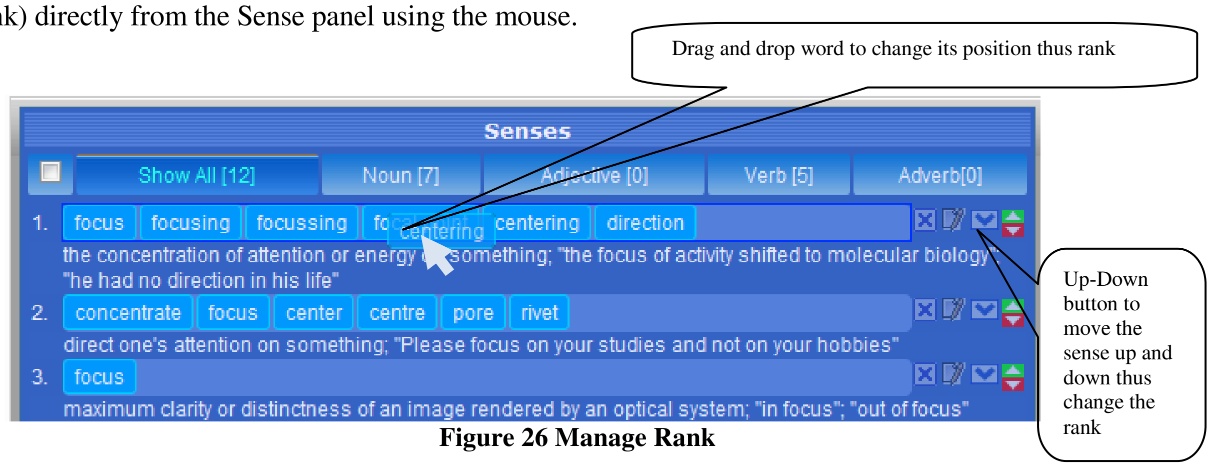
(2) **Post state change of the UI:** after pressing Delete Synset button the following changes will took place:

- CUD panel will be refreshed and show blank information.
- Synsets panel will be refreshed by removing the deleted synset form the synset list.
- Concept panel will be updated in the case also the concept is deleted.

Note that, deleting a synset is a language dependent operation, which means it will remove only that synset of the active language. There might exist other synsets in different languages for the same concept and those will not be removed.

8.14. Manage Rank

User can perform the sense and word’s rank related update operations (e.g., changing sense rank or word rank) directly from the Sense panel using the mouse.


Drag and drop word to change its position thus rank
Up-Down button to move the sense up and down thus change the rank
Senses
Show All [12]
Noun [7]
Adjective [0]
Verb [5]
Adverb[0]

1. focus focusing focussing fo centering centering direction
the concentration of attention or energy on something; "the focus of activity shifted to molecular biology"; "he had no direction in his life"

2. concentrate focus center centre pore rivet
direct one's attention on something; "Please focus on your studies and not on your hobbies"

3. focus
maximum clarity or distinctness of an image rendered by an optical system; "in focus"; "out of focus"

Figure 26 Manage Rank

8.14.1 Update Word Rank:

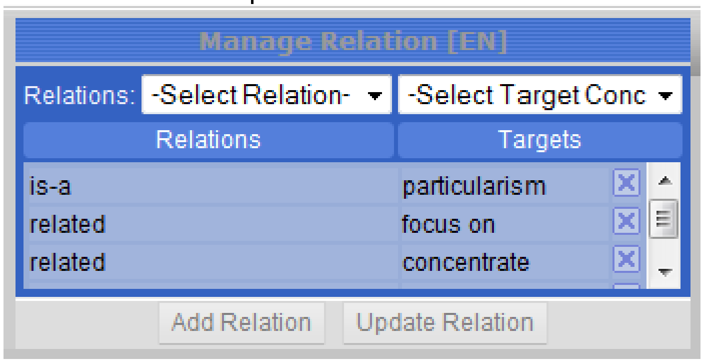
Visually words in a sense are positioned according to their rank w.r.t. the sense/synset horizontally from left to right in increasing order, which means the word with rank 1 would be in the first position, the word with rank 2 would be in the second position and so on. For example if we consider the 1st sense (see Figure 26 Manage Rank) the rank of “focus” is 1 , so it is in the 1st position, the rank of “focusing” is 2 thus it’s in the second position whereas in the 2nd sense the rank of the “concentrate” is 1 so it’s in the 1st position whereas the “focus” is in the 2nd.

8.14.2 Update Sense Rank:

Visually senses/synset associated to a word (the one selected from the word panel) are positioned according to their rank vertically from top to down in increasing order, which means that the sense with rank 1 would be in the first position of the sense/synset list, the sense with rank 2 would be in the second position and so on. The buttons ▼ and ▲ on the right corner of each sense/sunset block are used to change the sense rank of a sense/synset w.r.t the word selected in the word panel. If the user clicks▼, it will bring the corresponding sense/synset one step down and will increase the rank value and the next consecutive sense/synset will take its place. On the other hand, if the user clicks ▲ then the corresponding sense will swap with its previous sense in the same way.

8.15. Managing Concepts and Relations

To work on relations, the user first has to select a synset block from the synsets panel. The corresponding concept will appear in the concept panel. The user then has to click on the Edit button ✎ attached with the synset block. The CUD panel will appear with that selected Synset. Relations can be then managed from the Manage Relation section in the CUD panel.



Relations	Targets
is-a	particularism
related	focus on
related	concentrate

Figure 27 CUD panel (Manage relation)

(1) **Linking the (concept corresponding to the) synset with an existing concept:** To link a synset, or better its corresponding concept, to a concept the user first has to select the target concept from the concept hierarchy. If the concept has a label (then it has already a synset), the label will appear in the Manage Relation panel in the target concept field, otherwise the field will be blank. The user then selects a relation (e.g., is-a) using the dropdown relation list box right on the left of the target field. Finally the user presses the Add Relation button and the new concept will be added as a child node of the selected concept in the concept hierarchy.

(2) **Delete a relation:** To remove a relation the user has to click on the ☒ button of any relation from the Relation-Target table.

(3) **Update a relation:** Update a relation means update the relation's type only. To update a relation the user has to select the relation from the Relation-Target table that he wanted to update. The selected relation will appear in the relation and target concept fields. Only the relation dropdown list will enable and the target field will disable. The user can change the relation type (e.g., is-a, part-of) from the relation dropdown list to update. Finally the user clicks on "Update Relation" button to commit the change.

8.16. List of Icons and their meaning

List of icons that represent common tasks

Icon	meaning
➕	Add an object
☒	Delete an object
📝	Edit an object
🔻	Move down
🔺	Move up

8.17. Conclusion

In this Chapter we have presented our approach on information visualization and management of linguistic and concepts/ontologies. In our visualization method we represent ontology as a directed acyclic graph (DAG). This graph structure is based on a tree hierarchy and a Parent/Children List to visualize and navigate the resulting concept hierarchy as a DAG based on the sense/synset selection from the Senses panel. Because the visualization of all the connections between the concepts can be cluttering, we provide the user the possibility to filter the relations and get the requested part of the ontology. We also rationalized the design with appropriate theories wherever needed. A usability evaluation of this console is provided in Chapter 12.

Chapter 9

Having provided a brief theoretical description of the Domain knowledge in the chapter 5, in this chapter we describe the management console for it. We designed the domain management system to have a similar appearance to the linguistic and concept management system so that the user would not need to learn the systems separately.

9. Domain Management

9.1. Introduction

Visualization of domains is a complex task. A domain is something more than a hierarchy of concepts. It is enriched with role relations with various types of facets and each facet may belong to different category (e.g., entity, relation and attribute). Furthermore, each facet may have facet hierarchy, which could range from one or two level to hundreds or thousands. Therefore, it is not straightforward to construct a visualization that will effectively exhibit all this information and at the same time allow the user to easily carry out various operations on the domain.

The management console is meant to manage a flat list of domains. According to DERA, for each domain there is a finite list of facets that belong to the E, R and A fundamental categories. Each category may contain zero or more than one facet in a domain. The same facet can be used in different domain under different categories. A facet is a hierarchy of concepts connected with hypernym/hyponym and part meronym hierarchical relations. As an example, consider in Fig. 1 the domain Medicine and its constituents according to the DERA framework. Here there is exactly one facet in each category.

9.2. Functional description of the domain management system

So far we have described the theoretical background of faceted ontology. However building ontology is an expensive and time consuming laborious process. Traditionally, trained knowledge engineers build ontologies with the assistance of domain experts. As underlined in the State of the art section, most of the tools available for this task are too complicated. In many cases they also require users to be trained knowledge representation and also learn the logic. Therefore, starting from the matured experience of both library and computer sciences in knowledge representation and management, we plan to investigate the feasibility to develop a system able to:

- Provide efficient and effective tools for browsing and searching the classifications;
- Support unskilled users in building their own classifications in a flexible and effective way. This process will be accomplished on the basis of available knowledge, without forcing users to necessarily follow system suggestions;
- Support user in knowledge base evolution and customization;
- Import and manage a set of faceted classifications codifying knowledge provided by experts in each domain;

According to this view we can identify the following sub-tasks that would support by the system:

- **Building domain specific DEPA faceted classifications** : DEPA classifications representing domain knowledge in different domains are provided by domain experts. They will manually make according to the theory described above.

- **Finding a suitable visualization technique to represent domain and their facets**: we already have discussed the system (in previous section) to represent and manage linguistic knowledge. We need to investigate how to deal with the enhanced complexity due to the domain knowledge and their relations.
- **Exploring possible relations between domain and linguistic knowledge**: As a first hypothesis we can put in the domain knowledge only the most frequently used terms, the so called standard terms. We can then link each term in the domain knowledge with the corresponding synset in the linguistic one. This could mean that linguistic knowledge can be partitioned according to domains in several distinct subsets.
- **Managing universal knowledge:** The system takes in to account while editing domain and linguistic knowledge with particular emphasis to their relations. For example, when we add a new term in the domain knowledge we have to specify the corresponding synset in the linguistic knowledge, and when we delete a term either in the domain or linguistic knowledge we have to take into account related terms.
- **Assisting users in creating and maintaining their own classifications**: It is believed in library science that in order to be effective for search and navigation, facets should be ordered in decreasing order of extension, which is more general facets should be applied before more specific ones. This idea recalls the notion of classifications semantics [136], signifying that the system should suggest to the user the right way to arrange nodes in the classification.
- **Performing search and navigation:** Developing an efficient search and navigation environment to realize its potential benefits

The above mentioned tasks from implementation point of view we can categorized in to two different parts:

- **Search and navigation**: It includes Search, browsing domains and its components such as facets , entity, relations , attributes and also facets hierarchy. More in detail:
  a. Search and navigate Domains: The list (in alphabetical order) of already defined domains must be available on request to the user. A text search facility to search among them must be available.
  b. Search and navigate facets in a domain: By selecting a domain, it must be possible to list the facets in it by also emphasizing the corresponding category (E, R or A).
  c. Search and navigate in the Facets: By selecting a facet in a domain, it must be possible to navigate the hierarchical structure of a facet. For each node, the corresponding concept information must be visible. The navigation must be the same used for the navigation of the concept hierarchy in the ontological part.
  d. Search and navigate Concepts : It must be possible to search and navigation of the concept hierarchy in the ontological part to locate a concept that would use for defining the domain or facet.
- **Maintenance part** (e.g., CUD operations for create, update and delete operation):
  - **Operations on domain**: This is the list of operations that the user interface must support on domains:
    i. *Create a domain*: A domain is created by selecting the corresponding concept. At this purpose, the UI must provide a facility to search for a suitable concept by typing the name of the domain. It must be possible to create a new concept in case no suitable concept is found for it.
    ii. *Update the name/concept of a domain*: It must be possible to change the name/concept of a domain. Facets in it must be preserved.
    iii. *Add a facet to a domain*: It must be possible to search for defined facets (regardless if they are already associated to other domains), select one facet and add it to a domain using a category (E, R or A).
    iv. *Remove a facet from a domain*: It must be possible to select a facet from the domain and remove it from the domain. This does not result in the deletion of the facet.
    v. *Update the category of a facet*: By selecting a facet in a domain, it must be possible to change the category.

vi. *Delete a domain*: By selecting a domain, it must be possible to delete it. This causes the deletion of all the facets in it in the case they are not used by any other domain.

- **Operations on facets** : This is the list of operations that the user interface must support on a facet (not necessarily associated to a domain):

vii. *Create a facet*: A facet is created by specifying the concept of its root node. At this purpose, similarly to domains, the UI must provide a facility to search for a suitable concept by typing the name of the facet. It must be possible to create a new concept in case no suitable concept is found for it

viii. *Add a child note*: By selecting a node in a facet, called source, it must be possible to add a new child node to it, called target. Child nodes must be selected from those already connected to the source with a hierarchical relation in the conceptual part. In other words, the hierarchical relation should already exist. If the target node or the relation does not exist yet, the user must be able to create them by using the UI for the linguistic and ontological part.

ix. *Remove a child node*: By selecting a node in a facet, it must be possible to remove it from the facet (along with the relation connecting it to the parent. Note that this operation does not cause the removal of the relation from the ontological part. Note also that the removal of a node causes the removal of the whole subtree from the facet. Therefore, a notification should be given to the user to decide whether to confirm the operation or not. This operation is not applicable to the root node

x. *Delete a facet*: By selecting a facet from the list of facets in a domain, it must be possible to delete it. It is not possible to delete a facet if it is used in other domains.

The knowledge base contains the knowledge and the inference engine will use to create new knowledge as well as to perform update, delete, search and navigation on the knowledge stored in knowledge base. The goal of the subsequent sections is to make design specifications for a user-friendly management console to perform basic CRUD operations including search and navigation on the Domain part for the Universal Knowledge.

9.3. Search and navigation

9.3.1 Search and navigation of Domains

By domain search and navigation, we mean searching for or navigating the domains created under the DERA framework. The goal of domain visualization is to provide a meaningful context in which user can explore the body of knowledge as a whole. Providing the facility for domain search is one of the basic features. Since numbers of top domains are not many, a alphabetically ranked tabular version, as shown in Table 1, is the most efficient form for immediately identifying the domain. Basically, a specific domain and its corresponding description can be singled out immediately from the alphabetically ranked list.

Domains	Descriptions
Space	the unlimited expanse in which everything is located; "they tested his ability to locate objects in space"; "the boundless regions of the infinite"
Time	measure the time or duration of an event or action or the person who performs an action in a certain period of time;
..	…

Table 4: List of domains

Unfortunately, this technique encounters the physical problem if the domain list intended for visualization becomes too long. Filter can come to our aid. In particular, information visualization systems appear to be most useful when a user can refine the search result. The idea is initially show all the available top domains as shown in Table 1, pressing any key will automatically filter the domain whose first character matches the pressed key, and any subsequent keypress will further refine the sea rch considering also the second character, then the third and so on. This mechanism is an obvious solution and widely adopted. Like as Linguistic part, supporting search and navigation in multiple languages is one of the main features. Search result list would used to navigate through the domains and provide a means for selecting query scope

9.3.2 Search and navigation of Facets

Each domain consists of a set of facets and facets in a domain are grouped into specific elementary categories i.e., entity, relation and attribute. If we look at the conceptual model of Domain (see Figure 6 : Space domain (partial view)) we see the model is similar to the Linguistic and Concept Knowledge (see Figure 5 Conceptual model of Linguistic and Concept Knowledge). Instead of word list here we have a list of domains. Each word has one or more synsets whereas in this case each domain has a list of facets. Each synset has a concept and in this case each facet has a facet hierarchy. Therefore, like as before this model we can easily map with a directed graph. In this case, to present domain, facets and the facet hierarchy we can use three columns panel, where the first column would show the domain list, the second column is for facets correspond to the domain and the third column is for facet hierarchy corresponds to a facet. Each domain from the first column can treat as a parent node. Clicking on a domain would initiate the query and show a list of facets together with their descriptions for the domain that has been clicked, to an 'exploded' state in the second column. Clicking on a facet would show the corresponding facet hierarchy for the facet that has been clicked, to an 'exploded' state in the third column.
Facets are further categorized according to DERA. We assume user will start search with domain and they will get in to the facets through the domain navigation. So scope of the facets visualization depends on the search domain. For some of the domain its facet list could be quite long, as it will include the facets from all the category. We can use elementary category to further filter the query result. An automatic grouping according to the elementary category could shorten the long list by grouping the facets by elementary category. As discussed earlier in Linguistic part, A better way to show them as Tab view. A tab view provides a convenient way to present information in a multipane format. The tab control is displayed horizontally centered across the top edge of a content area. Each tab would contain one group represents a E, R or A. Users can easily switch between tabs to see the content associated with it.

9.3.3 Search and navigation of Facet hierarchy

A facet can have sub facets which create facet hierarchy. For example Figure 1 shows a facet *body of water* of *Space* domain, where *Ocean*, *Intel* and *Sea* are the three sub facets of it. As shown in the above figure the sub facet *Sea* is further subsumes three facets namely *Gulf, Cove* and *Bay*. A facet hierarchy generally consists of hierarchical relations. The label of the representing concept represents the node label of the facet hierarchy. Unlike concept hierarchy the facet hierarchy is generally is an IS-A hierarchy or more rarely by composition (PART-OF). Therefore, the concept expressed by the parent node is as expected more general than the concepts expressed by its children.
The more intuitive way to represent a facet hierarchy is through a rooted tree where the root label is the name of the facet and internal and leaf nodes' labels represent either its children of the facet. From the above analysis we can identify some essential elements that are need to visualize the facet hierarchy as follows:

- A facet tree that allow navigating through the facets. The root will be treated as a parent facet and its children will be shown as subsequent child(s). Notice that we also have to show the kind of each relation, in this case is-a and part-of in some way.
- A relation filter to reduce the complexity of the visualization, e.g. by selecting a specific relation and filtering out the others representing the facet hierarchy.
- Often tree nodes label are not enough descriptive, we should show the description in some way whenever it needed.

To visualize the facet hierarchy we will use the same representation technique that we have used in the linguistic part except that in this case we will not have the **"Parent/Children Box"** since there will be always only one parent. Also note that will have only one direction of display as the parent facet is always expected more general than its children facets. The added value of our visualization lies in its expressivity. The facets and their relationships between their children facets are easy to detect.

9.4. Domain Management Functionalities (create/update/delete)

We clearly separate the CUD (create/update/delete) operations part from the search and navigation to avoid the clutter. In the following sections, we describe in detail the CUD operations on domains, facets and facets hierarchy. We also present the pseudo-code whenever it is needed. Like as search and navigation, type of operation also varies from object to object. "CUD" operations can be divided into four different parts where each part is responsible for a set of related functionalities:

- **Components of Domain**: We identified the following components those are directly related with a domain:
    a. A concept that represents the domain. The concept must exist.
    b. A set of facets associated with the domain according to DERA category
- **Components of Facet** : We identified the following components those are directly related with facet:
    a. A concept that represents the facets. The concept must exist. Notice that, this facet is an isolated facet and not associate to any domain yet.
- **Components of Facet hierarchy**: We identified the following components those are directly related with relation:
    a. A source root facet
    b. A set of child facets whereas the parent node is more general than its children

9.4.1 Managing Domains, Facets and Facet hierarchy

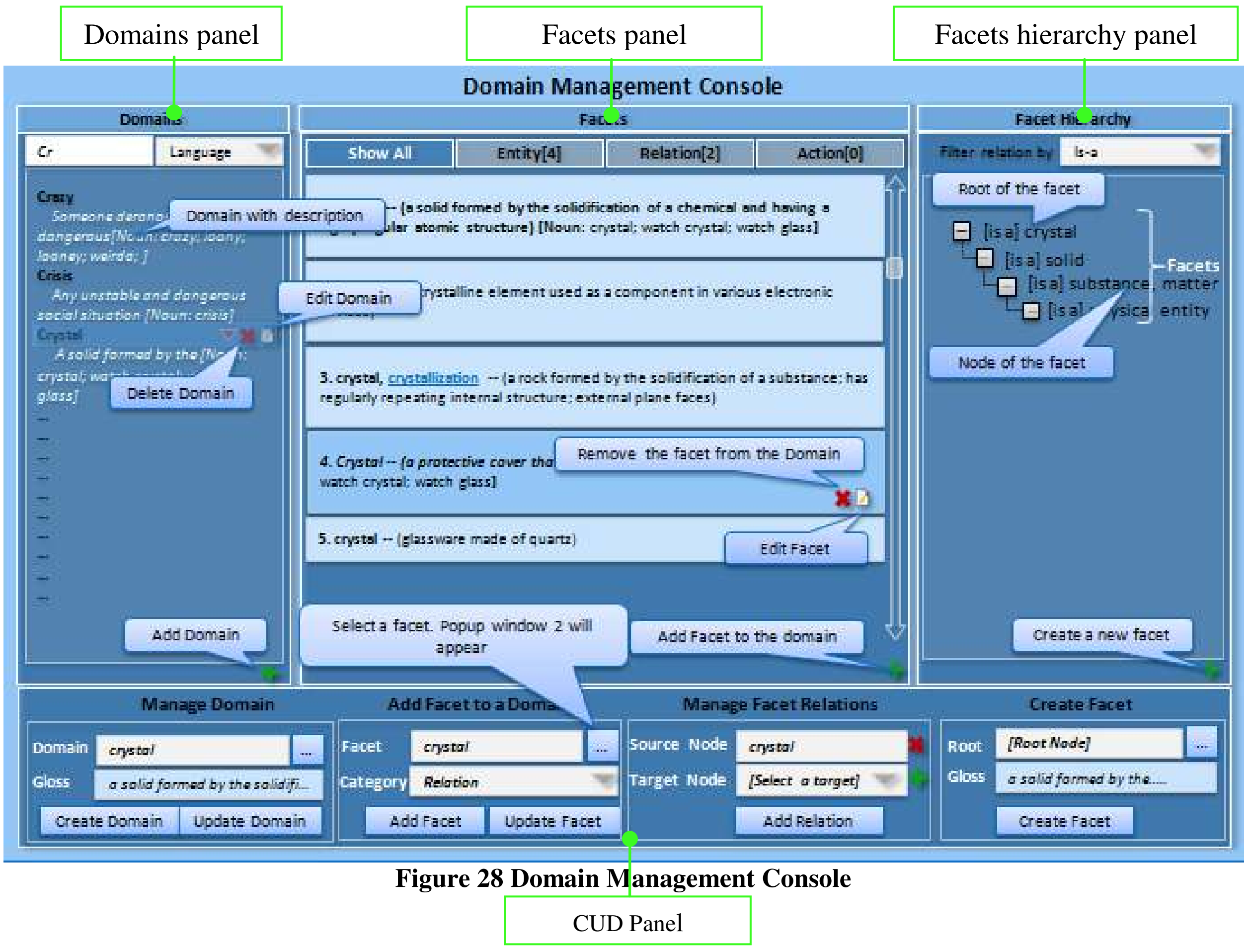
It is important to underline that for us the notion of Facet is central. The user can build domains and facets only using the existing concepts and relations built in linguistic and ontological part. In particular, these are the operations that a user can perform to manage domains, facets and facets hierarchy:

1. ***Create a domain***: A domain is created by selecting the corresponding concept. The concept must exist in the Linguistic and Ontological part. However this selection process is not so straightforward, It requires lots of background work. In general, this process starts with collecting the terms representing the relevant real world entities of the domain at hand. This is mainly done by interviewing domain experts and by reading available literature on that particular domain. Analysis of query logs, when available, abstracts, and glossaries, reference works can be extremely valuable to determine user's interests. Each term is analyzed and disambiguated into an atomic concept. When an atomic concept is identified it's been searched in the in the Linguistic and Ontological part. If the concept exist it will be select from there and if the concept doesn't exist then first it have to create in the Linguistic and Ontological part and then use it to make semantics effective.
2. ***Update a domain***: update by changing the concept that is used to describe the domain.
3. ***Delete a domain***: delete the domain from the domain list.
4. ***Add a facet to a domain*****:** Aassociate a facet to a domain using a category (E, R or A). First the facets are analyzed in order to identify their commonalities and their differences. The main goal is to identify as many distinguishing properties - called characteristics - as possible of the real world entities represented by the facets.
5. ***Removing a Facet from a Domain:*** removing a facet from a domain but not deleting the facet
6. ***Updating the category of a domain facet:*** update the category of a domain facet by changing its category, possible values are Entity, Property and Action .

7. ***Create a Facet:*** A facet is created as a root node by selecting the corresponding concept Linguistic and Ontological part. The concept must exist in the Linguistic and Ontological part.
8. ***Delete a Facet:*** delete a facet by selecting the root node of the facet from the "Facets hierarchy"
9. ***Add a node to a facet hierarchy:*** add a new child node to a facet hierarchy. This child node must exist in the Linguistic and Ontological part as a child of concept that represent the selected node. This process requires proper *Synthesis.* The synthesis aims at arranging the atomic concepts into facets by characteristic. At each level of the hierarchy - each of them representing a different level of abstraction. Similar concepts are grouped by a common characteristic. Children are connected to their parent through a genus-species or whole-part relation.
10. ***Remove a node from a facet:*** selecting a node from the facet hierarchy it is possible to remove it from the facet (along with the relation connecting it to the parent)

9.5. General overview of the console

The console consists of two different parts (1) "Search and navigation" and (2) "CUD (create/update/delete)" part. Further, the "Search and navigation" part is divided into three different panels namely: (a) Domains panel (b) Facets panel and (c) Facets hierarchy panel. Each part is responsible for a set of related functionalities. The specialty of this prototype is very "space efficient" while maintaining the navigation context. In addition to structural clarity, this design provides simple navigational structure to explore the challenging manifestation of the complexities of Domain knowledge.



Crazy
Someone derang... dangerous[Noun: crazy; loony; looney; weirdo;]
Crisis
Any unstable and dangerous social situation [Noun: crisis]
Crystal
A solid formed by the [Noun: crystal; wat... glass]

-- (a solid formed by the solidification of a chemical and having a ...ular atomic structure) [Noun: crystal; watch crystal; watch glass]

...rystalline element used as a component in various electronic ...

3. crystal, crystallization -- (a rock formed by the solidification of a substance; has regularly repeating internal structure; external plane faces)

4. Crystal -- (a protective cover tha... watch crystal; watch glass]

5. crystal -- (glassware made of quartz)

[is a] crystal
[is a] solid
[is a] substance, matter
[is a] ...ysica... entity

Domain: crystal
Gloss: a solid formed by the solidifi...

Facet: crystal
Category: Relation

Source Node: crystal
Target Node: [Select a target]

Root: [Root Node]
Gloss: a solid formed by the....

Figure 28 Domain Management Console

(1) **Search and navigation**: the upper part of the console is dedicated to search and navigation mainly, and consists of three different panels.

a. *Domains panel*: this panel provides the facilities for domain search, language selection, and a search result list based on the searched domain. Search result list is used to navigate through the domains and provide a means for selecting query scope.
b. *Facets panel*: The result of this panel depends on the search domain in domain panel. In general, it will show a list of facets for the domain that has been searched. Results are shown grouped by category.
c. *Facets hierarchy panel*: result of this panel depends on the facet selection from the facet panel. Facets and corresponding relations are shown as a tree.

(2) **CUD-Create/Update/Delete**: the lower part of the console is dedicated to CUD operations on domains, facets, facets hierarchy and relations between them.

In the following sections, we describe in detail the single functionalities.

9.6. Detailed description of the Search and Navigation Panel

9.6.1 Domains Panel

This panel provides the facilities for domain search and filtering based on language selection, autosuggestions, and a sorted search result list.

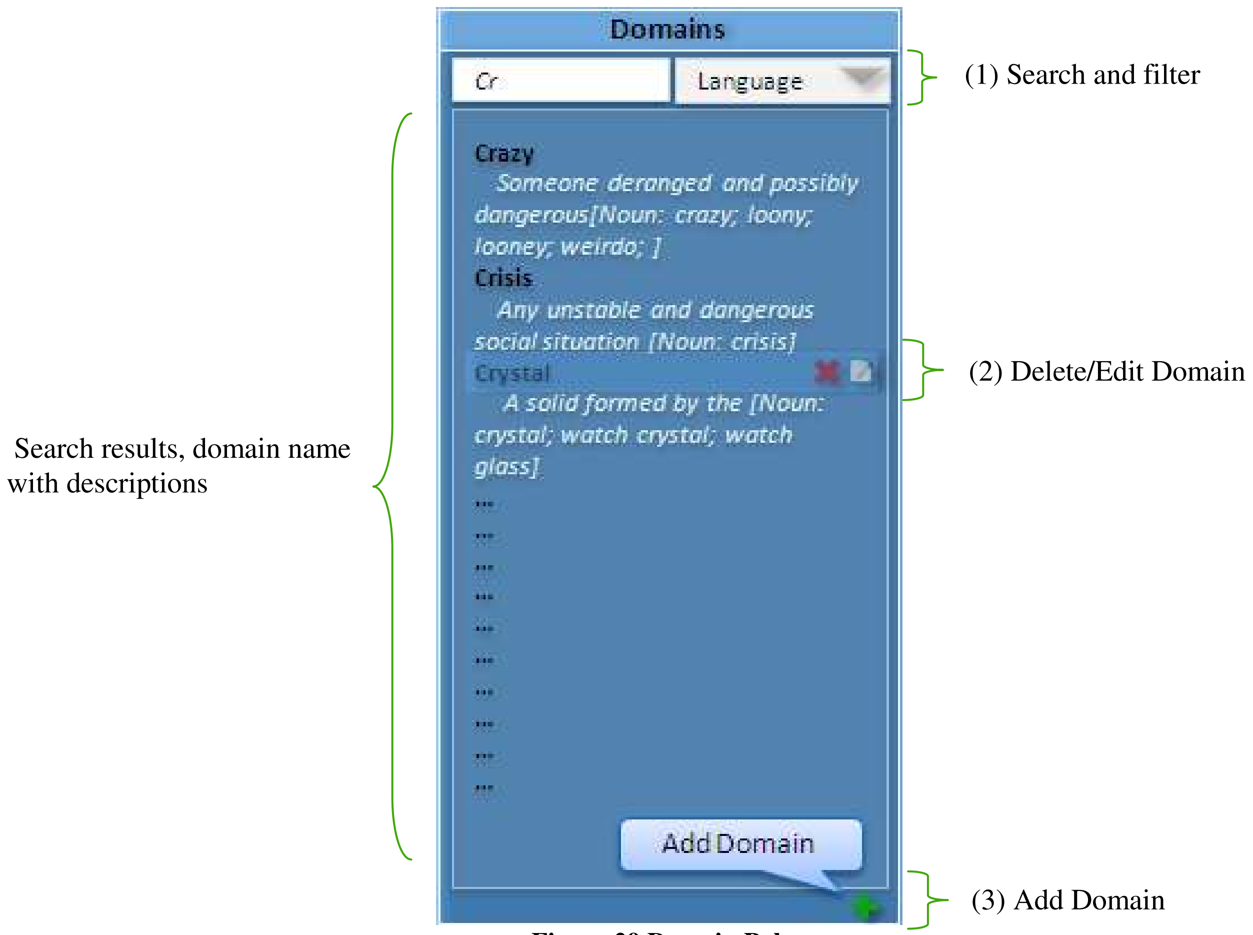

Domains
Cr
Language
Crazy
Someone deranged and possibly dangerous[Noun: crazy; loony; looney; weirdo;]
Crisis
Any unstable and dangerous social situation [Noun: crisis]
Crystal
A solid formed by the [Noun: crystal; watch crystal; watch glass]
Add Domain
(1) Search and filter
(2) Delete/Edit Domain
(3) Add Domain
Search results, domain name with descriptions

Figure 29 Domain Palen

(1) **Search and filtering** input panel consists of the following components:

a. Language selection combo: A dropdown list box for selecting the language. When the user selects a different language, all information in the UI remains the same but it is shown in the selected language.

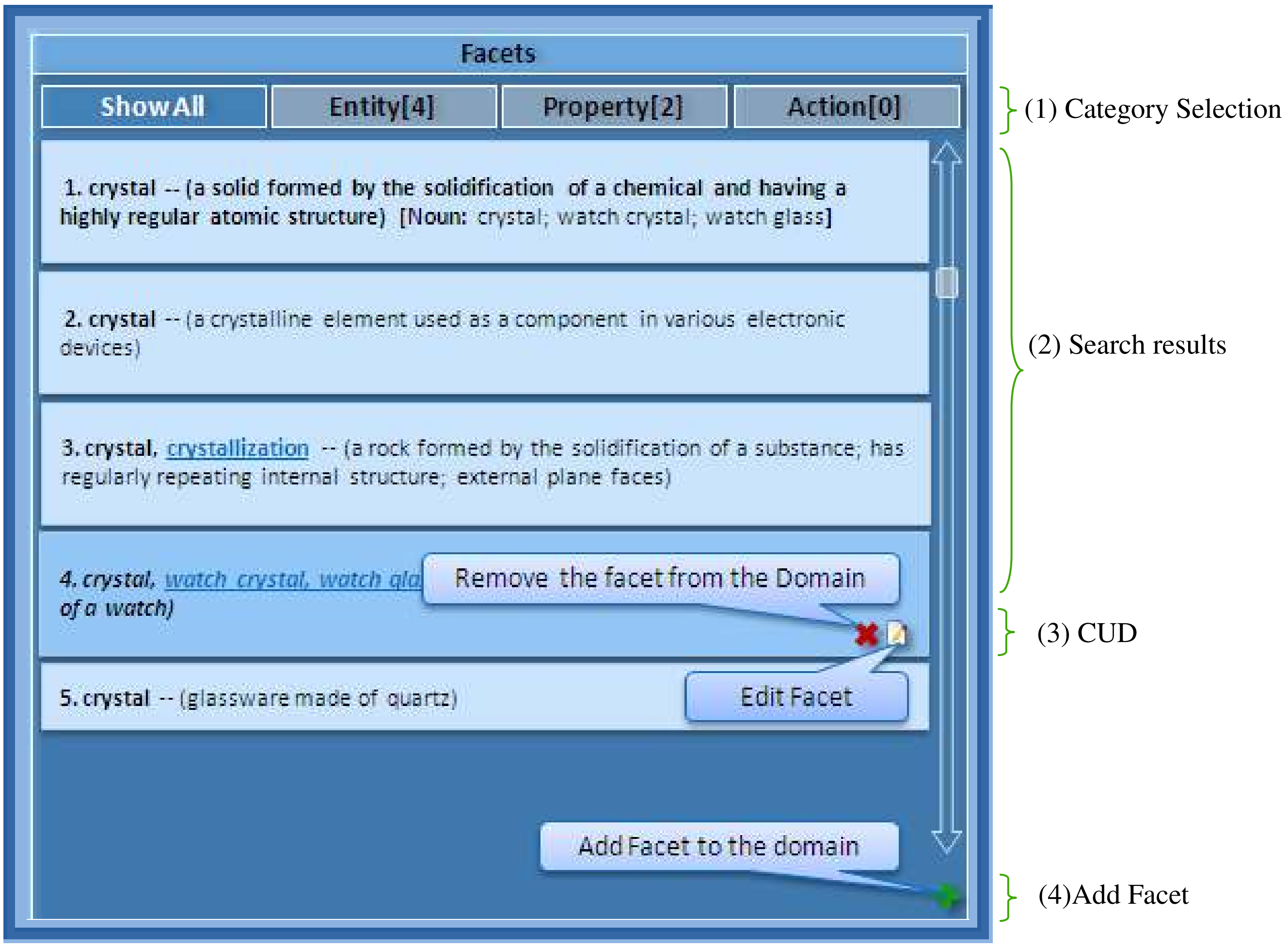
b. A text box for free text input: Figure 29 Domain Palen shows a situation in which a user is typing a domain name in the domain input text box. While typing a domain name a list of results (domain name + description) will appear in the search result list box in alphabetical order together with the searched domain (if defined), and then the user can select a domain from the list. The description associated to the domain is the gloss of the corresponding synset and its POS. Facet(s) for that selected domain would appear as a list in the Facet panel for that domain.

(2) **Delete/Edit domain.** The user can delete or Edit a domain by clicking on ✖ and 📝 respectively. Note that, the operations will be applicable only on the "selected" domain from the Domain panel.

(3) **Adding a new domain:** In the bottom of the domains panel there is an add button ✚ which is for adding a new domain.

9.6.2 Facets Panel

Facet panel is organized as a set of *Category* (**E**, **R** and **A**) selection tabs. The list for presenting search results - i.e. the facet associated to the selected domain - consists of a set of blocks, each block representing a facet. Each block contains a set of buttons to delete or update the facet. Finally, at the bottom of the result list, a plus button can be used to add a facet (i.e. associate an existing facet) to the current selected domain.


Facets

ShowAll | Entity[4] | Property[2] | Action[0]

1. crystal -- (a solid formed by the solidification of a chemical and having a highly regular atomic structure) [Noun: crystal; watch crystal; watch glass]

2. crystal -- (a crystalline element used as a component in various electronic devices)

3. crystal, crystallization -- (a rock formed by the solidification of a substance; has regularly repeating internal structure; external plane faces)

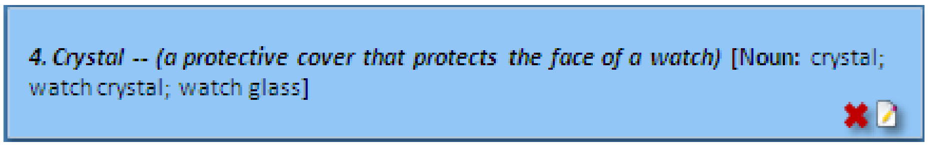
4. crystal, watch crystal, watch gla of a watch)

5. crystal -- (glassware made of quartz)

Figure 30Facet Panel

(1) **Category Selection**: The user can filter the result based on category (E, R and A) by clicking on any tab for category. Each tab also shows the number of facets for every category inside the brackets.

(2) **Facet search result**: It is a list representing the filtered facets (according to the selected category). Results are presented as blocks of information in a linear list. Each block represents a facet. Initially each block contains the facet name together with its synonyms, POS and corresponding gloss from the corresponding synset. User can further navigate by clicking on the facet. In this case, the facet hierarchy for the selected facet will appear in the facet hierarchy panel (see next section).


4. Crystal -- (a protective cover that protects the face of a watch) [Noun: crystal; watch crystal; watch glass]

(3) **CUD Operation on a facet**: user can delete or update facets by clicking on ✖ and 📝 respectively. Note that, the operations will be applicable only on the "selected" facet from the facet panel.

(4) **Adding a facet to a domain**: In the bottom of the facets panel there is an add button ➕ which is for associating an exisitng facet to a domain. This functionality is described in detail in Section 6.

9.6.3 Facets hierarchy Panel

Facets hierarchy panel (see Figure 31 Facet Hierarchy Panel) displays the resulting facet hierarchy based on the selection of a facet from the facets panel. This panel consists of a tree to visualize and navigate the facet.

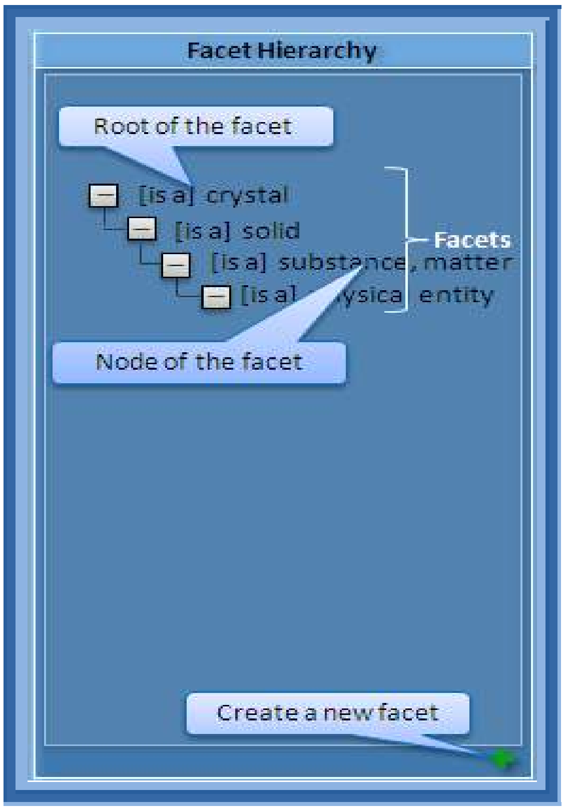


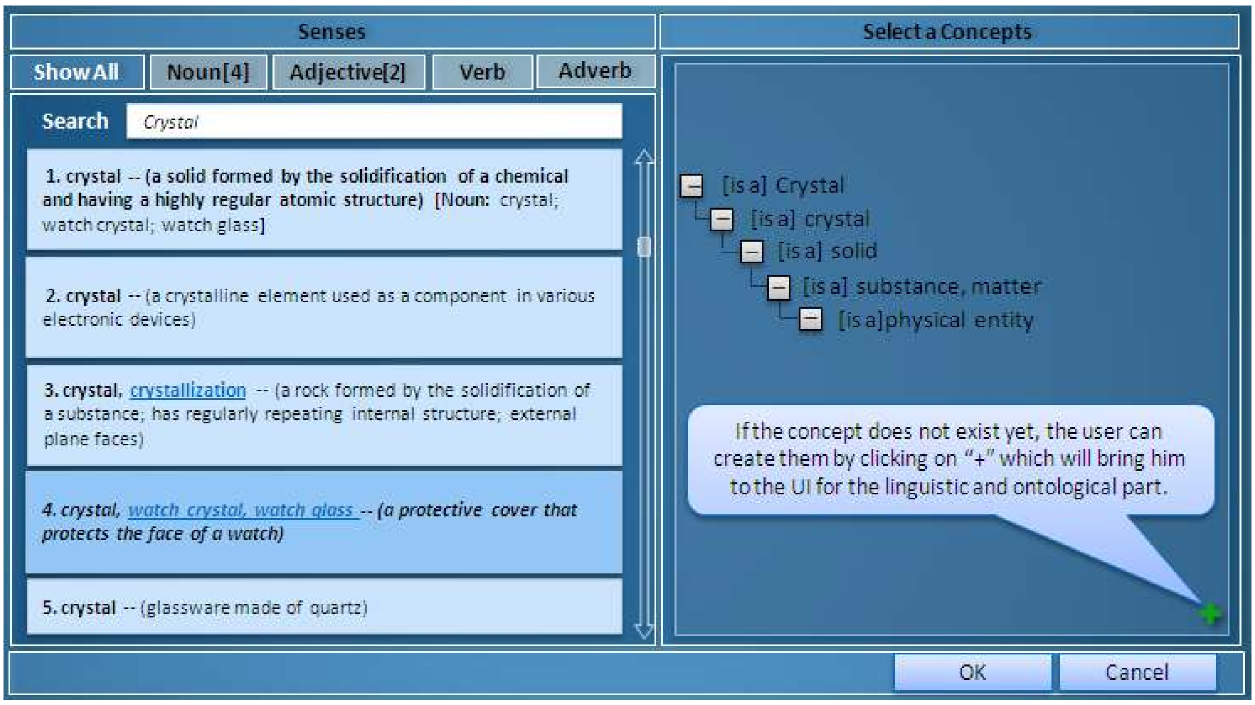
Figure 31 Facet Hierarchy Panel

(1) **Facet description:** A tooltip will show the description/gloss of the facet as a mouse over effect on any node of the tree.

(2) **The facet hierarchy:** The panel used to navigate the facet hierarchy as a tree. The navigation starts from the root node and can continue through the children. Notice that the kind of each relation (from source to target) is shown in squared brackets and can be of two types only: [is a] and [part of]. Facets are visually represented as trees. Nodes and relations are represented following the same principles already used in linguistic and concept part

(3) **Create a new facet:** In the bottom of the facet hierarchy panel there is an add button ✚ which is for creating a new facet. This functionality is described in detail in later Sections.

9.6.4 Pop Up window for concept selection



1. crystal -- (a solid formed by the solidification of a chemical and having a highly regular atomic structure) [Noun: crystal; watch crystal; watch glass]

2. crystal -- (a crystalline element used as a component in various electronic devices)

3. crystal, crystallization -- (a rock formed by the solidification of a substance; has regularly repeating internal structure; external plane faces)

4. crystal, watch crystal, watch glass -- (a protective cover that protects the face of a watch)

5. crystal -- (glassware made of quartz)

[is a] Crystal
[is a] crystal
[is a] solid
[is a] substance, matter
[is a]physical entity

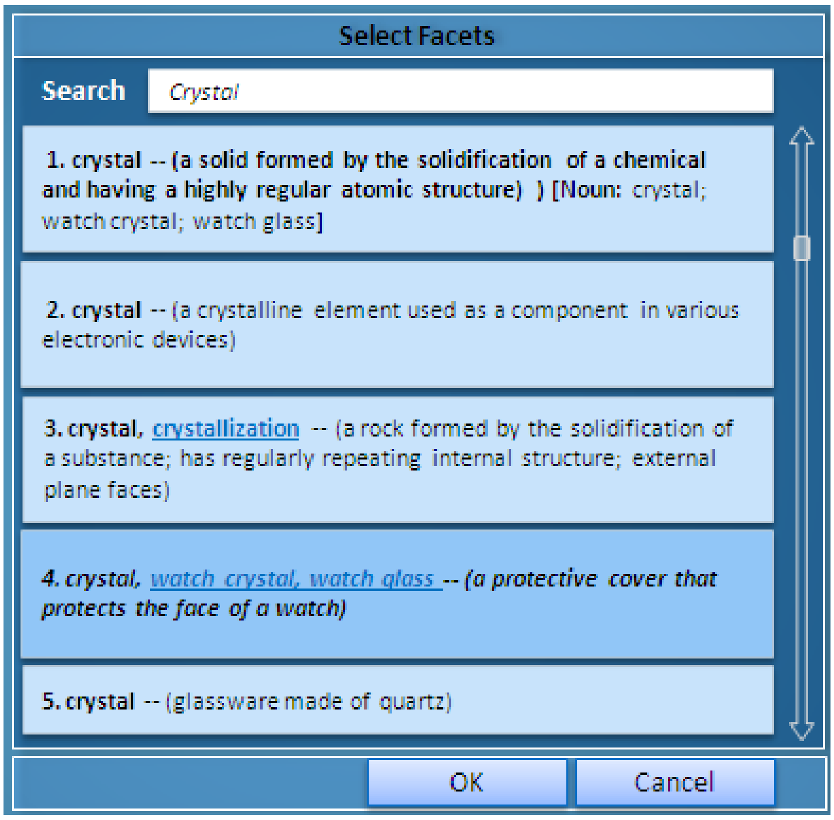
If the concept does not exist yet, the user can create them by clicking on "+" which will bring him to the UI for the linguistic and ontological part.

Figure 32 Select concept selection pop up window

Select Concept window (in the figure above) enables the user to select a concept from the Linguistic and Ontological part. Only an already existing concept can be selected from this pop up window. If the concept does not exist yet, the user can create it by clicking on the ✚ button which will bring the user to the UI for the Linguistic and Ontological part.

9.6.5 Pop Up window for facet selection

Select Facet window enables the user to select a facet from the existing facets. Only an already existing facet can be selected from this pop up window. If the facet does not exist yet, the user can create it from the CUD panel using the "Create Facet" segment (see Figure 34 CUD Panel).



Crystal

1. crystal -- (a solid formed by the solidification of a chemical and having a highly regular atomic structure)) [Noun: crystal; watch crystal; watch glass]

2. crystal -- (a crystalline element used as a component in various electronic devices)

3. crystal, crystallization -- (a rock formed by the solidification of a substance; has regularly repeating internal structure; external plane faces)

4. crystal, watch crystal, watch glass -- (a protective cover that protects the face of a watch)

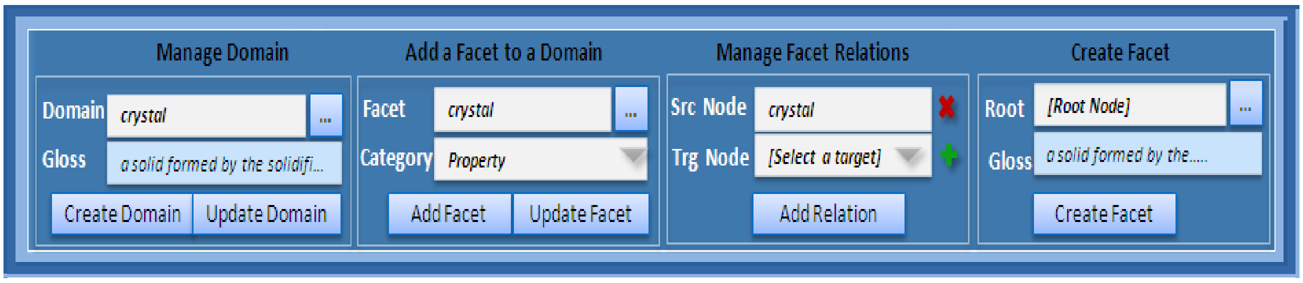
5. crystal -- (glassware made of quartz)

Figure 33 Select facet pop up window

9.7. Detailed description of the CUD Panel

CUD panel is dedicated for any sort of add, delete or update operations on domains, facets and relations between the nodes in a facet. To work on any existing domain or facet the user has to select a domain from the domain panel or select a facet from the facets panel or a node from the facet hierarchy panel and initiate the desired operation by clicking on the appropriate button provided there.

For example, selecting the 4th facet from the facets panel (see Figure 33 Select facet pop up window) and by clicking on the edit button, it would show all its information in the CUD panel (see Figure 34 CUD Panel), or selecting a node from the facet hierarchy would show the information in the CUD panel which facilities for managing relations for the corresponding facet. Alternatively, the user can also create a new relation between two nodes by using the ✚ button.



Domain: crystal
Gloss: a solid formed by the solidifi...
Facet: crystal
Category: Property
Src Node: crystal
Trg Node: [Select a target]
Root: [Root Node]
Gloss: a solid formed by the.....

Figure 34 CUD Panel

The overall CUD panel is divided in to four segments, namely “Manage domain”, “Add facet to a Domain”, “Manage Facet Relation” and “Create Facet”.

9.7.1 Manage Domain

The Manage Domain part of the CUD panel is dedicated to the management of the information related to domains. It consists of the following components:

a. Domain: a text field for entering a domain name. It must correspond to an existing synset/concept.
b. Gloss: a text field to display the gloss of the corresponding synset of the concept that has been selected to represent the domain
c. […] a browse button. Clicking on it will open a popup window "Select Concept" window (see Figure 33 Select facet pop up window) for selecting the concept used to describe the domain.
d. "Create Domain" button to confirm the creation of a new domain corresponding to the selected synset/concept.
e. "Update Domain" button to update the synset/concept of the domain.

9.7.2 Manage Domain facet

The second segment of the CUD panel is to associate an existing facet to a domain. It is dedicated to the management of the information directly related to the facet within a domain. It consists of the following components:

a. Facet: a text area for entering the root node of the facet (it must be selected from the existing facet using "Select Facet" window)
b. Category: a dropdown list box for selecting category; possible values are "Entity", "Property" and "Action".
c. […] a browse button. Clicking on it will open a popup window "Select Facet" window (see Figure 33 Select facet pop up window) for selecting the (root node of the) facet to add to the domain.
d. "Add Facet" button to confirm the association of the facet to the selected domain.
e. "Update Facet" button to change the category of a facet.

9.7.3 Manage Facet Relation

The third segment of the CUD panel is the manage facet relation. Relation related operations and operations on nodes can be done from here. This part consists of the following components:

a. **Source Node**: A text input followed by a cross button shows the node of the facet that has been selected from the Facet hierarchy. This node is called the source node. Clicking on the ✗ button will delete the selected node from the facet. If the node is the root then the whole facet will be deleted.
b. **Target Node**: A dropdown list followed by a ✚ button for selecting the target node.The dropdown list will show only the concepts that are targets of the source node under the is-a and part-of relations only. If the desired target doesn't exist yet, the user can create a new concept or relation by clicking on the ✚ button that will bring the user to the UI for the linguistic and ontological part.
c. "Add Relation" button to confirm the creation of the relation between the source node and the target node in the facet.

9.7.4 Create Facet

The rightmost part of the CUD panel is the "Create Facet", dedicated to the creation of a new facet starting from the root node. It consists of the following components:

a. Root Node: a text field for entering the root node of the facet. It must correspond to an existing concept.
b. Gloss: a text field for display the gloss of the corresponding synset of the concept that has been selected to represent the facet.
c. […] a browse button. clicking on it will open a popup "Select Concept" window for selecting the concept that would use to describe the facet.
d. "Create facet" button to confirm the creation of the new facet.

9.8. Managing Domains and Facets

It is important to underline that for us the notion of Facet is central. The user can build domains and facets only using the existing concepts and relations built in linguistic and ontological part. In this section we provide the details on how the operations described in section 3 are performed

9.8.1 Create a Domain

A domain is created by selecting the corresponding concept. The concept must exist in the Linguistic and Ontological part. At this purpose, the "**Select Concept**" window (see Figure 33 Select facet pop up window) provides a facility to search for a suitable concept by typing the name of the domain. In case no suitable concept is found for it the user can create a new concept clicking on the ➕ button of the "**Select Concept**" window which will bring the user to the UI for the linguistic and ontological part to create new concept.

(1) **Information to provide:** It is mandatory to provide the synset/concept associated to the domain and it must exist.

(2) **Operation description:** The user first clicks on the add domain ➕ button located on the bottom of the Domains panel. Clicking on add domain ➕ button will enable the CUD panel "Manage Domain" part with blank information. Only the **Create Domain** button of the CUD panel will be enabled. All other buttons will be disabled. User performs the following steps to create the domain:

   a. The user must search for a suitable concept using the […] button. The "Select Concept" Pop up window will appear (see figure 8).

   b. The user has to browse the concept hierarchy and select the appropriate concept that would describe the domain.

   c. The user press the OK button of the "Select Concept" window, the window will disappear and the selected concept will appear in the domain input field. The corresponding natural language description of the concept will be shown below.

   d. If the desired concept doesn't exist yet, the user can create a new synset/concept by clicking on the ➕ button from the "Select Concept" Pop up window that will bring him to the UI for the linguistic and ontological part.

   e. Finally, the Create Domain button has to be pressed for the final commit

(3) **Post state change of the UI**: after pressing the Create Domain button CUD panel will be refreshed. The new domain will appear in the list of domains.

9.8.2 Update a Domain

Update a domain means changing the concept that is used to describe the domain. To update a domain the user has to select the domain from the domain list of the domains panel and click on the 📝 button attached with the domain name. The selected domain will appear in the domain field of the "Manage Domain" panel. Only the Update domain button will be enabled. The user can change the concept using the "Select Concept" pop up window. Finally the user clicks on "Update Domain" button to commit the change.

(1) **Information to provide**: It is mandatory to provide the concept associated to the domain and it must exist.
(2) **Post state change of the UI**: after pressing the Add button the Domain panel will be refreshed. The updated domain will appear in the domains list.

9.8.3 Delete a Domain

To delete a domain the user first has to select the domain from the domains panel and click on the delete button ✗ attached with the domain name.

(3) **Operation description:** Clicking on ✗ button will delete the domain. However the corresponding concept will remain intact. This does not cause the deletion of the facets associated to it.

(4) **Post state change of the UI:** after pressing Delete Domain button the following changes will took place:

- Facets panel will be refreshed by removing the facets form the facet list.
- Facet hierarchy panel will be refreshed.

9.8.4 Add a facet to a domain

The user can associate a facet to a domain (regardless if it is already associated to other domains), by selecting one facet and associating it to the domain using a category (E, R or A).

(1) **Information to provide:** Mandatory information for adding a facet are:

- *Facet:* the facet must exist. The user selects the facet using the "Select facet" pop up window.
- *Category:* the user has to select a category from the category dropdown list. Possible values are Entity, Property and Action.

(2) **Operation description:** The user first clicks on the add facet ➕ button located on the bottom of the Facet panel. Clicking on add facet ➕ button will activate the "Add a facet to a Domain" segment of the CUD panel with blank information. Only the **Add Facet** button of the CUD panel will be enabled.

- The user must search for a facet using the […] button. It opens the search facet window.
- The user selects a facet from the "Select Facet" window and press OK button. If the facet does not exist yet, it has to be created using the "Create Facet" segment of the CUD panel.
- The selected facet will appear in the Facet input field
- *Select category:* the user selects a category from the category dropdown list. Possible values are Entity, Property and Action.
- Finally, Add Facet button will be pressed for final commit.

(3) **Post state change of the UI:** after pressing the Add button the Facets panel will be refreshed. The new facet will appear as a block in the facets panel.

9.8.5 Removing a Facet from a Domain

To remove a facet from a domain the user first has to select the facet from the facets panel and click on the delete button ✗ attached with the facet block.

(5) **Operation description:** Clicking on ✗ button will do the following:

- Remove the facet from the domain.

(6) **Post state change of the UI:** after pressing Delete Facet button the following changes will took place:

- CUD panel will be refreshed and show blank information.
- Facets panel will be refreshed by removing the facet form the facet list.
- Facet hierarchy panel will be refreshed and show blank information.

Note that, this operation causes the removal of the facet from the active domain only. Other domains might use the facet.

9.8.6 Updating the category of a facet:

The user can update the category of an existing facet as follows:

(1) **Information to provide:** Mandatory information:

- Category: the user has to set a category for the facet in the domain

(2) **Operation description:** To update a facet category of a domain:

- The user first selects a facet from the facets panel.
- The user clicks on the edit ✎ button associated to the facet block. As a consequence, the "Add a Facet to a Domain" segment of the CUD panel will be initialized with the information of the selected facet. Only the Update Facet button of the CUD panel will be enabled.
- The user selects the desired category from the category dropdown list. Possible values are Entity, Property and Action.
- Finally, the Update Facet button will be pressed to commit the operation.

(3) **Post state change of the UI:** after pressing the Update facet button the Facets panel will be refreshed. The facet will be shown in the updated category.

9.8.7 Create a Facet

A new facet is created starting from the root node as follows:

(5) **Operation description:**

- The user clicks on the "Create facet" button ✚ in the "Facet hierarchy" panel. This will activate the "Create Facet" segment of the CUD panel.
- The user must search for a suitable concept for the root node of the facet using the […] button. The "Select Concept" Pop up window will appear (see Figure 33 Select facet pop up window).
- The user has to browse the concept hierarchy and select the appropriate concept that would describe the domain.

- The user press the OK button of the "Select Concept" window, the window will disappear and the selected concept will appear in the domain input field. The corresponding natural language description of the concept will be shown below.
- If the desired concept doesn't exist yet, the user can create a new synset/concept by clicking on the ✚ button from the "Select Concept" Pop up window which will bring him to the UI for the linguistic and ontological part.
- Finally, the user clicks on "Create Facet" button from the CUD panel.

(6) **Post state change of the UI:** after pressing the Create facet button the following changes will took place:

- The Facet hierarchy panel will be refreshed by showing the root node of the created facet.
- If any domain is selected the system will prompt the user asking that if he want to add this new facet to the selected domain. If the user get agree the system initiate to add this facet to the domain, remember that the user have to select also a category (e.g., E,R,A).

Notice that the new facet is not associated to any domain yet.

9.8.8 Delete a Facet

To delete a facet the user first has to select the root node of the facet from the "Facets hierarchy" panel. This will activate the "Manage Facet Relation" segment of the CUD panel.

(1) **Operation description:**

- The selected node will appear in the "Source Node" field of the "Manage Facet" segment followed by a ✖ button.
- Clicking on ✖ button will delete the facet from the facet hierarchy. However the corresponding concept will remain intact. This causes also the removal of the facet from all the domains where it is used. A confirmation window will be shown asking the user for a confirmation.

(2) **Post state change of the UI:** after pressing Delete facet button the following changes will took place:

- Facets hierarchy panel will be refreshed with blank information.
- The facet will be removed from the facet list.

Note that, deleting a facet will delete only the facet but not the corresponding concept from the ontological part that were representing the facet.

9.8.9 Add a node to a facet hierarchy

By selecting a node in a facet, called *source node*, the user can add a new child node to it, called the *target node*, using the "Manage Facet Relations" (see Figure 34 CUD Panel) segment of the CUD panel. To do this the user must proceed as follows:

(1) **Information to provide:** Mandatory information:

- Source node: The user must select a source node from the facet hierarchy.
- Target node: The selection of the node is done from a dropdown list. The dropdown list will show only the concepts that are targets of the *source node* under the is-a and part-of relations only. If the desired target doesn't exist yet, the user can create a new concept or relation by clicking on the ✚ button which will bring the user to the UI for the linguistic and ontological part.

(2) Operation description:

- The user selects a *source* node from the facet hierarchy. As a consequence, the "Manage Facet relations" segment of the CUD panel will be initialized with the information of the selected node as follows:
  1. The source node (the selected node) will appear in the "Source Node" field.
  2. The "Target Node" dropdown list will be populated with the existing targets of the source node under the is-a and part-of relations taken from the linguistic and ontological part and not used yet in the facet. The kind of relation must be shown in the usual way.
- The user selects a node from the already populated dropdown list.If the desired target node or the relation does not exist yet, the user clicks on the add button ✚ just beside the dropdown target list which will bring the user to Linguistic and Ontological part for their creation.
- Finally the user presses the Add Relation button and the target node will be added as a child node of the selected source node in the facet hierarchy.

(3) **Post state change of the UI:** after pressing the Add button the following changes will took place:

- The new child node will appear in the facet hierarchy panel.

9.8.10 Remove a node from a facet hierarchy

By selecting a node from the facet hierarchy it is possible to remove it from the facet (along with the relation connecting it to the parent). This is done in the same way as described in Section 8.8 with the difference that the selected node is a generic node. Note that this operation does not cause the removal of the relation from the ontological part. Note also that the removal of a node causes the removal of the whole subtree from the facet. Therefore, a notification will be given to the user to decide whether to confirm the operation or not.

Chapter 10

10. Entity Type Management Consol

10.1. Functional overview and requirement description

In chapter 6 we have described the theoretical background of Entity type and attribute definitions and also identified a set of basic requirements those should support by the developed system. Therefore, starting from the previous discussion we can identify the following sub-tasks that would support by the system:

- **Building Attribute definition**: An attribute definition provides a template for the creation of instance of a given data type. Attribute definition will be manually made according to the theoretical approach described in chapter 6.
- **Building Entity type definition**: an entity type describes instances of a particular class by defining the attributes will be manually made according to the theoretical approach described chapter 6. As a first step we would put in the Entity system only the attribute definitions. Subsequently, we can then define etypes using these attributes.
- **Finding a suitable visualization technique to represent Etype and attributes** : We need to investigate how to deal with the enhanced complexity due to the multilingual etypes and attributes, which facilitates the comparison among aligned languages
- **Managing Attributes :** While editing attribute, the system should takes in to account different uses of etypes in the different *etypes*.
- **Managing Etypes:** The system takes in to account while editing Etype with particular emphasis to their uses. For example, when we delete a etype , we have to take into account if the etype is used by any Entity. Its deletion should be forbidden if it is in use.
- **Performing search and navigation:** Developing an efficient multilingual search and navigation environment to realize the potential benefits of Entity type system.

From an implementation point of view, the abovementioned tasks can be categorized into two different parts:

- **Search and navigation part:** It includes Search, browsing attributes, etypes and its values such as concepts from linguistic part. More in details:
  e. **Search and navigate attributes**: The list (in alphabetical order) of already defined attributes must be available on request to the user. A text search facility to search among them must be available.
  f. **Search and navigate etypes**: The list (in alphabetical order) of already defined etypes must be available on request to the user. A text search facility to search among them must be available.
  g. **Search and navigate attributes of a etype:** By selecting a etype, it must be possible to list the associated attributes of that etype together with the corresponding values of those attributes.
- **Maintenance part:** e.g., creation, updating and deletion operations. This part is, in turn, articulated into operations on Etype and attribute definitions:
  a. **Operations on attributes**: This is the list of operations that the user interface must support on attributes (not necessarily associated to an etype):
    i. *Create a attribute*: An attribute definition is created by selecting the corresponding concept. At this purpose, the UI must provide a facility to search for a suitable concept

by typing the name of the attribute. It must be possible to create a new concept in case no suitable concept is found for it. The data type of the attribute is selected among the predefined ones (see chapter 6).

ii. *Set/ Update the attribute Data type of an attribute* : When selecting an attribute, it must be possible to change the name/concept of an attribute.

iii. *Delete an Attribute* : By selecting an attribute definition, it must be possible to delete it. The system takes in to account its use while deleting attribute.

b. **Operations on etypes** : This is the list of operations that the user interface must support on an etypes:

i. *Create an eType* : An eType is created by selecting the corresponding concept. At this purpose, the UI must provide a facility to search for a suitable concept by typing the name. It must be possible to create a new concept in case no suitable concept is found for it.

ii. *Update a eType*: When selecting an eType, it must be possible to change the representing concept of an existing eType.

iii. *Remove an Attribute of an eType*: It must be possible to select an attribute from the eType and remove it from the eType. This does not result in the deletion of the attribute.

iv. *Add an Attribute to an eType*: It must be possible to search for defined attributes (regardless if they are already associated to other eTypes), select one attribute definition and add it to an eType.

v. *Set/Update the mandatory value of an attribute of an Etype*: It must be possible to change the mandatory value of an Etype.

vi. *Set/Update the Set value of an Etype*: It must be possible to change the Set value of an Etype.

vii. *Set/Update the domain restriction of an attribute of an Etype*: It must be possible to set the domain restriction value(s) of an Etype in a convenience way. Note that in case of relational attribute the values must correspond to existing Entities.

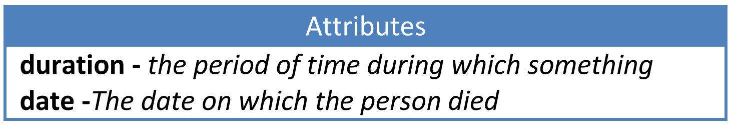
All operations of creation of new attributes and etypes, updating, deletions, search and navigation will be pursued in the Knowledge Base through an Inference engine. In subsequent sections, we will make more precise design specifications for a user-friendly management system to perform basic CRUD operations including search and navigation on the Entity type part for the Universal Knowledge

10.2. Search and navigation

In designing entity type search and navigation we tried to use the similar technique that we have used for the other systems such as domain and linguistic. This will allow the user not to learn every system from the scratch.

10.2.1 Search and navigation of attributes

The idea is to show the search result as an alphabetically ranked list of attributes together with their descriptions, as shown in Figure 2, thus combining the use of keyboard and a scrolling list to make navigation more efficient. Basically, a specific attribute can be singled out easily from the alphabetically ranked list as they have also description attached.



Attributes
duration - *the period of time during which something*
date -*The date on which the person died*

description - *a statement that represents something in*

10.3. Search and navigation of eTypes

We can implement the search and navigation of *etype* in the same way we did for search and navigation of *attributes*

Entity Types
person - *a human being; "there was too much for one person to do"*
organization - *a group of people who work together*

10.4. Search and navigate attributes of a etype

Each etype consists of a set of attributes. To present etype, attributes of the etypes and their values we can use a three columns panel similar to linguistic and domain part, where the first column is for etype list, the second column is for attributes corresponding to the etype and the third column is for values corresponding to the attribute. Each etype from the first column can be treated as a starting node. Clicking on a etype would initiate the query and show a list of attributes for the etype that has been clicked therefore passing to an 'exploded' state in the second column. Clicking on a attribute would show the corresponding value, therefore passing to an 'exploded' state in the third column.

10.5. Management functionalities (create/update/delete)

In the following sections, we describe in detail the CUD operations on attributes , etypes and the attributes of etype. Like as search and navigation, type of operation also varies from object to object. "CUD" operations can be divided into four different parts where each part is responsible for a set of related functionalities:

- **Components of Etype**: We identified the following components those are directly related with a etype:
    c. A concept that represents the etype. The concept must exist.

    d. A set of attributes associated with the etype.
- **Components of Attribute** : We identified the following components those are directly related with attribute:
    b. A concept that represents the attribute. The concept must exist. Notice that, this attribute could be an isolated attribute and not associate to any etype yet.

    c. Any data type defined under etype model (see chapter 6 for details)
- **Components of an attribute of an Etype** : We identified the following components those are directly related with relation:
    c. A meta attribute , possible values are "Strictly Mandatory" , "Mandatory" or "Suggested"

    d. A Boolean value that will indicate if the attribute can be instantiated to a set of values or not.

    e. May have a set of concepts that will determine the boundary of the attribute as a domain restriction.

10.6. General overview of the console

The console consists of two different Tabs : 1) Etype navigation and 2) Etype management

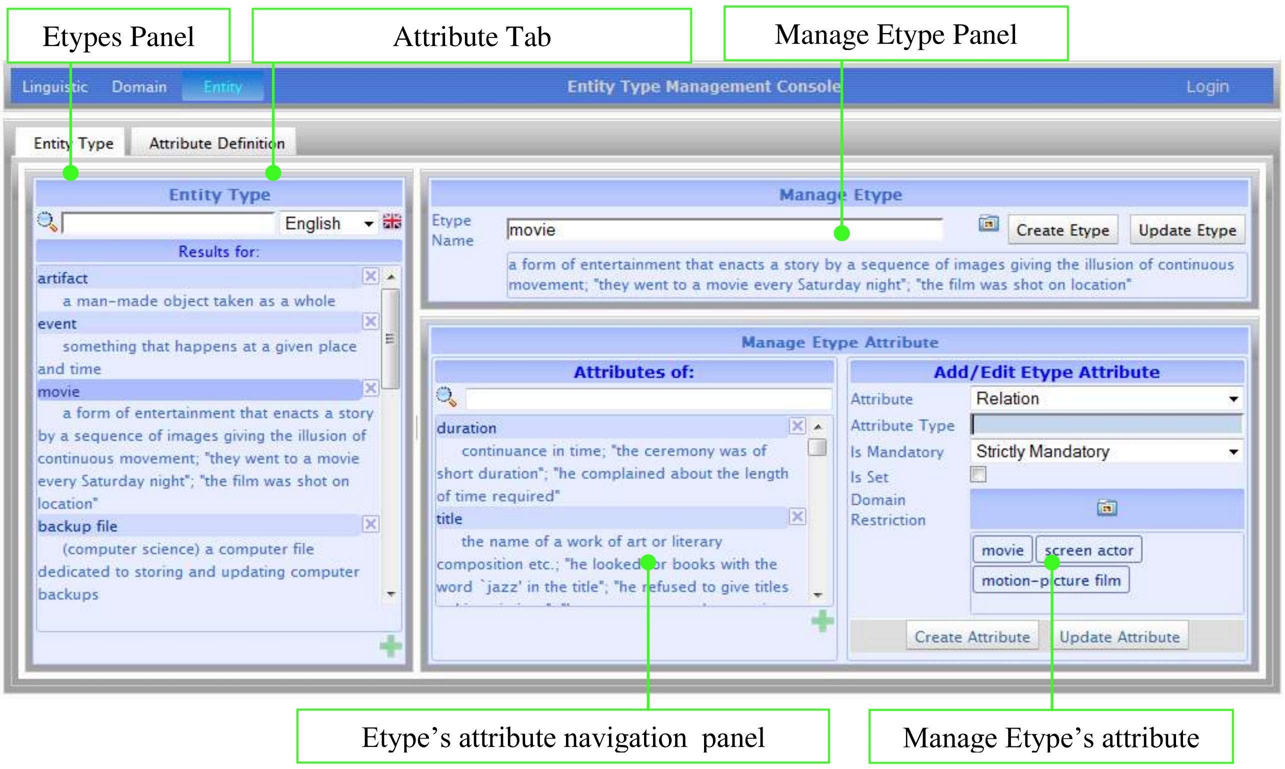
10.6.1 Etype navigation and management Tabs

Etype navigation and management tab dedicated to search, navigation and management of Etype. This tab further divided into three different panels namely:

a. **Entity Type search and navigation**: this panel provides the facilities for Etype search, language selection, and a search result list based on the searched Etype. Search result list is used to navigate through the Etypes and provide a means for selecting query scope.

b. **Manage Etype**: This panel is to create new Etype. However the result of this panel also depends on the search Etype in *Etype panel* while a Etype is selected for update.

c. **Manage Etype Attribute (Create/Update)**: The result of this panel depends on the search Etype in *Etype panel*. In general, it will show a list of attributes for the Etype that has been searched.



artifact
a man-made object taken as a whole
event
something that happens at a given place and time
movie
a form of entertainment that enacts a story by a sequence of images giving the illusion of continuous movement; "they went to a movie every Saturday night"; "the film was shot on location"
backup file
(computer science) a computer file dedicated to storing and updating computer backups

Etype Name: movie
a form of entertainment that enacts a story by a sequence of images giving the illusion of continuous movement; "they went to a movie every Saturday night"; "the film was shot on location"

duration
continuance in time; "the ceremony was of short duration"; "he complained about the length of time required"
title
the name of a work of art or literary composition etc.; "he looked for books with the word `jazz' in the title"; "he refused to give titles

Attribute: Relation
Is Mandatory: Strictly Mandatory
movie | screen actor | motion-picture film

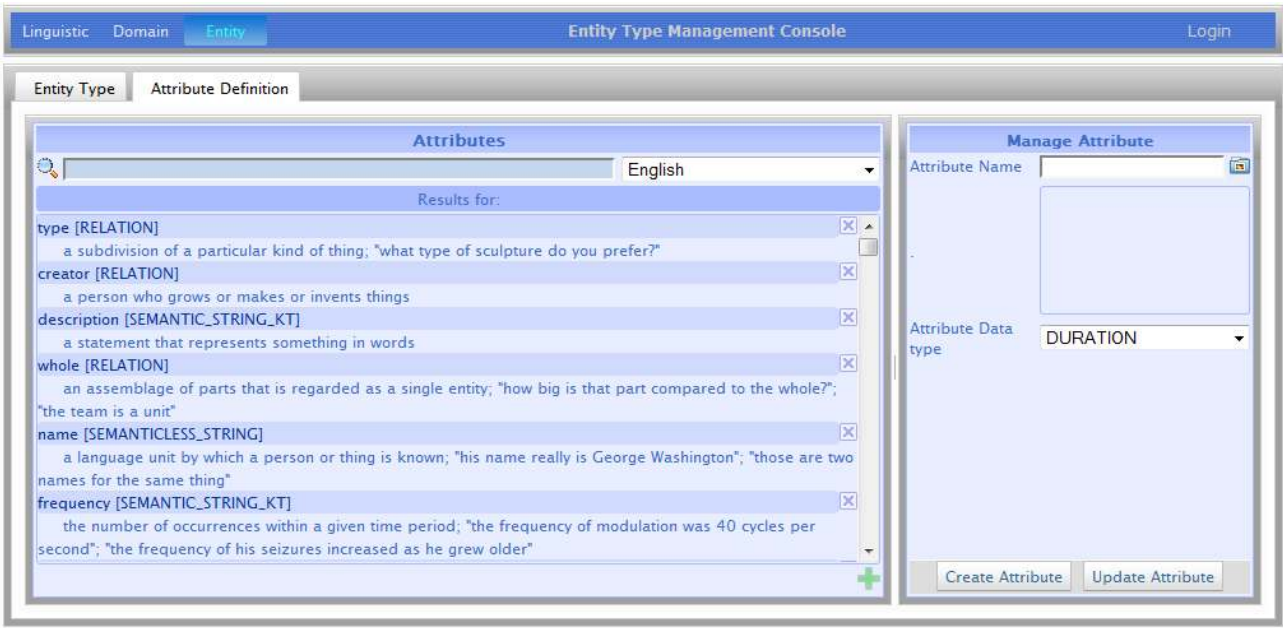
Figure 35 Etype Management Console

10.6.2 Attribute navigation and management Tab

Attribute navigation and management Tab is dedicated to search ,navigation and management of attribute. This tab further divided into two different panels namely:

a. **Attributes search and navigate panel**: This panel provides the facilities for attribute search, language selection, autosuggestions, and a search result list based on the searched attribute. Search result list is used to navigate through the attributes and provide a means for delete and update attribute.

b. **Manage Attribute (Create/Update)**: result of this panel depends on the attribute selection from the attribute panel. This panel is dedicated to create and update operations on attributes.


Linguistic Domain Entity Entity Type Management Console Login

Entity Type | Attribute Definition

Attributes

English

Results for:

type [RELATION]
a subdivision of a particular kind of thing; "what type of sculpture do you prefer?"
creator [RELATION]
a person who grows or makes or invents things
description [SEMANTIC_STRING_KT]
a statement that represents something in words
whole [RELATION]
an assemblage of parts that is regarded as a single entity; "how big is that part compared to the whole?"; "the team is a unit"
name [SEMANTICLESS_STRING]
a language unit by which a person or thing is known; "his name really is George Washington"; "those are two names for the same thing"
frequency [SEMANTIC_STRING_KT]
the number of occurrences within a given time period; "the frequency of modulation was 40 cycles per second"; "the frequency of his seizures increased as he grew older"

Manage Attribute

Attribute Name:

Attribute Data type: DURATION

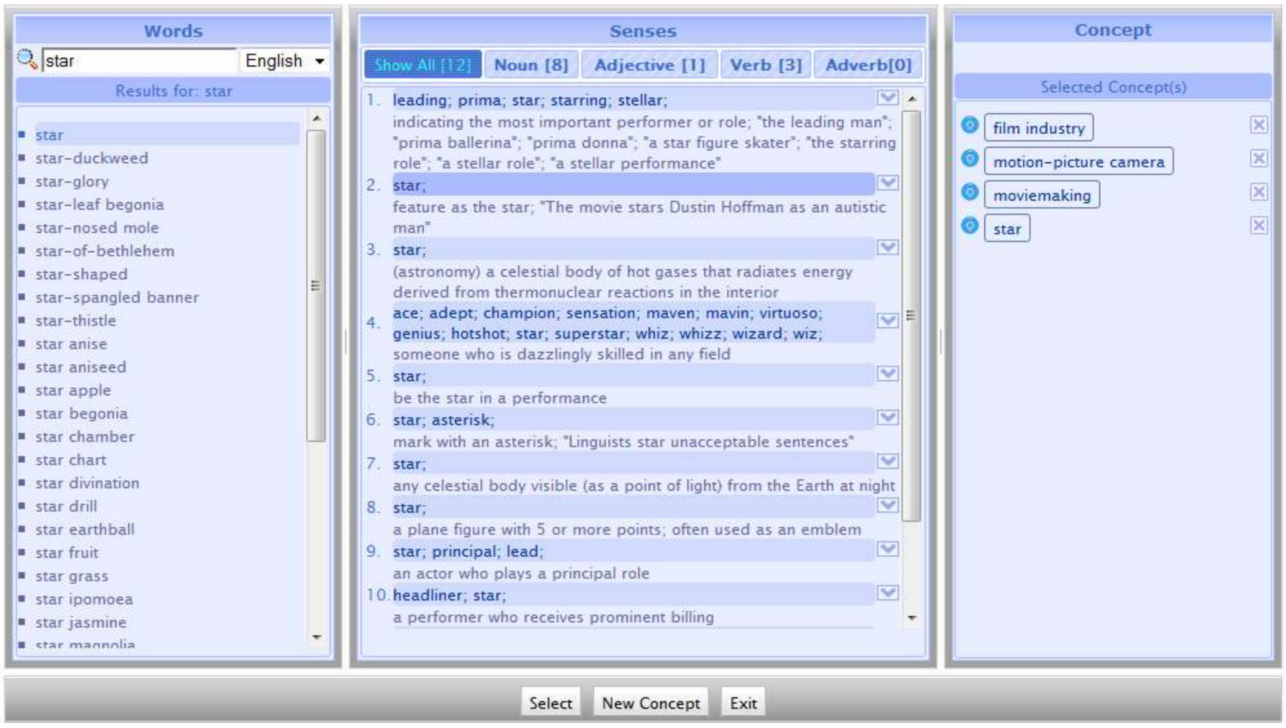
Create Attribute | Update Attribute

Figure 36 Etype Attribute Management Console

10.6.3 Pop Up window for concept selection

See chapter 9. Domain Management section (Figure 32 Select concept selection pop up window) Pop Up window for concept selection

10.6.4 Pop Up window for selecting domain restriction


Words

star | English

Results for: star

- star
- star-duckweed
- star-glory
- star-leaf begonia
- star-nosed mole
- star-of-bethlehem
- star-shaped
- star-spangled banner
- star-thistle
- star anise
- star aniseed
- star apple
- star begonia
- star chamber
- star chart
- star divination
- star drill
- star earthball
- star fruit
- star grass
- star ipomoea
- star jasmine
- star magnolia

Senses

Show All [12] | Noun [8] | Adjective [1] | Verb [3] | Adverb[0]

1. leading; prima; star; starring; stellar;
indicating the most important performer or role; "the leading man"; "prima ballerina"; "prima donna"; "a star figure skater"; "the starring role"; "a stellar role"; "a stellar performance"
2. star;
feature as the star; "The movie stars Dustin Hoffman as an autistic man"
3. star;
(astronomy) a celestial body of hot gases that radiates energy derived from thermonuclear reactions in the interior
4. ace; adept; champion; sensation; maven; mavin; virtuoso; genius; hotshot; star; superstar; whiz; whizz; wizard; wiz;
someone who is dazzlingly skilled in any field
5. star;
be the star in a performance
6. star; asterisk;
mark with an asterisk; "Linguists star unacceptable sentences"
7. star;
any celestial body visible (as a point of light) from the Earth at night
8. star;
a plane figure with 5 or more points; often used as an emblem
9. star; principal; lead;
an actor who plays a principal role
10. headliner; star;
a performer who receives prominent billing

Concept

Selected Concept(s)

- film industry
- motion-picture camera
- moviemaking
- star

Select | New Concept | Exit

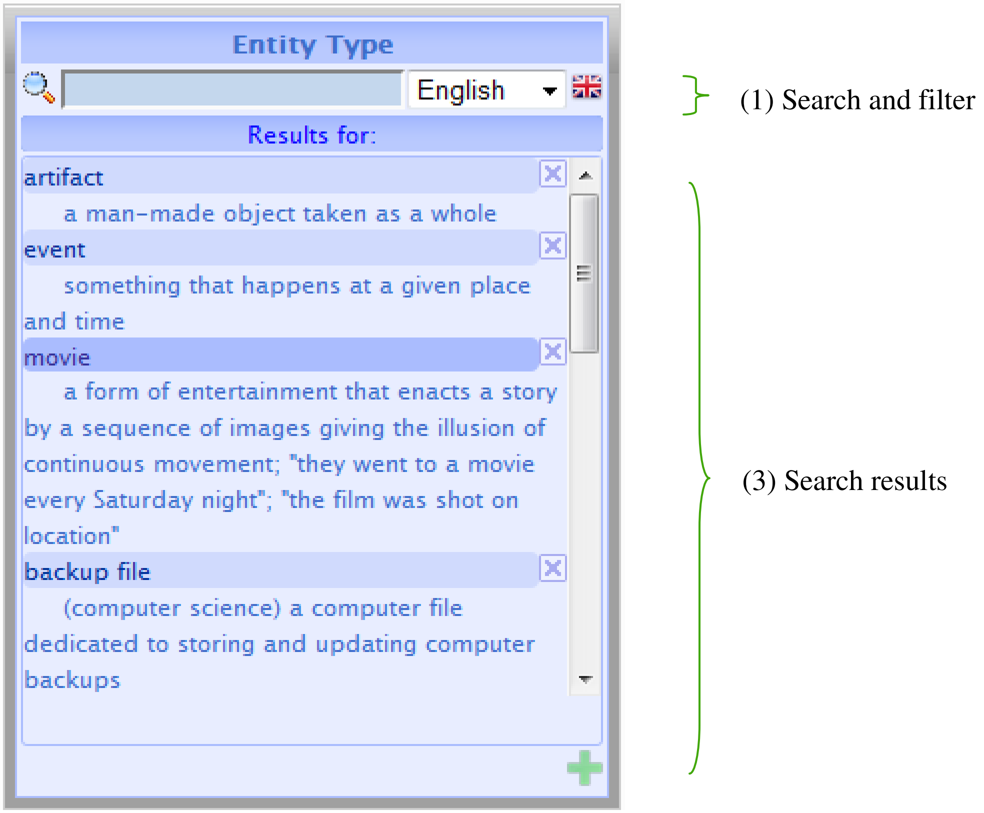
Figure 37 Domain Select Window

Domain Select window (see Figure 37 Domain Select Window) enables the user to select a list of concepts from the Linguistic and Ontological part that would be used to impose as domain restriction of the an Etype attribute. Only an already existing concept can be selected from this pop up window. If the concept does not exist yet, the user can create it by clicking on the "New Concept" button which will bring the user to the UI for the Linguistic and Ontological part. The right most concept panel (see Figure 37 Domain Select Window) shows the list of selected concept those are chosen as a domain restriction for an Etype attribute

10.7. Detailed description of Etype navigation and management Tab

10.7.1 Etypes panel

This panel provides the facilities for Etype search and filtering based on language selection, a sorted search result list.



artifact
a man-made object taken as a whole
event
something that happens at a given place and time
movie
a form of entertainment that enacts a story by a sequence of images giving the illusion of continuous movement; "they went to a movie every Saturday night"; "the film was shot on location"
backup file
(computer science) a computer file dedicated to storing and updating computer backups

Figure 38 Etypes Panel

(1) **Search and filtering** input panel consists of the following components:

3. *Language selection combo*: A dropdown list box for selecting preferred language. When the user selects a different language, all information in the UI remains the same but it is shown in the selected language.
4. *A text box for free text Etype name input*: figure 38 shows a situation in which a user is typing an etype in the Etype input text box. After typing a Etype and the user presses the enter key, two possible scenarios could take place:
   - The Etype exists: a list of results will appear in the search result list box in alphabetical order together with the searched Etype; The user can select a Etype from the list. Attribute(s) for that selected Etype would appear as a list in the Etype Attributes panel for that Etype.
   - The Etype does not exist: it can occur due to misspelling or language setting or simply because the Etype does not exist in the KB. In this case, autosuggestion panel will show a suggestion.

(2) **Search result**: Search result is shown as a list in the search result list box in alphabetical order together with the searched Etype. It is a list representing the filtered Etypes (according to the selected language and searched text). Results are presented as blocks of information in a linear list.

Each block represents a Etype, Each block contains the Etype name together with its corresponding gloss.

(3) **Delete an Etype.** User can delete an Etype by clicking on the cross ☒ button attached with the Etype block. Note that, the operations will be applicable only on the "selected" Etype from the Etype panel.

(4) **Adding a new Etype**: In the bottom of the Etypes panel there is an add button ✚ which is for adding a new Etype. This functionality is described in detail in Section 6.

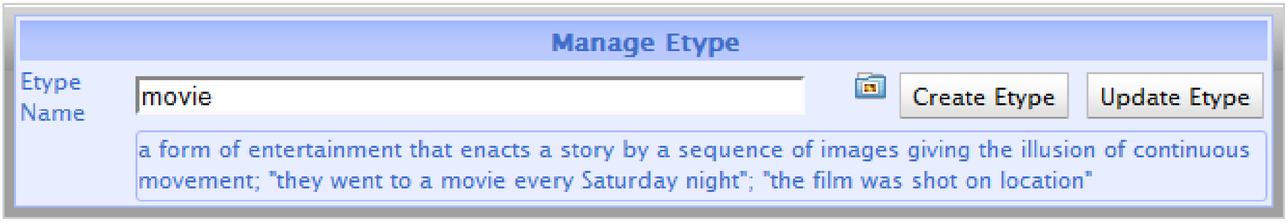
10.7.2 Manage Etype panel

The Manage Etype panel is dedicated to the management of the information related to Etypes. It consists of the following components:

e. **Etype Name**: a text field for entering a Entering a etype name

f. **Concept brows button**: **[…]** a brows button. Clicking on it will open a popup window "Select Concept" window (see Figure 32 Select concept selection pop up window) for selecting the concept used to describe the Etype..

g. Finally two buttons: "Create Etype" and "Update Etype". One of the two buttons would be disabled according to the cases.



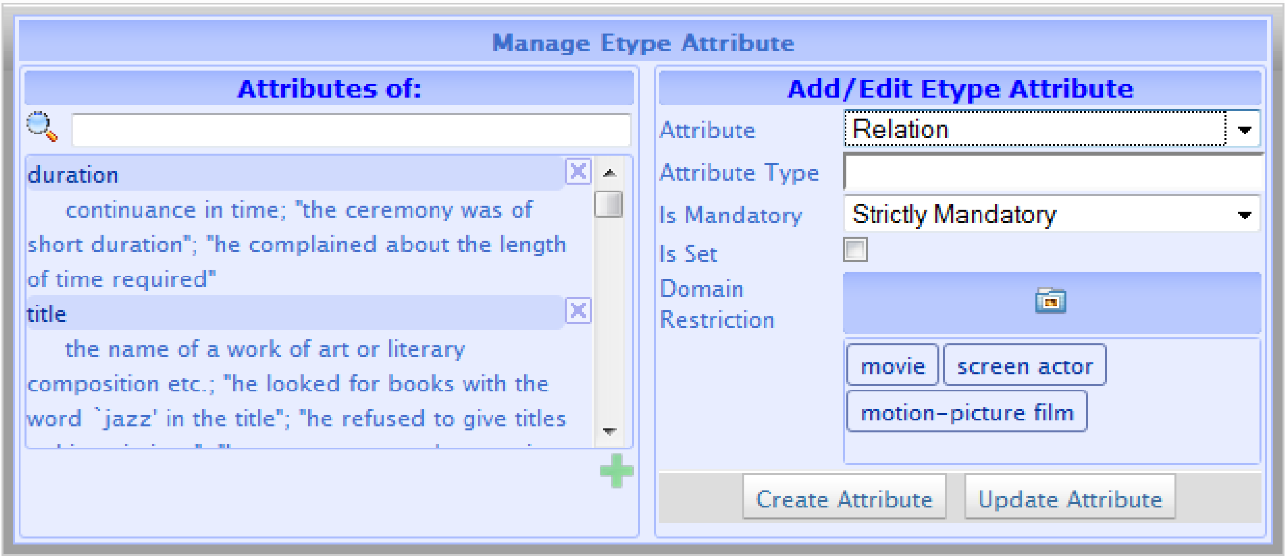
Etype Name: movie

a form of entertainment that enacts a story by a sequence of images giving the illusion of continuous movement; "they went to a movie every Saturday night"; "the film was shot on location"

Figure 39 Manage Etype Panel

10.7.3 Manage Etype Attribute panel

Manage Etype is further divided into two segments, namely "Attributes", "Add/Edit Etype Attribute.



duration

continuance in time; "the ceremony was of short duration"; "he complained about the length of time required"

title

the name of a work of art or literary composition etc.; "he looked for books with the word `jazz' in the title"; "he refused to give titles

Attribute: Relation

Attribute Type:

Is Mandatory: Strictly Mandatory

Is Set: ☐

Domain Restriction: movie, screen actor, motion-picture film

Figure 40 Manage Etype Attribute Panel

(1) **Etype Attributes***:* this panel is organized as a list of attributes. The list for presenting search results - i.e. the attributes associated to the selected Etype - consists of a set of blocks, each block representing a Attribute. Each block also contains a delete button to remove the attribute from the associated Etype. Finally, at the bottom of the result list, a plus button can be used to add a attribute (i.e. associate an existing attribute) to the current selected Etype.

(2) **Add/Edit Etype Attribute :** The second segment of the Manage Etype Attribute Panel is to associate an existing attribute to an Etype. It is dedicated to the management of the information directly related to the attribute within a Etype. It consists of the following components:

a. **Attribute**: a dropdown box for selecting a attribute from the existing attributes

b. **Attribute type**: a non editable text area that would show the attribute type of the selected attribute

c. **Is Mandatory**: a dropdown list box for selecting if this attributes is mandatory for the Etype or not ; possible values are "Strictly Mandatory", "Mandatory" and "Suggested".

d. Is Set: isSet is a Boolean value that indicates whether the attribute can be instantiated to a set of values.

e. **[…]** a browse button. Clicking on it will open a popup window "Select Concepts" window (see Figure 32 Select concept selection pop up window) for selecting the concept or a set of concepts to which will impose the domain restriction e.g., to define a specific sub set of values redefining the domain of the data type, putting a set of restriction on the domain of possible values. Table 3 provide the currently supported domain restrictions

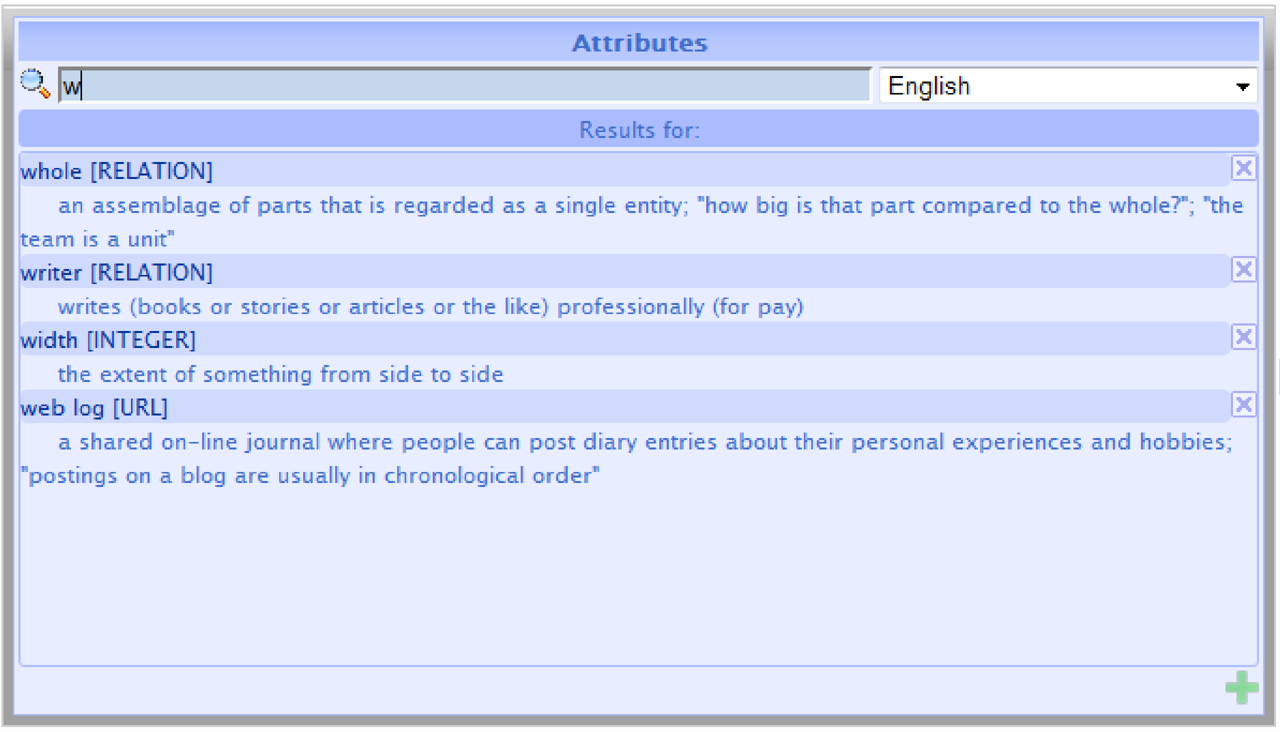
f. "Add Attribute" button to confirm the association of the attribute to the selected domain.

g. "Update Facet" button to change the category of a facet.

10.8. Detailed description of Attribute navigation and management Tab

10.8.1 Attributes search and navigation panel

This panel provides the facilities for Attribute search and filtering based on language selection, autosuggestions, and a sorted search result list



whole [RELATION]
an assemblage of parts that is regarded as a single entity; "how big is that part compared to the whole?"; "the team is a unit"

writer [RELATION]
writes (books or stories or articles or the like) professionally (for pay)

width [INTEGER]
the extent of something from side to side

web log [URL]
a shared on-line journal where people can post diary entries about their personal experiences and hobbies; "postings on a blog are usually in chronological order"

Figure 41 Attribute search and navigation

(1) **Attribute Search and filtering** input panel consists of the following components:

a) *Language selection combo*: A dropdown list box for selecting preferred language. When the user selects a different language, all information in the UI remains the same but it is shown in the selected language.

b) *A text box for free text Attribute name input*: figure 41 shows a situation in which a user is typing an Attribute in the Attribute input text box. After typing a Attribute and the user presses the enter key, two possible scenarios could take place:

- The Attribute exists: a list of results will appear in the search result list box in alphabetical order together with the searched Attribute; The user can select a Attribute from the list. Selected Attribute would appear in the Manage Attributes panel for performing Update operation.

- The Attribute does not exist: it can occur due to misspelling or language setting or simply because the Attribute does not exist in the KB. In this case, autosuggestion panel will show a suggestion.

(2) **Search result**: Search result is shown as a list in the search result list box in alphabetical order together with the searched Attribute. It is a list representing the filtered Attribute (according to the selected language and searched text). Results are presented as blocks of information in a linear list. Each block represents a Attribute. Each block contains the Attribute name together with its corresponding gloss.

(3) **Delete an Attribute.** User can delete an Attribute by clicking on the cross ☒ button attached with the Attribute block. Note that, the operations will be applicable only on the "selected" Attribute from the Attribute panel.

(4) **Adding a new Attribute**: In the bottom of the Attributes panel there is an add button ➕ which is for adding a new Attribute. This functionality is described in detail in later Sections.

10.8.2 Add/Edit Attribute panel

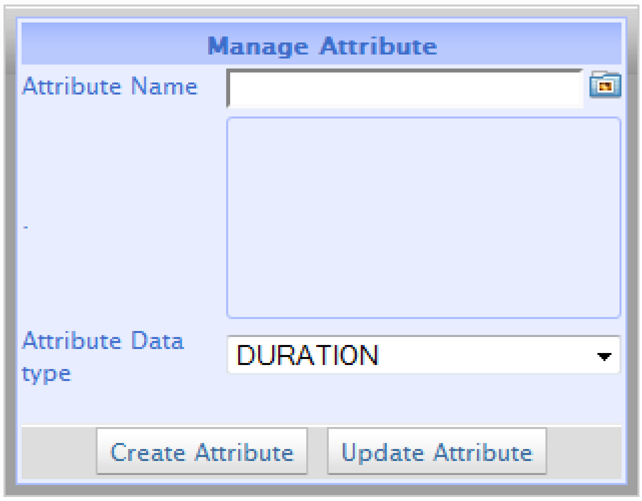


Figure 42 Manage Attribute

The second segment of the Attribute Tab is Add/Edit Attribute Panel. It is dedicated to the management of the Attribute e.g., create new attribute, update existing attribute. It consists of the following components:

d. **Attribute Name**: a text area for entering the attribute (it must be selected from the existing concept using "Select Concept" window)

e. **Attribute Data Type**: a dropdown list box for selecting Data Type; possible values are "Boolean" , "Integer", "Float", "Moment"," Duration"," Semantic-Less String", Semantic "String", "Entity" and "URL".

f. **[…]** a browse button. Clicking on it will open a popup window "Select Concept" window for selecting the concept that would be used for the Attribute.

g. "Create Attribute" button to create a new attribute.

h. "Update Attribute" button to change the data type of a attribute.

10.9. Manage Attribute

It is important to underline that for Etype the basic building block is Attribute. The user can create Attributes only using the existing concepts built in linguistic and ontological part. In this section we provide the details on how the operations described in section 3 are performed.

10.9.1 Create an Attribute

An Attribute is created by selecting the corresponding concept. The concept must exist in the Linguistic and Ontological part. At this purpose, the "**Select Concept**" window (see Figure 32 Select concept selection pop up window) provides a facility to search for a suitable concept by typing the name of the Attribute. In case no suitable concept is found for it the user can create a new concept clicking on the ✚ button of the "**Select Concept**" window that will bring the user to the UI for the linguistic and ontological part to create new concept.

(1) **Information to provide:** It is mandatory to provide the synset/concept associated to the Attribute and it must exist.

(2) **Operation description:**

The user first clicks on the add Attribute ✚ button located on the bottom of the Attributes panel. Clicking on add Attribute ✚ button will enable the "Manage Attribute" panel with blank information. Only the **Create Attribute** button of the "Manage Attribute" panel will be enabled. All other buttons will be disabled. User performs the following steps to create the Etype:

- The user must search for a suitable concept using the […] button. The "Select Concept" Pop up window will appear.
- The user has to browse the concept hierarchy and select the appropriate concept that would describe the Attribute.
- The user press the OK button of the "Select Concept" window, the window will disappear and the selected concept will appear in the Attribute name input field. The corresponding natural language description of the concept will be shown below.
- If the desired concept doesn't exist yet, the user can create a new synset/concept by clicking on the ✚ button from the "Select Concept" Pop up window which will bring him to the UI for the linguistic and ontological part.
- Finally, the Create Attribute button has to be pressed for the final commit.

(3) **Post state change of the UI:** after pressing the Create Attribute button the following changes will take place:

CUD panel will be refreshed. The new Attribute will appear in the list of Attributes.

10.9.2 Update an Attribute

Update an Attribute means changing the concept that is used to describe the attribute. To update a attribute the user has to select the attribute from the attribute list of the Attributes panel. The selected attribute will appear in the attribute field of the "Manage Attribute" panel. Only the Update attribute button will be enabled. The user can change the concept using the "Select Concept" pop up window. Finally the user clicks on "Update Attribute" button to commit the change.

(1) **Information to provide:** It is mandatory to provide the synset/concept associated to the Etype and it must exist.

(2) **Post state change of the UI:** after pressing the Add button the following changes will took place:

Etype panel will be refreshed. The updated Etype will appear in the Etypes list.

10.9.3 Delete an Attribute

To delete an attribute the user first has to select the attribute from the Attributes panel and click on the delete button ☒ attached with the attribute name.

(7) **Operation description:** Clicking on ☒ button will do the following:

- Delete the attribute however the corresponding synset/concept will remain intact. This does not cause the deletion of the synset/concept associated to it.

(8) **Post state change of the UI:** after pressing Delete attribute button the following changes will took place:

- Attributes panel will be refreshed by removing the Attributes form the Attribute list.

10.10. Managing Etypes and the attributes associated with an Etype

The user can create Etypes only using the existing Attributes. In this section we provide the details on how the operations described in section 3 are performed.

10.10.1 Create an Etype

A Etype is created by selecting the corresponding concept. The concept must exist in the Linguistic and Ontological part. At this purpose, the "**Select Concept**" window (see Figure 32 Select concept selection pop up window) provides a facility to search for a suitable concept by typing the name of the Etype. In case no suitable concept is found for it the user can create a new concept clicking on the ✚ button of the "**Select Concept**" window which will bring the user to the UI for the linguistic and ontological part to create new concept.

(1) **Information to provide:** It is mandatory to provide the synset/concept associated to the Etype and it must exist.

(2) **Operation description:**

The user first clicks on the add Etype ✚ button located on the bottom of the Etypes panel. Clicking on add Etype ✚ button will enable the "Manage Etype" panel with blank information. Only the **Create Etype** button of the "Manage Etype" panel will be enabled. All other buttons will be disabled. User performs the following steps to create the Etype:

- The user must search for a suitable concept using the […] button. The "Select Concept" Pop up window will appear.
- The user has to browse the concept hierarchy and select the appropriate concept that would describe the Etypr.
- The user press the OK button of the "Select Concept" window, the window will disappear and the selected concept will appear in the Etype input field. The corresponding natural language description of the concept will be shown below.
- If the desired concept doesn't exist yet, the user can create a new synset/concept by clicking on the ✚ button from the "Select Concept" Pop up window which will bring him to the UI for the linguistic and ontological part.
- Finally, the Create Etype button has to be pressed for the final commit.

(3) **Post state change of the UI:** after pressing the Create Etype button the following changes will take place:

- CUD panel will be refreshed. The new Etype will appear in the list of Etypes.

10.10.2 Update an Etype

Update a Etype means changing the concept that is used to describe the Etype. To update a Etype the user has to select the Etype from the Etype list of the Etypes panel. The selected domain will appear in the domain field of the "Manage Etype" panel. Only the Update Etype button will be enabled. The user can change the concept using the "Select Concept" pop up window. Finally the user clicks on "Update Etype" button to commit the change.

(1) **Information to provide:** It is mandatory to provide the concept associated to the Etype and it must exist.

(2) **Post state change of the UI:** after pressing the Add button Etype panel will be refreshed. The updated Etype will appear in the Etypes list.

10.10.3 Delete an Etype

To delete a Etype the user first has to select the Etype from the Etypes panel and click on the delete button ☒ attached with the Etype name.

(1) **Operation description:** Clicking on ☒ button will do the following:

- Delete the Etype. However the corresponding synset/concept will remain intact. This does not cause the deletion of the Attributes associated to it.

(2) **Post state change of the UI:** after pressing Delete Etype button the following changes will took place:

- Etype Attributes panel will be refreshed by removing the Attributes form the Attribute list.

10.10.4 Add an attribute to an Etype

The user can associate an attribute to an Etype (regardless if it is already associated to other Etypes), by selecting one attribute and associating it to the Etype by providing necessary constrains.

(1) **Information to provide:** Mandatory information for adding an attribute are:

- Attribute*:* the attribute must exist. The user selects the attribute from the attribute drop down list.
- Is Mandatory: the user has to select a value from the is mandatory dropdown list. Possible values are "Strictly Mandatory" , "Mandatory" or "Suggested"
- Is Set: if the attribute can be instantiated to a set of values than the user have to checked this checkbox otherwise unchecked.
- Domain restriction:

(2) **Operation description:** The user first clicks on the add Attribute ✚ button located on the bottom of the Etype's Attributes panel. Clicking on add Attribute ✚ button will activate the "Add/Edit Etype Attribute" segment of the "Manage Etype Attribute" panel with blank information. Only the **Add Attribute** button of the "Add/Edit Etype Attribute" panel will be enabled.

- The user must select an attribute from the dropdown attribute list.
- The user must select a value from the is mandatory dropdown list. Possible values are "Strictly Mandatory" , "Mandatory" or "Suggested"
- The user selects a facet from the "Select Facet" window and press OK button. If the facet does not exist yet, it has to be created using the "Create Facet" segment of the CUD panel.
- The user checked the checkbox if the attribute can be instantiated to a set of values otherwise unchecked.
- The selected facet will appear in the Facet input field
- If need to set domain restriction:
  - The user must search for a suitable concept(s) using the […] button. The "Select Concepts" Pop up window will appear.
  - The user has to browse the concepts and select the appropriate concept(s) that would use as the domain restriction for a particular attribute of the Etypy.
  - The user press the OK button of the "Select Concepts" window, the window will disappear and the selected concept(s) will appear in the "Domain Restriction" input field.
- Finally, Add Attribute button will be pressed for final commit.

(3) **Post state change of the UI:** after pressing the Add Attribute button the Etype's attribute panel will be refreshed. The new attribute will appear as a block in the Etype's attributes panel.

10.10.5 Update the property of an attribute of an Etype

The user can meta property of an attribute associated with an Etype (regardless if it is already associated to other Etypes), by selecting one attribute an etype and then by changing any of these values:

- Is Mandatory
- Is Set
- Domain restriction

(1) **Operation description:** To update an attribute's property of an etype the user first has to select the attribute from the etype attributes panel.

- The user makes necessary changes any of the above mentioned properties.
- Finally, Update Attribute button will be pressed for final commit.

(2) **Post state change of the UI:** after pressing the Update button the following changes will take place: Etype's attribute panel will be refreshed. The updated Attribute will appear in the Attributes list.

10.10.6 Removing an Attribute from an Etype

To remove a attribute from a etype the user first has to select the attribute from the etype attributes panel and click on the delete button ✖ attached with the attribute block.

(1) **Operation description:** Clicking on ✖ button will do the following:

- Remove the attribute from the etype.

(2) **Post state change of the UI:** after pressing Delete Attribute button the following changes will took place:

- Etype attributes panel will be refreshed by removing the attribute form the attribute list.

Note that, this operation causes the removal of the attribute from the active etype only. Other etypes might use the attribute.

Chapter 11

11. Evaluation for Linguistic and Concept Knowledge Management Consol

Usability evaluation is a core component of user-centered systems design which is a measure of interface quality that refers to the effectiveness and efficiency of a tool and the user's satisfaction with which users can perform tasks with it [101]. In the human-computer interaction domain, interactions involve the user, the tools, and the ways the user and the tools work together. This chapter contains the report of usability evaluation for Linguistic and Concept Knowledge Management Consol. The evaluation was conducted with two domain expert participants. The evaluation was performed to evaluate effectiveness, efficiency and satisfaction.

11.1. Evaluation method

Evaluating a system with a sample of users performing a set of well organize predetermined tasks is usually considered to yield the most realistic and reliable evaluation of a system's usability [101]. This method is known as *User-based* method [101]. This method also gives clear trace of vital problems. Some of the drawbacks of this method stated in [101] are: time consuming, expensive for large sample of users, requires running prototype for evaluation. Lewis [102] shows that the actual number of sample size mainly depend on the type of faults one looks for to discover and their relative probability of occurrence. However properly performed user-based methods with real users always give the truest estimate of a system's efficiency, effectiveness, and satisfactorily. In our evaluation we asked the users to perform a set of well defined tasks with our developed system, during performing the tasks we also have recorded each task completion time to measure the speed of performance. As mentioned above, system testing using user-based method is often constrained due to large sample of users' limitation. This constrain leads to substantial interest among HCI communities in finding how to achieve the maximum information from the smallest sample of users also determining what could the minimum sample size to perform an effective evaluation. There are popular myths that say a greater part of problems is possible to determine with only 2 or 3 users.

Evaluation criteria:

- *Effectiveness*: whether or up to what extent the task is completed successfully.
- *Efficiency*: The time taken to achieve the goals.
- *Satisfaction*: Measured by a questionnaire administered after completion of all the tasks.

Sample of Users:

For a reliable usability test, the test participants (i.e., sample of users) should be representative of the intended users [103]. Our system is particularly suited for the expert users (i.e., domain experts, ontology engineers) devoted in knowledge creation process. However the intended user of our system may or may not have computer expertise. Considering these facts, we have selected two postdoctoral research fellows with library science background as a domain expert.

Evaluation Procedure:

To obtain reliable measures it is essential that the test procedure should be as natural as possible. There were no interactions with the moderator. The participant carried out the tasks alone and they were asked what to do or achieve, but not how to do it. The users were familiar with the structure of the ontology they were asked to work on but we didn't demonstrate the tools before. Our intension was to see also if our system is self-explanatory enough to guide the novice users (i.g., user without any prior knowledge about this tools) to perform the intended task. We prepared task package scenarios to comprehend various operations that can be performed with our system. It has been created in two groups –

[1] **Basic tasks**: navigation on words, senses and concepts. Users were assigned tasks to test the different parts of the system (i.e. search an navigate within the same language, multilingual search and navigation, concept hierarchy browsing etc) and were required to find specific concept to test cross language retrieval from English to Italian.
[2] **Advance tasks**: CUD on Ontology. Users were asked to perform create, update and delete operations.

There were 10 tasks for search and navigation and 15 tasks for CUD. Benchmark time was set by the developer of the application and two expert users participated in performing tasks.

Efficiency was measured in time needed to get an answer from the system and the time taken to complete a CUD operation.

Users' successfully completed tasks were used to assess the effectiveness of the UI presentation. The assumption was that a good self-descriptive presentation would allow the user to complete a high percentage of the CRUD operations those were asked to perform.

In order to measure the user satisfaction level, we used heuristic approach with set of questions. After the tasks are completed, users are asked to provide feedback through a survey questionnaire where they describe in more detail the performance and perceptions of the system. The answers were graded e.g. low, moderate and high. 15 areas and different observations of satisfaction matrix were used for measuring the user satisfaction level.

In this way we have derived the measures of effectiveness, efficiency and satisfaction, based on the evaluation results and advice we identified the problems and determined modifications.

11.2. Performance evaluation

For search and navigations of words, senses and concepts, we put ten tasks. While we present the detail user performance in the graph, the following table shows a comparison of aggregated performance for the given tasks.

	Benchmark	User 1	User 2
Time taken for 10 tasks	01:31.26	05:17.90	03:09.80

Table 5 Aggregate of task package 1, navigation on words, sense and concepts.

While User-2 took longer than User-1, both took almost twice the benchmark time. One of the most important reasons is that the users were not briefed about the UI before the session, nor they were guided to conduct a wild evaluation. Other reasons were revealed in the satisfaction evaluation.

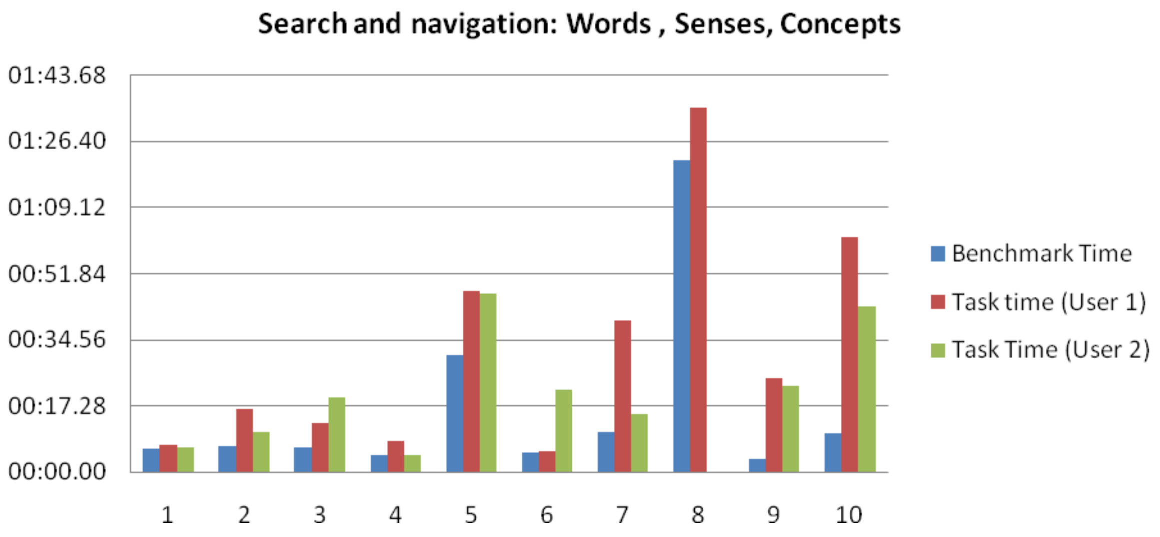


Figure 43 Comparison of performance for each task.

Perhaps task 8 has already taken our attention. The task was to – Find its verb concept where "cat" has more general concept "colonization". We put the task by accidental discovery and found it again. However, User-1 managed to find it, don't know how.

There were 15 main tasks for CRUD on ontology was asked to perform. The aggregate is as follows –

	Benchmark	User 1	User 2
Time taken for 15 tasks	07:58.88	16:12.55	13:51.68

Table 6Aggregate of task package 2, CRUD on ontology

The difference between benchmark time and performed time should have been caused by the same above reason and reasons that shall be discussed later.

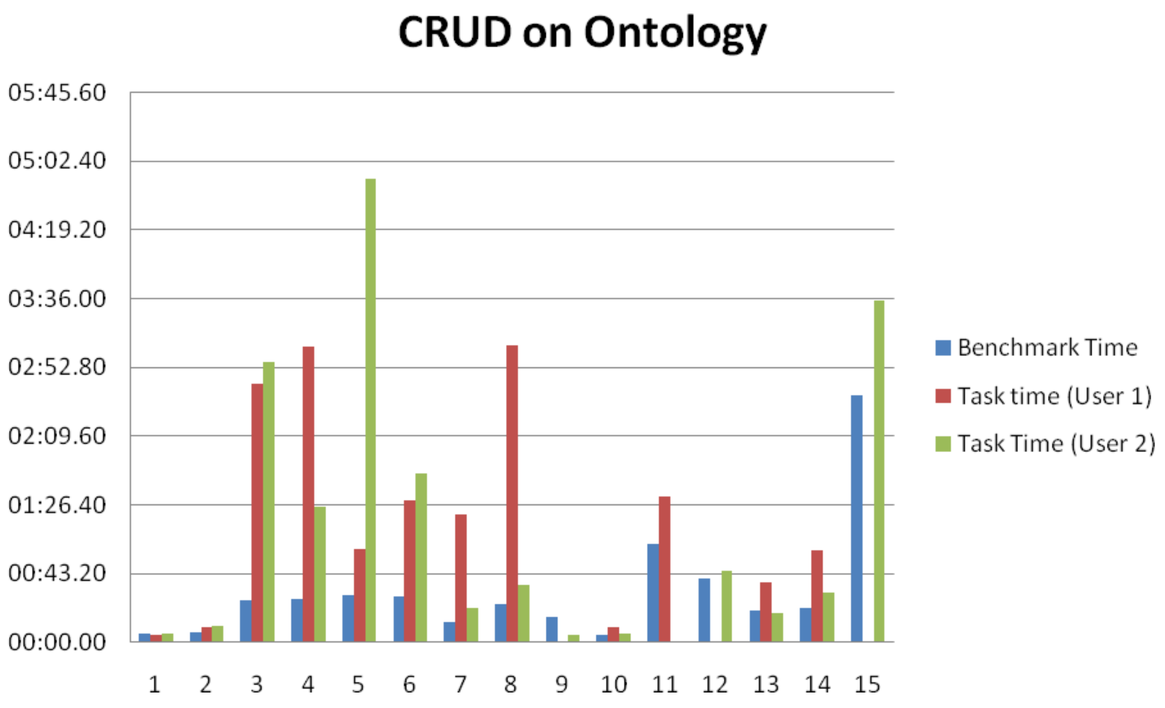


Figure 44 CRUD on ontology.

User-2 took surprisingly longer time for task 5 (i.e., Create a Sense to be associated with a concept with no Sense in the language) and User-1 took longer for task 8 (i.e., Update a Word of a Sense - the properties of a word associated to a Sense). This is probably resulted from the misunderstanding of interface elements which later discussed the in recommendation section.

Given the fact that users had no previous experience of using the system, therefore the task accomplishment time using the entire new system is acceptable.

11.3. Effectiveness evaluation

The number of tasks correctly completed out of the 10 basic and the 15 advanced tasks were used to measure the system effectiveness. This can be seen as a sort of system effectiveness in terms of self descriptiveness of the system on the basis of the total work done by the user. The rationale behind this is that as the system was not demonstrated before to the sample users to see if the system is self-explanatory enough to guide the novice users, therefore the more number of tasks the user could solve would shows the more the system is effective. In this experiment the first 10 basic operations (i..e., search and navigation) are to measure the effectiveness of the display whereas the 15 advanced operations (i.e., CUD) are to measure the effectiveness of the management mechanism of the system.

For the 10 basic operations user-1 was able to complete all the tasks whereas the user-2 left one (i.e., task 8) which indicate the strong effectiveness of the system presentation. On the other hand from the 15 advanced tasks user-1 was able to complete 13 tasks whereas the user-2 completed 14 tasks, given the fact that performing advanced work on an unknown system and completing 87 to 93 percents of work shows the significant effectiveness of the system's management mechanism.

11.4. Satisfaction evaluation

The elements of satisfaction were grouped in to the following categories with the number of observations.

No.	Features of user satisfaction	No. of observations
1	Visibility of System Status	11
2	Match Between System and the Real World	5
3	User Control and Freedom	6
4	Consistency and Standards	6
5	Recover from Errors	4
6	Error Prevention	3
7	Recognition Rather Than Recall	4
8	Flexibility and Minimalist Design	4
9	Aesthetic and Minimalist Design	3
10	Skills (3)	3
11	Pleasurable and Respectful Interaction	2

Table 7 Satisfaction matrix being used.

A total of 51 observations were graded by both users. Following is the summary of users rating.

No.	Feature	Low	Moderate	High
1	Visibility of System Status (11)	0	5	6
2	Match Between System and the Real World (5)	0	2	3
3	User Control and Freedom (6)	1	2	3

4	Consistency and Standards (6)	1	2	3
5	Recover from Errors (4)	1	3	0
6	Error Prevention (3)	0	1	2
7	Recognition Rather Than Recall (4)	0	2	2
8	Flexibility and Minimalist Design (4)	0	2	2
9	Aesthetic and Minimalist Design (3)	1	2	0
10	Skills (3)	0	2	1
11	Pleasurable and Respectful Interaction (2)	0	0	2
	Total	4	23	24

Table 8 This summary table shows a very high satisfaction level of using the system.

Though the above summary shows a very satisfaction level, but some individual observations proved fatal in the system and added in recommendation section. The relevant judgments were analyzed and were taken into account when the final version of the system was released. Following graphs shows the rating by two users for each feature. To understand the graph, consider the feature 1 (Figure 45 Features rated by User 2) where there were 11 observations. Out those 11, 2 graded as ‘Low”, 2 graded as “Moderate” and 7 graded as “High”. For this feature, the system scores 2.45 out of 3, which is quiet, a high rating!

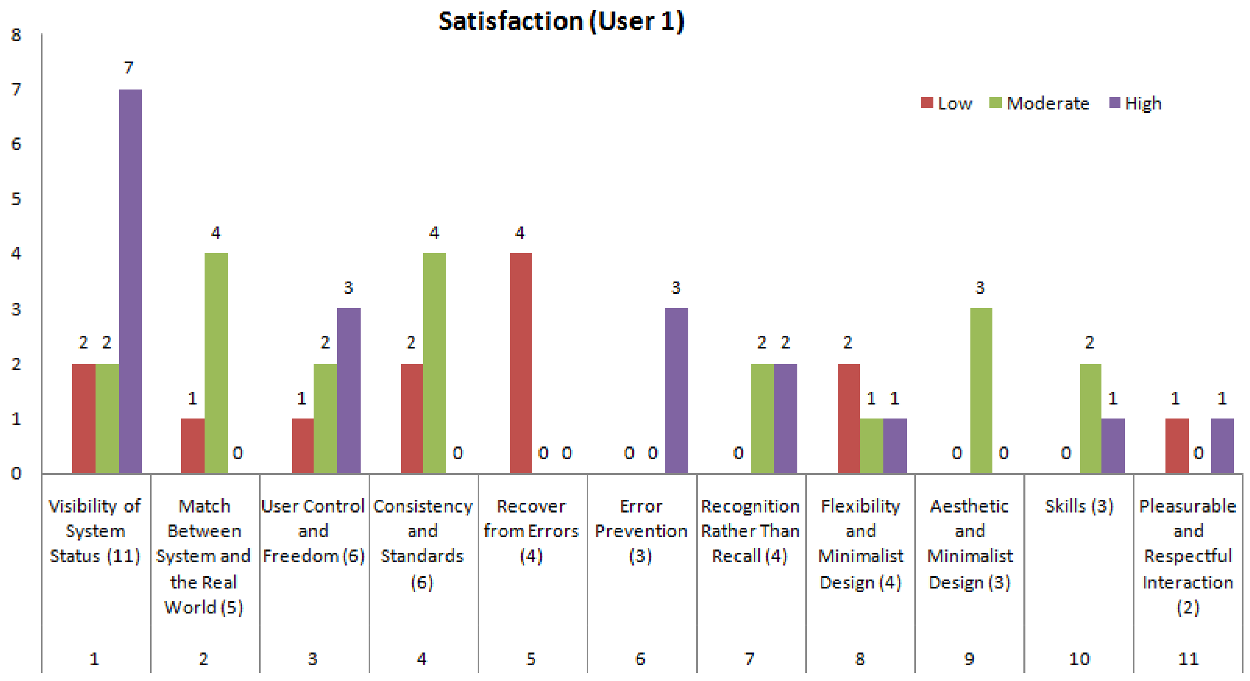


Figure 45 Features rated by User 2

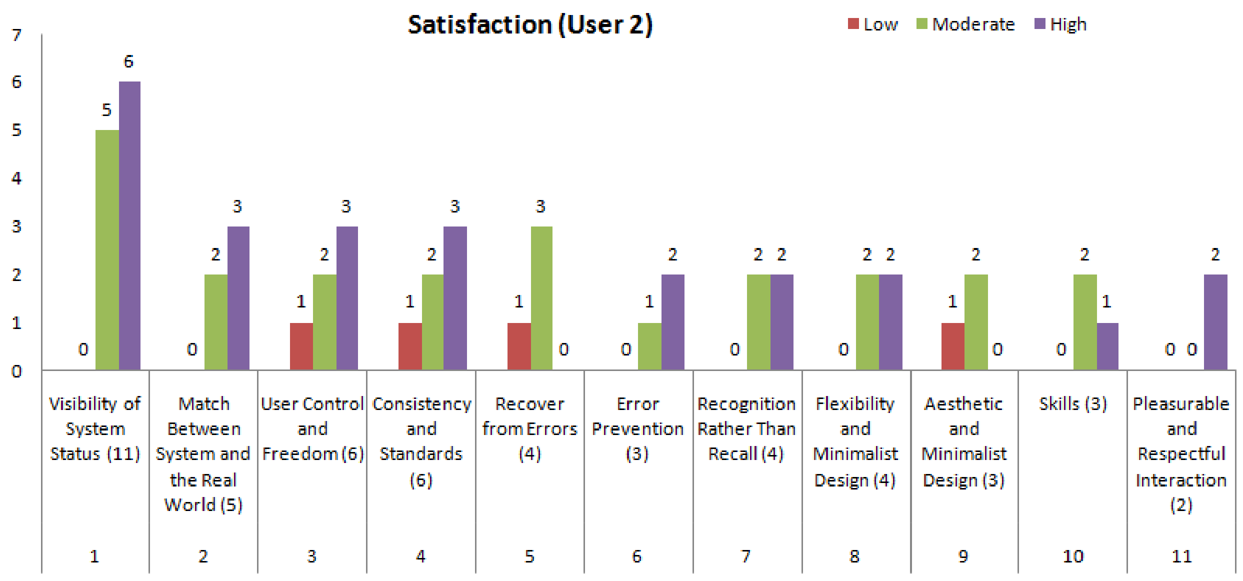


Figure 46 Features rated by User 1

Chapter 12

In this chapter we describe a model, Distributed Background Knowledge Infrastructure to enable semantic search. This Chepter organized as follows. Section 12.1 expands more on the Search problems. Section 12.3 describes the Semantic search approach, specifically *CSearch*. Section 12.4 described the limitation current approaches. Section 12.5 describes how syntactic search can be implemented on top of Structured P2P. Section 12.6 describes our solution *CANet* overlay followed by a example scenario. Section 12.8 we compare our work with other related work. In Section 12.9 we presented an evaluation of our current implementation and thus present the conclusion and the future work.

12. CANet: A Structured Distributed Background Knowledge Infrastructure for Supporting Semantic Search

Semantic search systems deal with meaning. To enable an effective semantic search, it is important to have sufficient metadata and background knowledge along with a suitable structure to utilize this for semantic reasoning. To achieve this goal we propose the *Concept Aware Network* (*CANet*), a structured peer-to-peer overlay network that uses classification-based representation of the resources to enable semantic resource discovery, which is built on available semantic attributes, on each peer. Documents stored by users in peers are classified in a manual, semi-automated or automated annotation of metadata. Classification consists of a set of nodes that hold the concepts that are interesting for the user. Each classification reflects the local view of user's interest. Related nodes among classifications are interconnected by means of semantic links that represent the semantic relationships between nodes, which can perform semantic reasoning on each other's contents. Initially *CANet* platform is build for supporting Concept Search by providing Distributed background knowledge and Inverted Index for semantic search over structured P2P network. However any search engines requires background knowledge for semantic reasoning can be build on the top of *CANet*. A prototype of *CANet* has developed.

12.1. Introduction

The revolution of Internet and the Web has taken the computer and information technology into a new age. The amount of information on the web is growing fast. The progress of information and communication technologies has given access to a large amount of information, which is increasingly becoming difficult to comprehend or manage. Thus, there is a strong need for a search system that can efficiently search, filter, interpret, and summarize the information according to the user needs.
Conventional search engines implement search for documents by using syntactic search, i.e., words or multi-word phrases are used as atomic elements in document and query representations. To perform a successful semantic search a semantic search engine must have the capability to understand the meaning of the web content. This can achieve by processing background knowledge and metadata that describe the content. One of the main obstacles providing web support is that, at present, the background knowledge of web content is not easily machine-accessible. Of course, there are various IR approaches to retrieve information. But their capabilities are still very limited to syntactic level without the semantics of user-provided information, are known to suffer in general from low precision while being good at recall. Semantic Web technologies offer us a new approach for managing information and processes, the fundamental principle of which is the creation and use of a semantic metadata that encompasses knowledge acquisitions and presentations.
Nowadays, the major search engines are based on a centralized architecture and are controlled by single authority despite their significant contribution to basic user needs. They attempt to create a single index for the whole Web. But the size, dynamics, and distributed nature of the Web make the search problem extremely hard, i.e., a very powerful server farm is required to have complete and up-to-date knowledge about the whole network to index it. Peer-to-Peer computing paradigm aim at decentralized organization principles of data and cooperative information management appeared as an alternative to centralized search engines for searching web content. In this objective they bring new degrees of freedom for changing information architectures and exchanging information between different peers in a network. Require-

ments for computational and storage resources, robustness, and scalability are major advantages of the P2P architecture over the centralized architecture. Each peer in the P2P network organizes only a small portion of the total information by creating its local index, while being able to access the information stored in the whole network. Thus peer-to-peer (P2P) computing paradigm appeared as an alternative to centralized search engines for searching web content.
In this paper we propose a semantic overlay network called *CANet* on top of distributed hash table (DHT) [104, 105, 106, 107] for supporting semantic reasoning by providing distributed background from divers sources distributed among all the peers in the network.

12.2. Search Problems

Information retrieval (IR) systems aim at mapping certain user information needs expressed by a natural language query *q*, to a finite or countably infinite set of documents *d* in the document collection *D* which can be of interest to user to satisfy user's specific and contingent information needs by ordering them according to their relevance. Therefore *IR* can be represented as a mapping function:

$$\mathrm{IR} : \mathrm{Q} \rightarrow \mathrm{D} \qquad (1)$$

Conventional search engines implement the mapping function in Equation 1 by exploiting syntactic search with some degree of filtering mechanism, serving well to meet basic user needs. A word or a multi-word phrase is used as an atomic element (term) in document and query representations. A syntactic matching of words (match) is used for matching document and query terms. Syntactic matching is implemented as search for equivalent words, words with common prefixes, or words within a certain edit distance with a given word with some degree of filtering mechanism. A big advantage of centralized search engines is that the number of messages needed in the query process is reduced to a minimum. It provides efficient bandwidth usage and quick response times. However, still there are noticeable problems associated with centralized syntactic search approaches:

12.2.1 Low Precision

Considering that most relevant pages are retrieved, they are of little use if another thousand or so mildly relevant or irrelevant documents were also retrieved. The reasons for low precision are:

- *Polysemy*: The same word may have multiple meanings and, therefore, query results, computed by a syntactic search engine, may contain documents where the query word is used in a meaning that is different from what the user had in mind. e.g., Check → Bank check or Verification?
- *Complex concepts*: Syntactic search engines fall short of taking into account complex concepts formed by natural language phrases and in discriminating among them. e.g., Computer table→ A laptop computer is on a coffee table.

12.2.2 Low Recall

Another less frequent problem with current search engines is low recall. Often it happens that we don't get any answer for our request, or that relevant pages are not retrieved. The reasons for low recall are:

- *Synonymy*: Two different words can express the same meaning in a given context and, therefore, query results, computed by a syntactic search engine, may miss documents where the meaning of a query word is expressed by a different word. e.g., Student and Pupil
- *Related concepts*: Syntactic search does not take into account concepts that are semantically related to the query concepts. e.g., Car → Volvo, FIAT, BMW

12.3. Semantic Search Approach

In order to address the above mentioned problems of syntactic search, *Concept Search* (*CSearch*) [21] extend syntactic search with semantics. *CSearch* reuses retrieval models (*Model*) and data structures (*Data Structure*) of syntactic search with the difference in that now words (*W*) are substituted with complex concepts (*C*) and syntactic matching of words (*WMatch*) is extended to semantic matching of concepts (*SMatch*) .The main idea is to keep the same machinery which has made syntactic search so successful, but to modify it so that, whenever possible, syntactic search is substituted with semantic search, thus improving the system performance. As a special case, when no semantic information is available, *CSearch* reduces to syntactic search, i.e.,the results produced by *CSearch* and syntactic search are the same. This idea is schematically represented in the equation below:

$$\boxed{\mathit{Syntactic\ Search} \xrightarrow{Term(W\rightarrow C),Match(WMatch\rightarrow SMatch)} \mathit{CSearch}}$$

Below we briefly describe how the words in *W* are converted into the complex concepts in *C* and also how the semantic matching *SMatch* is implemented in *CSearch*. We refer the interested reader to [21] for a complete account.

12.3.1 From natural language to formal language

To solve the problems related to the ambiguity of natural language, namely, the problems of polysemy and synonymy, we need to move from words, expressed in a natural language, to concepts (word senses), expressed in an unambiguous formal language.

12.3.2 From words to phrases

We assume that any piece of information can be encoded as an atomic concept and then, we can codify a set of atomic concepts to build complex concept. To solve the problem related to complex concepts, we need to analyze natural language phrases, which denote these concepts. We compute Complex concepts by analyzing meaning of the words and phrases and expressed in a propositional Description Logic (DL) language [31]. Single words are converted into atomic concepts uniquely identified as *lemma-sn*, where *lemma* is the lemma of the word, and *sn* is the sense number in *BK* (e.g., WordNet). For instance, the word *car* used in the sense of a *automobile*, which is the first sense in the *BK*, is converted into the atomic concept *car-1*. The conversion of words into atomic concepts is performed as follows. First, we look up and enumerate all meanings of the word in the *BK*. Next, we perform word sense filtering, i.e., we discard word senses which are not relevant in the given context (see [21, 108] for more details). Noun phrases are translated into the logical conjunction of atomic concepts corresponding to the words. Descriptive phrase, defined as a set of noun phrases connected by coordinating conjunction *OR*, are translated into logical disjunction of formulas corresponding to the noun phrases. In general case, complex concepts (*C*) can be represented as disjunctions (*t*) of conjunctions ($\sqcup$) of atomic concepts (*A*) without negation. Every document is represented as an enumerated sequence of conjunctive components $\sqcap A^d$possibly connected by "$\sqcup$".

$$C \equiv \sqcup \sqcap A^d \qquad (2)$$

12.3.3 From string similarity to semantic similarity

The problem with related concepts can be solved by incorporating knowledge about term relatedness (what we call the "*Background Knowledge" (BK)*). For instance, it can be statistical knowledge about word co-occurrence, lexical knowledge about synonyms and related words, or ontological knowledge about classes, individuals, and their relationships. *CSearch* allows search for documents describing complex concepts which are semantically related to complex concepts in the user query. Formally a query answer $QA(C^q,T)$, in *CSearch*, is defined as follows:

$$QA(C^q,T)=\{d \mid \exists C^d \in d, \text{s.t. } T \models C^d \sqsubseteq C^q\} \qquad (3)$$

where C^q is a complex query concept extracted from the query *q*, C^d is a complex document concept extracted from the document *d*, and *T* is a terminological knowledge base (i.e., the *BK*) which is used in order to check if C^d is more specific then C^q. A small fragment of *T* is represented in Figure 4. *T* can be thought of as an acyclic graph, where links represent subsumption axioms in the form $A_i \sqsubseteq A_j$, with A_i and A_j atomic concepts. Query answer $QA(C^q,T)$, defined in Equation 4, is computed by using a positional inverted index. A posting list P(t) is a list of all postings for term *t*: P(t) =[<*d, freq ,* [*position*]>] , where <*d, freq,* [*position*]> is a posting consisting of a document *d* associated with term *t*, the frequency freq of *t* in *d*, and a list [*position*] of positions of *t* in *d*. *CSearch* adopt a positional inverted index to index conjunctive components $\sqcap A^d$ by all more general or equivalent atomic concepts from *T* , where the inverted index dictionary consists of atomic concepts from *T*. The posting list P(A) for an atomic concept *A* stores the positions of conjunctive components $\sqcap A^d$, such that, $\sqcap A^d \sqsubseteq A$. Now the query answer $QA(C^q,T)$ can be computed just by merging posting lists. For instance, positions of conjunctive components $\sqcap A^d$ which are more specific than complex query concept $\sqcap A^q$, i.e., $\sqcap A^d \sqsubseteq \sqcap A^q$, can be computed by intersecting the posting lists for all the atomic concepts A^q in $\sqcap A^q$.

12.4. Limitations Of Current Approaches

The main limitation of *CSearch* is that it is a centralized system, i.e., the *BK* and the *inverted index* are stored in a single place. As any other centralized system, *CSearch* cannot scale without the need for powerful servers. It is also not realistic to assume that a single user can have a complete *BK* for all the possible domains; therefore, reasoning is always performed with respect to an incomplete *BK*. Almost every NLP application nowadays requires a certain level of semantic analysis of natural languages. Processing natural languages must require information about words and their meanings as background knowledge. Different Lexical knowledgebase's are served as useful resources to support this purpose. One of the widely used lexical knowledgebase is WordNet common in various applications including Information Retrieval (different types of applications can be found in [109]). However the diversity and the heterogeneity of natural language cannot be covered by single adaptation of WordNet even though they contain sufficiently wide range of words. Despite their adaptation for several languages and domains they don't cover special domain vocabulary and also that they are discrete and scattered across multiple physical locations. There is no uniform way to access them. See [110] for the problems which emerges when a part of the knowledge is missing.

It is important to express the documents in such that it can be used for semantic reasoning during search process. User generated classification hierarchies have always been used by humans as the most effective and intuitive way to organize their knowledge according to their subjective view of a domain of interest [1, 111]. Nodes in this classification specify concepts which are interesting for the user. Accordingly, the whole classification specifies the user interest profile. But what attributes the problem of unified human knowledge is the lack of support for uniform access to multiple Knowledge Bases located in different physical location. Highly dynamics and distributed nature of the Web make it unlikely that any centralized system can ever have complete and up-to-date knowledge about the whole network; therefore a distributed architecture is essential. Importantly, the scalability issue of distributed systems is our classic consideration in this paper.

12.5. Structured P2P

Distributed Hash Tables (DHTs) have been proposed as a way to enable an efficient discovery of objects in a very large structured P2P network [104, 105, 106, 107]. In DHT, every object is associated with a key, which is transformed into a hash using some hash function. The range of the output values of the hash function forms an ID space. Every peer in the network is responsible for storing a certain range of keys. Values, e.g., objects or information about objects, are stored at the precisely specified locations defined by the keys. The two main operations provided by DHT are:

- *put (key, value)* - stores the value on the peer responsible for the given key.
- *get (key)* → *value* - finds a peer responsible for the key and retrieve the value for the key.

A straightforward way to implement syntactic search is to use the DHT to distribute peers' inverted indices in the P2P network [112]. Peers locally compute posting lists *P(t)* for every term *t* and store them in the network by using the DHT 'put' operation. The *key* in this case is a term *t* while the value is a posting list *P(t)* associated with *t*. In DHT, each peer is responsible for a few terms and for every term *t* the peer merges all the posting lists *P(t)* for *t* from all the peers in the network. In order to find a set of documents which contain a term *t* we just need to contact the peer responsible for *t* and retrieve the corresponding posting list. The DHT 'get' operation does exactly this. In order to search for more than one term, we, first, need to retrieve posting lists for every single term, and then to intersect all these posting lists. The above approach has several problems (see e.g. [113, 114]). One problem is storage, for a large document collection, the number and the size of posting lists can be also large. Therefore, the storage needed to store the posting lists can potentially be bigger than the storage peers can allocate. Traffic a problem occurs as posting lists needs to be transferred when peers join or leave the network. Searching with multiple terms requires intersection of posting lists, which also need to be transferred. In the case of huge posting lists, bandwidth consumption can exceed the maximum allowed [109]. Popularity of terms can vary enormously. It can result in an extremely imbalanced load e.g., some peers will store and transfer much more data than others.

Several approaches were proposed in order to address the described above problems and to improve performance of information retrieval in structured P2P networks. Some of optimization techniques that can improve the performance of posting lists intersecting are summarized in [109]. Caching of results for queries with multiple terms is discussed in [105, 115]. In [115], only those queries are cached which are frequent enough and simple flooding is used for rare queries. In [113], only top terms are used for indexing of each document. Moreover, the term lists are stored on peers responsible for these top terms. Notice that by using only the top terms we can decrease the quality of search results. Automatic query expansion is proposed as a way to address this problem [113]. Some techniques to balance the load across the peers are also presented in [113]. Normally users are interested only in a few (*k*) high quality answers. An example of the approach for retrieving top k results, which does not require transmitting of entire posting lists, is discussed in [116]. In [117], indexing is performed by terms and term sets appearing in a limited number of documents. Different filtering techniques are used in [117] in order to make vocabulary to grow linearly with respect to the document collection size. In [116], it was proposed to index a peer containing a document and not the document itself. At search time, first, those peers are selected, which are indexed by all the terms in the query, then, the most promising peers are select, and finally, local search is performed on these peers. However these approaches are implementing syntactic search. Therefore, the problems of syntactic search, i.e., problems of polysemy, synonymy, complex concepts, and related concepts, can also affect the quality of the results produced by these approaches

12.6. CANet

To address the above problems we developed *CANet*, an overly network which exploit DHT as a structured P2P network for the implementation of concept overly, that deals with complex concept to support complex queries which goes beyond the *key* or just *keyword* search. *CANet* extend the centralized version of *CSearch* to the distributed level.

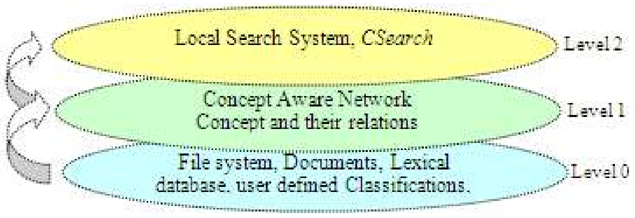

Local Search System, CSearch
Level 2
Concept Aware Network
Concept and their relations
Level 1
File system, Documents, Lexical database, user defined Classifications.
Level 0

Figure 47 CANet Layers

CANet organize the information in three layers by creating a peer-network-peer architecture (See Fig.38). The bottom layer, or peer resource layer, stores the files, pear meta-information, user-generated classifications, links and meta information for semantic reasoning. Middle layer, the *CANet overly*, summarizes and indexes the essential information from the bottom layer in order to facilitate efficient and quick search processes within *CANet*. We do this as follows, first, we extend the reasoning with respect to a single background knowledge *T* to the reasoning with respect to the background knowledge T_{P2P} which is distributed among all the peers in the network. Second, we extend the centralized inverted index (II) to distributed inverted index build on top of DHT. The idea is schematically represented in the equation below. *CSearch* works on the top layer as a local search system.

$$\boxed{CSearch \xrightarrow{Knowledge(T \rightarrow T_{P2P}), Index(II \rightarrow DHT)} P2P\ CSearch}$$

In the following, we show how the distributed background knowledge T_{P2P} are implemented in *CANet* using DHT to provide an efficient distributed semantic indexing and retrieval.

12.6.1 Distributed Background Knowledge

To access the background knowledge *T*, stored on a single peer, *CSearch* needs at least the following three methods:

getConcepts(W) returns a set of all the possible meanings (atomic concepts *A*) for word W. For example, *getConcepts(canine) → {canine-1 ('conical tooth'), canine-2 ('mammal with long muzzles')}.*

getChildren(A) returns a set of all the more specific atomic concepts which are directly connected to the given atomic concept *A* in *T* . For example, with respect to *T* in Figure 4, *getChildren (carnivore-1) → { canine-2, feline-1}.*

getParents(A) returns a set of all the more general atomic concepts which are directly connected to the given atomic concept *A* in *T* . For example, with respect to *T* in Figure 4, *getParents (dog-1) → {canine-2}.*

In order to provide access to background knowledge T_{P2P} distributed over all the peers in the *CANet* overlay, we create distributed background knowledge *DBK* using DHT. In *DBK*, each atomic concept *A* is identified by a unique concept ID (A_{ID}) which is composed from peer ID (P_{ID}), where peer is a creator of the atomic concept, and local concept ID in the Knowledge Base of the peer. Every atomic concept *A* is represented as a 3-tuple: *A* = <A_{ID},*POS*,*GLOSS*>, where A_{ID} is a concept *ID* ; *POS* is a part of speech; and *GLOSS* is a natural language description of *A*. In the rest of the paper, for the sake of presentation, instead of complete representation <A_{ID}, *POS*,*GLOSS*> we use just *lemma-sn*.

12.6.2 Indexing

Atomic concepts are indexed by words using the DHT 'put' operation, e.g., *put(canine,{canine-1, canine-2})*. Moreover, every atomic concept is also indexed by related atomic concepts together with the corresponding relations. We use a modification of the DHT 'put' operation *put(A, B, Rel)*, which stores atomic concept *B* with relation *Rel* on the peer responsible for (a hash of) atomic concept *A*, e.g., *put(canine-2, dog-1, 'v')*, *put(canine-2, carnivore-1, 'w')*.

After *DBK* has been created, *getConcepts(W)* can be implemented by using the DHT 'get' operation, i.e., *getConcepts(W) = get(W)*. Both methods *getChildren(A)* and *getParents(A)* are implemented by using a modified DHT 'get' operation *get(A, Rel)*, i.e., *getChildren(A) = get(W, 'v')* and *getParents(A) = get(W, 'w')*. The operation *get(A,Rel)* finds a peer responsible for atomic concept *A* and retrieve only those atomic concepts *B* which are in relation *Rel* with *A*.

Let us now see how *DBK* can be bootstrapped. At the beginning we have one single peer in the P2P network and *DBK* is equivalent to the background knowledge *T* of this peer. For example, *T* can be created from WordNet. A new peer joining the *CANet* bootstraps its own background knowledge from *DBK* by doing the following three steps. First, the peer computes a set of words which are used in the local document collection. Second, the peer download from *DBK* a set of all the atomic concepts which are associated with these words by using 'getConcepts' method. Finally, the peer downloads all the more general atomic concepts by recursively calling '*getParents*' method.

After bootstrapping, a user of each peer can extend *DBK* in the domain of her expertise according to her needs. The user can add a new atomic concept *A* to *DBK* by providing a set of words *W*, a part-of-speech *POS*, and a gloss *GLOSS*. By using the 'getConcepts' method, the peer retrieves from *DBK* all the atomic concepts *A* indexed by words in *W*. Then, glosses of retrieved concepts are compared with the *GLOSS* provided by user. We use gloss-based matchers from [31, 21, 118]. If no similar concepts are found, then *A* is created in the peer's local background knowledge and indexed in *DBK*. The user can also add a new meaning (i.e.,to assign an atomic concept *A*) to a word *W*. Similarly to how it was described above, before adding a new information, system checks if this information is not already in the system. Moreover, the user can define a new relation between atomic concepts in *DBK*. Before adding a new relation, system first checks if this relation does not introduce cycles in *DBK*. Cycles should be prevented because we need to have an acyclic *DBK*. Also system checks if the given relation cannot be decomposed into a sequence of existing relations. This step is done in order to minimize the amount of stored information.
Notice, that by extending peer's background knowledge *T* to *DBK* which stores T_{P2P} , we are likely to have a higher coverage on words, atomic concepts, and relations. Therefore, we can enable semantics to a higher extend in the semantic continuum, e.g., when user types a word which is not present in her *T* , she can use atomic concepts from background knowledge of other peers stored in *DBK*.

12.6.3 Query Answering

The query answer defined in Equation 3, can be extended to the case of distributed search by taking into account that the document collection D_{P2P} is equivalent to the union of all the documents from all the peers in the network (where each document d is uniquely identified by a document ID) and also that background knowledge T_{P2P} is distributed among all the peers.

$QA(C^q,T_{P2P})=\{d\in D_{P2P} | \exists C^d \in d, \text{ s.t. } T_{P2P} \models C^d \sqsubseteq C^q \}$ (4)

Let us consider a subset $QA(C^q,T_{P2P},A)$ of the query answer $QA(C^q,T_{P2P})$. $QA(C^q,T_{P2P} ,A)$ consists of documents *d* which contain at least one complex concept C^d which is more specific than the complex query concept C^q and contains atomic concept *A*.

$QA(C^q,T_{P2P},A)=\{d\in D_{P2P} | \exists C^d \in d, \text{ s.t. } T_{P2P} \models C^d \sqsubseteq C^q \text{ and } \exists A^d \in C^d, \text{ s.t. } A^d=A^d \}$
(5)

If by *C*(*A*) we denote a set of all atomic concepts A^d, which are equivalent to or more specific than concept *A*, i.e.,

$C(A)=\{A^d | T_{P2P} \models A^d \sqsubseteq A \}$ (6)

then, it can be shown that, given Equation 6, the query answer $QA(C^q,T_{P2P})$ can be computed as follows where A^* is an arbitrarily chosen atomic concept A^q in conjunctive component $\sqcap A^q$.

$QA(C^q,T_{P2P}) = \bigcup_{(\sqcap A^q)\in C^q} \bigcup_{A\in C(A^*)} QA(C^q,T_{P2P},A)$ (7)

Given Equation 7, the query answer can be computed by using a recursive algorithm described below. The algorithm takes as input complex query concept C^q and computes as output a query answer *QA* in five macro steps:

Step1 Initialize a query answer: *QA* = null

Step2 Select one atomic concept *A* from every conjunctive component $\sqcap A^q$ in complex query concept C^q. For every selected *A*, repeat steps 3, 4, and 5.

Step 3 Compute $QA(C^q,T_{P2P},A)$ and add the results to *QA*.

Step 4 Compute a set C_{ms} of all more specific atomic concepts *B* which are directly connected to the given atomic concept *A* in $T_{P2P.}$

Step 5 If $C_{ms} \neq$ null, then for every atomic concept in C_{ms}, repeat steps 3, 4, and 5.

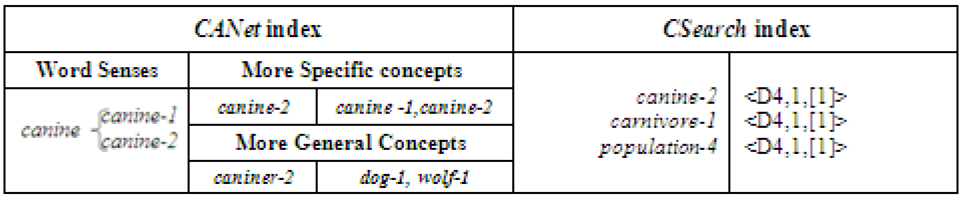
Note, that on step 2, atomic concepts *A* can be selected arbitrarily. In order to minimize the number of iterations, we chose *A* with the smallest number of more specific atomic concepts. The smaller the number, the fewer times we need to compute $QA(C_q, T_{P2P}, A)$ on step 3.

In the following, we, first, show how documents are indexed in P2P CSearch, and then we show how the described above algorithm can be implemented.

In *CANet*, complex concepts are computed in the same way as in *CSearch* (see Section 12.3). The only difference is that now if an atomic concept is not found in the local background knowledge *T*, then T_{P2P} is queried instead. *CANet* also uses the same document representation as *CSearch*. After document representations are computed, the indexing of documents is performed as follows. Every peer computes a set of atomic concepts *A* which appear in the representations of peer's documents. For every atomic concept *A*, the peer computes a set of documents *d* which contain *A*. For every pair <*A,d*>, the peer computes a set *S*(*d,A*) of all the complex document concepts C^d in *d*, which contain *A*.

$S(d,A) = \{C^d \in d \mid A \in C^d\}$ (8)

For example, if *d* is document D1 in Figure 3 and *A* is equivalent to *dog-1*, then $S(d,A) = \{small\text{-}4 \sqcap baby\text{-}3 \sqcap dog\text{-}1\}$. For every *A*, the peer sends document summaries corresponding to *A*, i.e., pairs <*d* , *S*(*d,A*) >, to a peer P_A responsible for *A* in *DBK*. The peer P_A indexes these summaries using the local *CSearch*. In total, every peer in the network is responsible for some words and for some atomic concepts. Peers maintain the following information for their words and concepts: (1) For every word, the peer stores a set of atomic concepts (word senses) for this word, (2) For every atomic concept, the peers stores a set of direct more specific and more general atomic concepts , (3) Document summaries <*d* , *S*(*d,A*)> for all the atomic concepts *A* (for which the peer is responsible) are stored on the peer and indexed in the local *CSearch*, i.e., the summaries are indexed in the positional inverted index.



CANet index			*CSearch* index	
Word Senses	More Specific concepts			
canine {*canine-1*, *canine-2*}	*canine-2*	*canine -1,canine-2*	*canine-2* *carnivore-1* *population-4*	<D4,1,[1]> <D4,1,[1]> <D4,1,[1]>
	More General Concepts			
	caniner-2	*dog-1, wolf-1*		

Figure 48 An example of the peers index information stored on the peer responsible for a single word canine and for a single atomic concept canine-2

12.7. Example Scenario

In this section we have described the different steps of the algorithm for computing the query answer in *CANet*:

Step 1 A peer P_I initiates the query process for complex query concept C^q and initialize the query answer *QA*.

Step 2 For every $\sqcap A^q$ in C^q, P_I selects *A* in $\sqcap A^q$ with the smallest number of more specific atomic concepts. For every selected *A*, C^q is propagated to the peer P_A responsible for *A*.

Step 3 P_A receives the query concept C^q and locally (by using *CSearch*) computes the set $QA(C^q, T_{P2P}, A)$. The results are sent directly to P_I . On receiving new results $QA(C^q, T_{P2P}, A)$, P_I merges them with *QA*. An (intermediate) result is shown to the user.

Step 4 P_A computes the set C_{ms} by querying locally stored more specific concepts (e.g., see 'More specific concepts' in Figure 2).

Step 5 P_A propagates C^q to all the peers P_B responsible for atomic concepts *B* in C_{ms}, i.e., Step 2 is repeated on every P_B.

Note, that, in order to optimize query propagation, peer P_A can pre-compute addresses of peers P_B which are responsible for more specific atomic concepts, and use DHT to locate such peers only when precomputed information is outdated.

An example of how the query answer $QA(C^q, T_{P2P}; A)$ is computed is given in Figure 40. Peers, represented as small circles, are organized in a DHT overlay, represented as a ring. *A* query consisting of a single query concept C^q *=little-4* ⊓ *canine-2* is submitted to peer P_1 . Let us assume that atomic concept *canine-2* has smaller number of more specific atomic concepts then concept *little-4*. In this case, C^q is propagated to a peer $P_{canine\text{-}2}$, i.e., the peer responsible for atomic concept *canine-2*. The query propagation is shown as a firm line in Figure 3. $P_{canine\text{-}2}$ searches in a local *CSearch* index with C^q. No results are found. $P_{canine\text{-}2}$ collects all the atomic concepts which are more specific then *canine-2*, i.e., atomic concepts *dog-1* and *wolf-1*. Query concept C^q is propagated to peers $P_{dog\text{-}1}$ and $P_{wolf\text{-}1}$. $P_{dog\text{-}1}$ finds no results while $P_{dog\text{-}1}$ finds document D_1. D_1 is an answer because it contains concept *small-4* ⊓ *baby-3* ⊓ *dog-1* which is more specific than *little-4* ⊓ *canine-2*. D_1 is sent to P_1 , which presents it to the user. The results propagation is shown as a dash line in Figure 3. Both peers $P_{\text{dog-1}}$ and $P_{\text{wolf-1}}$ have no more specific concepts than *dog-1* and *wolf-1*, therefore they do not propagate C^q to any other peers.

Note that the further we go in propagating query, the less precise is the answer. For instance, the user searching for *canine-2* might be more interested in documents about concept *canine-2* than in documents about concept *dog-1*, and she can be not interested at all in documents about very specific types of dogs. In *CANet*, we allow user to specify the max allowed distance in numbers of links between atomic concepts in T_{P2P} . Notice that this distance is similar to a standard time-to-live (TTL) [112]. In order to compute the query answer for a more complex query, the intersection of posting lists needs to be computed. Since our approach is not replacing syntactic search but extending it with semantics, for an efficient implementation of the intersection, we can reuse the optimization techniques developed in P2P syntactic search (see e.g. Section 5).

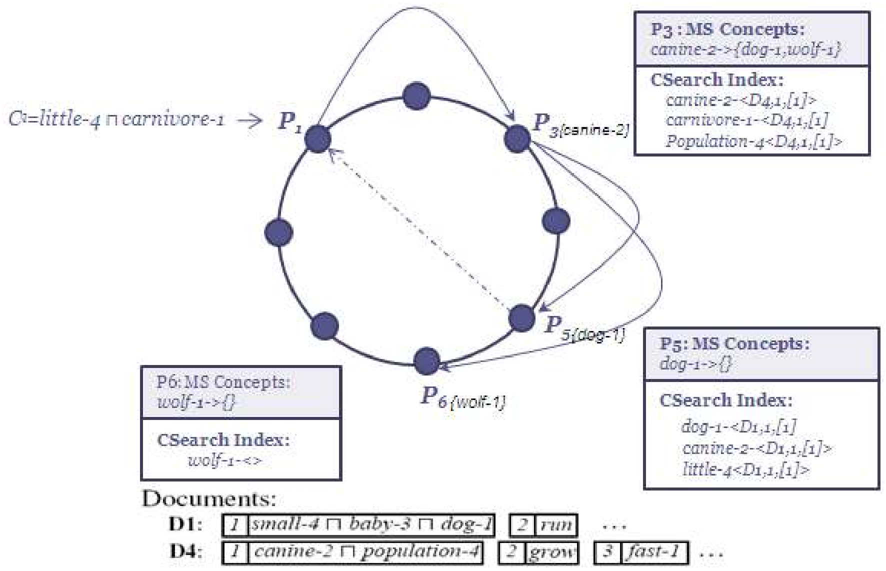


Figure 49 Query answering

Other syntactic techniques, e.g., for ranking and merging of query results, can also be reused in *CANet*. For this, words, in these techniques, need to be replaced by concepts and syntactic matching needs to be replaced by semantic matching. Notice that in some P2P search approaches, instead of a single document, a group of documents, a peer, or a group of peers are indexed and searched. Our approach can be adapted to these problems: a group of documents, peers, or a group of peers should be annotated by complex concepts and then they can be indexed in the same way as a single document.

12.8. Related Work

A number of P2P search approaches have been proposed in the literature (for an overview see [112]). Examples of how full text retrieval can be efficiently implemented on top of structured P2P networks are described in [114, 119, 113, 122, 115, 116, 117,]. All of these approaches are based on syntactic matching of words and, therefore, the quality of results produced by these approaches can be negatively affected by the problems related to the ambiguity of natural language. *CANet* is based on semantic matching of concepts which allows it to deal with ambiguity of natural language. Note that, since our approach extends syntactic search and does not replace it, the optimization techniques which are used in P2P syntactic search can be easily adapted to *CANet*. Some P2P search approaches use matching techniques which are based on the knowledge about term relatedness. For instance, statistical knowledge about term co-occurrence is used in [120]. Knowledge about synonyms and related terms is used in [121]. Differently from these approaches, *CANet* is based on semantic matching of complex concepts and knowledge about concept relatedness is distributed among all the peers in the network

12.9. Performance evaluation

The possibility of OS resource allocation noise (e.g., thread scheduling) while performing the evaluation has been reduced by mean deviation over 10 samples on similar settings.We increased the number of peers with a factor of 10 and the highest number of peers taken is 60, and they ran parallel on a single machine. 340,000 concepts were randomly equally distributed among the peers without duplication. Five levels of concept depth were considered where the last level (Level Max) recursively looks for the last of concept hierarchy. The first level computes the first child of given query and other consecutive levels accounts for up to forth descendants. The sample query string is composed of 19 words with 55 senses in total. In order to obtain imperative data we conducted the evaluation for second level, third, fourth and max level. The concepts obtained from the CVs of different peers at each level are 2707, 8203, 17307 and 30034 respectively.

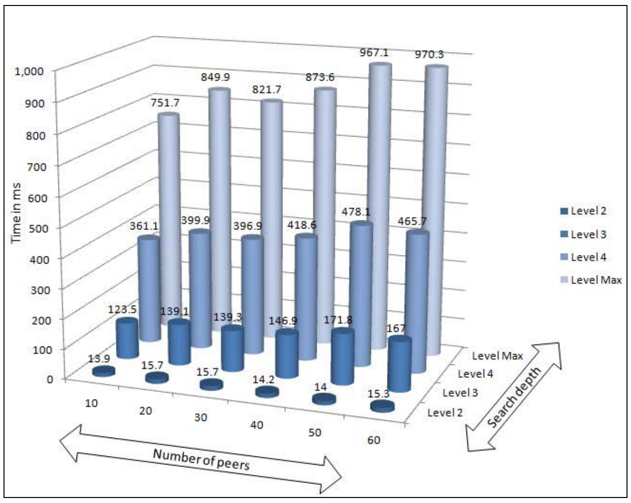


Figure 50 Query propagation time over number of peers and concept depth.

The above graph shows along the Y axis that the numbers of peers exerts very little impact on the performance while other settings remain same. It could be due to the small number of peers taken for evaluation or the sampling was too close. However, the distinguishable difference could be observed with different concept level.

The following graph shows a decrease in performance at different concept depth which takes a linear computational time. The supposition we made here that a concept depth cannot go in infinite in any case.

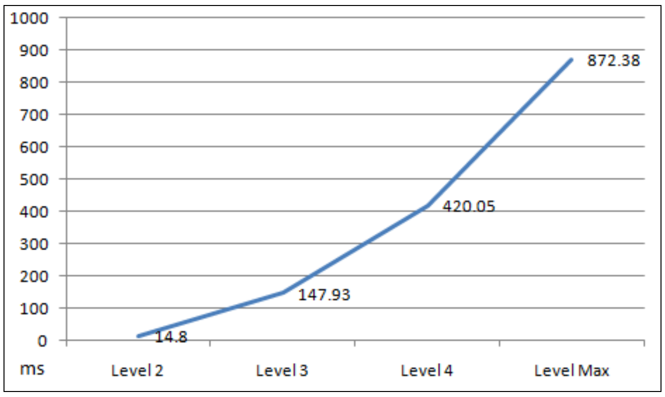


Figure 51 Time increases with the increase of concept depth.

12.10. Conclusion

In this chapter, we have presented an approach, called *CANet*, which allows for a semantic search on top of distributed hash table (DHT). There are two main aspects in which *CANet* extends *CSearch*: (i) centralized document index is replaced by distributed index build on top of DHT; (ii) reasoning with respect to single background knowledge is extended to the reasoning with respect to the background knowledge distributed among all the peers in the network. *CANet* addresses the scalability problem of *CSerarch* and the ambiguity problem of natural language in P2P syntactic search. Our evaluation shows that the solution is highly scalable both in terms of number of peers and distributed background knowledge. While the differentiated value between level 3 and 4 is 0.352, for same settings, this value between level 4 and max depth (could have been more than 5) is 0.481.

Chapter 13

13. Conclusion and future work

In this thesis we have presented the three-level architecture (*The universal knowledge , the background knowledge and the user knowledge*) of a high quality knowledge base for solving the semantic heterogeneity problem, namely the problem of diversity in knowledge, and therefore support interoperability. We have shown that, the only meaningful way to approach the semantic heterogeneity problem is to allow and support knowledge diversity in language, culture, purpose and belief. This can be achieved by helping people to organize and make accessible their knowledge and create powerful tools to support interoperability. The more people will be able or will want to share the more they will interoperate. As a part of the solution we have presented the details of the different parts of the *universal knowledgebase* , namely *linguistic and concept knowledge*, *domain knowledge* and *ent*ity(this thesis covered entity type of the entity) , which will be able to manage knowledge allowing users to personalize and evolve it

We have presented the three-level architecture whereas in this we focused on the construction and maintenance of the ***universal knowledge*** part. From *Entities* part we covered the *e-types* which provide constraints about the metadata (i.e., kind of attributes) associated to the entities of that kind. But we didn't need to build a system for managing entity also. Moreover this work didn't include the local ***background knowledge*** part. Though the *UK* and the *BK* have the same structure , construction of the ***universal knowledge*** and local ***background knowledge*** should provide the right amount of flexibility necessary to take into account all dynamics connected to the knowledge management (creation, indexing, browsing and searching, evolution and transfer) taking into account different viewpoints. They should represent the fundamental means to interpret classifications built locally by users, by translating them into their formal representation (namely into lightweight ontologies), indexed objects and their annotations and to enable the execution of some basic semantic services on top of them, such as semantic matching, search and navigation. It also includes the development of the necessary algorithms and tools for supporting these activities. This in turn will introduce the research challenge to solve.

We also have presented an p2p indexing approach, called CANet, which allows for a semantic search on top of distributed hash table (DHT). Future work includes: (i) the development of techniques which can control the quality of a user input and in general to control the quality of *DBK*; (ii) the development of document relevance metrics based on both syntactic and semantic similarity of query and document descriptions; (iii) evaluating the efficiency of the proposed solution.

Bibliography

1. F. Giunchiglia and I. Zaihrayeu. Lightweight ontologies. In The Encyclopedia of Database Systems. Springer, 2008.
2. T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. Scientific American, (284(5)):34–43, May 2001.
3. F. Giunchiglia, M. Marchese, and I. Zaihrayeu. Towards a theory of formal classification. *In Proceedings of the AAAI-05 Workshop on Contexts and Ontologies: Theory, Practice and Applications* (C&O-2005), Pittsburgh, Pennsylvania, USA, 2005.
4. Genesereth. , Nilsson,, Foundation of Artificial Intelligence, 1987.
5. Neches, R., Fikes, R. E., Finin, T., Gruber, T. R., Patil, R., Senator, T., & Swartout, W. R. Enabling technology for knowledge sharing. AI Magazine, 12(3):16-36, 1991.
6. Hayes, P. J. The Second Naive Physics Manifesto, in Hobbs and Moore (eds.), *Formal Theories of the Common-Sense World*, Norwood: Ablex, 1985.
7. Zhu, H. and Madnick, S. (2006a) 'A Lightweight Ontology Approach to Scalable Interoperability', VLDB Workshop on Ontologies-based techniques or DataBases and Information Systems (ODBIS'06), September 11, 2006, Seoul, Korea.
8. S. Auer, J. Lehmann, and S. Hellmann. LinkedGeoData - adding a spatial dimension to the web of data. *In Proc. of 7th International Semantic Web Conference (ISWC),* pages 731–746, 2009.
9. G. M. Sacco, 2000. Dynamic Taxonomies: A Model for Large Information Bases. IEEE Transactions on Knowledge and Data Engineering, Vol. 12 (3) pp. 468–479.
10. D. Soergel, 1972. A Universal Source Thesaurus as a Classification Generator. Journal of the American Society for Information Science 23(5), pp. 299–305.
11. S. R. Ranganathan, 1965. The Colon Classification. In S. Artandi, editor, Vol IV of the Rutgers Series on Systems for the Intellectual Organization of Information. New Brunswick, NJ: Graduate School of Library Science, Rutgers University.
12. J. McCarthy, 1987. Generality in Artificial Intelligence. Communications of the ACM. Vol. 30, No. 12, pp. 1030-1035. Also in ACM Turing Award Lectures, The First Twenty Years, ACM Press, 1987.
13. G. M. Sacco, 2000. Dynamic Taxonomies: A model for Large Information Bases A demo can be founded at http://www.dbworldx.di.unito.it
14. M. J. Bates, 1989. The design of browsing and Berrypicking techniques for the online search interface.
15. OSAF web site (http://wiki.osafoundation.org/Journal/)(accessed January 2011)
16. D. Soergel, 1999. Design of an integrated information structure interface.
17. Y. Kalfoglou and M. Schorlemmer. Ontology Mapping: The State of the Art. The Knowledge Engineering Review, 18(1):1{31, 2003.
18. N.F. Noy. Semantic Integration: a Survey Of Ontology-Based Approaches. SIGMOD Record, 33(4):65{70, December 2004.
19. F. Giunchiglia, V. Maltese, A. Autayeu, 2008. Computing minimal mappings. University of Trento, DISI Technical Report: http://eprints.biblio.unitn.it/archive/00001525/
20. H. Wache, T. Voegele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Huebner. Ontology-Based Integration of InformationA Survey of Existing Approaches. *In Proc. of the WS on Ontologies and Information Sharing,* 2001.
21. Fausto Giunchiglia, Uladzimir Kharkevich, Ilya Zaihrayeu. Concept Search. *In Proceedings of ESWC'2009. pp.429~444*
22. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel Schneider. The Description Logic Handbook : Theory, Implementation and Applications. Cambridge University Press, 2003.
23. T.H. Nelson, 1988. Managing immense storage. BYTE Vol. 13 (1) pp. 225-238.
24. B. Hjorland, 2008. What is Knowledge Organization? In Knowledge Organization: International Journal devoted to Concept Theory, Classification, Indexing and Knowledge Representation 35(2/3) pp. 86-101.

25. V. Broughton et al, 2005. Knowledge Organization. European Curriculum Reflections on Library and Information Science Education, chapter 7, pp. 133-148.
26. B. Vickery, 2008. On ‘knowledge organisation’. Appeared on Brian Vickery web site: http://www.lucis.me.uk/.
27. D. Soergel, B. Lauser; A. Liang, F. Fisseha, J. Keizer, S. Katz, 2004. Reengineering thesauri for new applications. The AGROVOC example. Journal of Digital Information, vol. 4 issue 4, article No. 257.
28. N. Grabar, P. Zweigenbaum, 2000. Automatic acquisition of domain-specific morphological resources from thesauri. *In Proceedings of RIAO.*
29. http://www.internetworldstats.com/stats.htm
30. http://www.techpluto.com/web-20-services
31. F. Giunchiglia, M. Marchese, I. Zaihrayeu, 2006. Encoding Classifications into Lightweight Ontologies. Journal of Data Semantics 8, pp. 57-81.
32. Protege-2000, The Protege-2000 website,Stanford Medical Informatics,http://protege.stanford.edu
33. Sintek, M. Ontoviz tab: Visualizing Prot´eg´e ontologies, 2003 http://protege.stanford.edu/plugins/ontoviz/ ontoviz.html.
34. Katifori, A., Torou, E., Halatsis, C., Lepouras, G. & Vassilakis, C.. A Comparative Study of Four Ontology Visualization Techniques in Protégé: Experiment Setup and Preliminary Results, *Proceedings of the IV 06 Conference*. 2006
35. M. A. Storey, R. Lintern, N. Ernst and D. Perrin. Visualization and Protégé. *In 7th International Protégé Conference.* 2004.
36. Kalyanpur, A., Parsia, B., Sirin, E., et al., Debugging Unsatisfiable Classes in OWL Ontologies. *Journal of Web Semantics,* 2005. 3(4)
37. Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer and D. Wenke, OntoEdit: collaborative ontology engineering for the semantic web. In: First International Semantic Web Conference (ISWC’02)Lecture Notes in Computer Science vol. 2342, Springer, Berlin (2002), pp. 221–235.
38. S. Bechhofer, I. Horrocks, C. Goble and R. Stevens, OilEd: a reason-able ontology editor for the Semantic Web. In: Joint German/Austrian conference on Artificial Intelligence (KI’01)Lecture Notes in Artificial Intelligence vol. 2174, Springer, Berlin (2001), pp. 396–408.
39. S. Youn, A. Arora, P. Chandrasekhar, P. Jayanty, A. Mestry, S. Sethi, - Survey about Ontology Development Tools for Ontology-based Knowledge Management, http://www-scf.usc.edu/~csci586/projects/ontology-survey.doc, accessed on January 15, 2011
40. I. Horrocks, U. Sattler and S. Tobies, Practical reasoning for expressive description logics. *In: 6th International Conference on Logic for Programming and Automated Reasoning* (LPAR’99)Lecture Notes in Artificial Intelligence, Springer, Berlin (1999), pp. 161–180
41. J. Domingue, Tadzebao and Webonto: Discussing, Browsing and Editing Ontologies on the Web, *in: Proc. 11th Knowledge Acquisition Workshop (KAW98)*, Banff, 1998.
42. Julio C. Arpírez, Oscar Corcho, Mariano Fernández-López, and Asunción Gómez-Pérez. WebODE: a scalable workbench for ontological engineering. *In Proceedings of the 1st international conference on Knowledge capture (K-CAP '01). ACM*, New York, NY, USA, 6-13, 2001.
43. A. Farquhar, R. Fikes, J. Rice, The Ontolingua Server: A Tool for Collaborative Ontology Construction, *in: Proc. 10th Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW96)* Banff, 1996, pp. 44.1–44.19
44. Swartout, B., Patil, R., Knight, K. and Russ, T. “Ontosaurus: a tool for browsing and editing ontologies,” Gaines, B.R. and Musen, M.A. (eds.), *Proceedings of Tenth Knowledge Acquisition Workshop,* 1996.
45. Top Quadrant Inc. 2007. Topbraid Composer. http://www.topbraidcomposer.com
46. George A. Miller. WordNet: A Lexical Database for English. *Communications of the ACM* Vol. 38, No. 11: 39-41, 1995.
47. P Pesnik. 1995. Disambiguating noun grouping with respect to wordnet senses. *In Proceedings of 3rd Workshop on Very Large Corpora.*

48. Palmer, M., Dang, H. T., Fellbaum, C.: Making Fine-grained and Coarse-grained Sense Distinctions, Both Manually and Automatically. *Journal of Natural Language Engineering*. 13, 137–163 ,2006.
49. Mihalcea, R., Moldovan, D.I.: EZ Wordnet: principles for automatic generation of a coarse grained WordNet. *In: Proc. of FLAIRS,* 2001
50. F. Segond, A. Schiller, G. Grefenstette, and J. Chanod. 97. An experiment in semantic tagging using hidden markov model tagging. *In Proceedings of the Workshop in Automatic Information Extraction and Building of Lexical Semantic Resources*, pages 78-81.
51. J. Morato, M. Marzal, J. Lloréns and J. Moreiro. WordNet Applications. *In: Proc. of the 2nd Int. Conf. Global WordNet*, Brno, Czech Rep. 2004
52. A. Bosca, D. Bomino, and P. Pellegrino. Ontosphere: more than a 3d ontology visualization tool. *In SWAP, the 2nd Italian Semantic Web Workshop*.
53. C. Plaisant, J. Grosjean and B.B. Bederson, “SpaceTree: Supporting Exploration in Large Node Link Tree, Design Evolution and Empirical Evaluation,” *In Proceedings of IEEE Symposium on Information Visualization*, pp. 57-64, 2002.
54. A. Cockbrun and B. McKenzie, Evaluating the Effectiveness of Spatial Memory in 2-D and 3-D Physical and Virtual Environments. *In Proc. ACM Conf. Human Factors in Computing Systems (CHI 2002)*, pp. 203-210, 2002.
55. Herman, I., Melanc¸On, G., Aand Marshall M. S. Graph visualization and navigation in information visualization: A survey. *IEEE Trans. Visual. Comput/ Graph*. Vol. 6, No. 1, January–March. 24–43. 2000.
56. B. Shneiderman, "The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations." Proc. 1996 IEEE Visual Languages, Sept. 1996, pp. 336-343.
57. Michael Denny. Ontology Building: A Survey of Editing Tools http://www.xml.com/pub/a/2002/11/06/ontologies.html
58. C. Fellbaum, WordNet: An Electronic Lexical Database. MIT Press, 1998.
59. D. Lenat, 1995. CYC, a large-scale investment in knowledge infrastructure. Communications of the ACM, 38, no. 11 (November).
60. John Sinclair, Tuscan Word Centre. Developing Linguistic Corpora: a Guide to Good Practice Appendix: How to build a corpus http://www.ahds.ac.uk/guides/linguistic-corpora/appendix.htm
61. Pinto, 1999. Some Issues on Ontology Integration. *Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods (KRR5)* 7.1–7.12.
62. Klein, Combining and relating ontologies: An Analysis of Problems and Solutions. In IJCAI-2001 Workshop on Ontologies and Information Sharing, pp. 53–62. 2002
63. P. Shvaiko, J. Euzenat, 2007. Ontology Matching. Springer-Verlag New York, Inc. Secaucus, NJ, USA.
64. http://www.techpluto.com/web-20-services (accessed January 2011)
65. Giuliano, C., Lavelli, A., Pighin, D., Romano, L.: FBK-IRST: Kernel methods for semantic relation extraction. *In: Proceedings of the 4th International Workshop on Semantic Evaluations. pp. 141–144. Association for Computational Linguistics* (2007)
66. Culotta, A., Sorensen, J.: Dependency tree kernels for relation extraction. *In: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics.* pp. 423–431. Association for Computational Linguistics (2004)
67. IEEE P1600.1. Standard Upper Ontology Working Group (SUO WG). http://suo.ieee.org/
68. Ackoff, R. L., *"From Data to Wisdom"*, Journal of Applied Systems Analysis, Volume 16, 1989 p 3-9.
69. M. Hearst and others, 2001-2006 – Flamenco project: http://flamenco.berkeley.edu
70. M. Hearst, 2006. Clustering versus Faceted Categories for information exploration.
71. OSAF web site (http://wiki.osafoundation.org/Journal/) (accessed January 2011)
72. F. Giunchiglia, I. Zaihrayeu, V. Maltese. Faceted classifications and the Get-Specific classification principle, KnowDive internal report.

73. Nutter, J.T. 1998. Epistemology. In: S. Shapiro, Editor, Encyclopedia of artificial intelligence, John Wiley.
74. Ranganathan, S. R. (1967). Prolegomena to library classification. New York: Asia Publishing.
75. G.Bhattacharyya, 1975. POPSI: Its fundamentals and procedure based on a general theory of subject indexing languages. Library Science with a slant to Documentation. 16 (1): 1-34.
76. Stockdale, C. and Possin , C. Spatial Relations and Learning. http://www.newhorizons.org/spneeds/inclusion/teaching/stockdale.html
77. Nicola Guarino, Massimiliano Carrara, and Pierdaniele Giaretta. An ontology of meta-level categories. In Jon Doyle, Erik Sandewall, and Pietro Torasso, editors, KR'94: Principles of Knowledge Representation and Reasoning, pages 270-280. Morgan Kaufmann, San Francisco, California, 1994.
78. Nicola Guarino and Christopher A. Welty. Identity, unity, and individuality: Towards a formal toolkit for ontological analysis. *In ECAI,* pages 219-223, 2000.
79. Nicola Guarino and Christopher A. Welty. An overview of ontoclean. In S. Staab and R. Studer, editors, Handbook on Ontologies, pages 151-159. Springer Verlag, 2004.
80. Christopher A. Welty and Nicola Guarino. Supporting ontological analysis of taxonomic relationships. Data Knowledge Engineering, 39(1):51-74, 2001.
81. Robert Spence. Information Visualisation. Addison-Wesley, Reading, MA, 2001.
82. Colin Ware. Information Visualization: Perception for Design. Morgan Kaufmann, San Francisco, CA, 2nd edition, 2004.
83. K. Stuart Card, Jock D. Mackinlay, and Ben Shneiderman. Readings *in Information Visualization: Using Vision to Think. Morgan Kaufmann, San Francisco,* 1999
84. M. Tory and T. M¨uller. Human factors in visualization research. *IEEE Transactions on Visualizationand Computer Graphics, 10(1),* 2004.
85. Riccardo Mazza, Introduction to Information Visualization
86. Robert Jacobson, editor. Information Design. MIT Press, Cambridge, MA, 1999.
87. Jonathan Hey, The Data, Information, Knowledge, *Wisdom Chain: The Metaphorical link*, published at Intergovernmental Oceanographic Commission - OceanTeacher: a training system for ocean data and information management, 2004.
88. T. D. Wilson, *The nonsense of 'knowledge management'*, Information Research, Vol. 8 No. 1, Paper Number 144, October 2002 [online: http://informationr.net/ir/8-1/paper144.html].
89. Clark D. http://www.nwlink.com/~donclark/performance/understanding.html accessed December 2004 (accessed February 2011)
90. K. Stuart Card, Jock D. Mackinlay, and Ben Shneiderman. Readings in Information Visualization: Using Vision to Think. Morgan Kaufmann, San Francisco, 1999.
91. Colin Ware. Information Visualization: Perception for Design. Morgan Kaufmann, San Francisco, CA, 2nd edition, 2004.
92. Alan Hashimoto; Mike Clayton. Visual Design Fundamentals: A Digital Approach, Third Edition
93. Kurt Koffka. Principles of Gestalt Psychology. Harcourt, New York, 1935
94. Wolfgang Kohler. Gestalt Psychology. Liveright, New York, 1929
95. Max Wertheimer. Untersuchungen zur lehre von der gestalt ii. Psychologische Forschung,1923
96. Anderson, J. R. (2000). Learning and memory: An integrated approach. New York: John Wiley & Sons.
97. Anne Treisman. Preattentive processing in vision. Computer Vision, Graphics, and Image Processing, 31(2):156–177, August 1985.
98. George A. Miller. The magical number seven, plus or minus two. The Psychological Review, 63(2):81–97, 1956
99. Cooper, G. (1998). Research into Cognitive load Theory and Instructional Design at UNSW. Dr. Graham Cooper, School of Education Studies, The University of New South Wales. http://education.arts.unsw.edu.au/CLT_NET_Aug_97.HTML
100. http://www.wisegeek.com/what-is-raw-data.htm (accessed February 2011)

101. Dillon, A. (2001) Usability evaluation. In W. Karwowski (ed.) Encyclopedia of Human Factors and Ergonomics, London: Taylor and Francis.
102. Lewis, J (1994) Sample sizes for usability studies: additional considerations. Human Factors, 36(2) 368-378.
103. Nigel Bevan (June 2007) Designing a User-Based Evaluation. http://www.nigelbevan.com/papers/Designing Based Evaluation.pdf (accessed February 2011)
104. Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications,* 2001.
105. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for nternet applications. IEEE/ACM Transactions on Networking, 11(1):17{32, February 2003. I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
106. Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *IFIP/ACM Middleware,* 2001.
107. Ben Y. Zhao, John D. Kubiatowicz, and Anthony D.Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, UC Berkeley, January 2001.
108. Zaihrayeu, L. Sun, F. Giunchiglia, W. Pan, Q. Ju, M. Chi, and X. Huang. From web directories to ontologies: Natural language processing challenges. *In 6th International Semantic Web Conference (ISWC 2007)*. Springer, 2007.
109. Petr Sojka, Karel Pala, Pavel Smrz, Christiane Fellbaum, Piek Vossen (Eds.): *GWC 2004, Proceedings*, pp. 270-278. Masaryk University, Brno, 2003
110. Fausto Giunchiglia, Pavel Shvaiko, and Mikalai Yatskevich. Discovering missing background knowledge in ontology matching. *In Proc. of ECAI*, 2006.
111. Fausto Giunchiglia, Maurizio Marchese, and Ilya Zaihrayeu. Encoding classications into lightweight ontologies. In Journal on Data Semantics (JoDS) VIII,Winter 2006.
112. John Risson and Tim Moors. Survey of research towards robust peer-to-peer networks: Search methods. Computer Networks, 50:3485-3521, 2006.
113. Chunqiang Tang and Sandhya Dwarkadas. Hybrid global-local indexing for efficient peer-to-peer information retrieval. *In NSDI'04: Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation,* 2004.
114. Jinyang Li, Boon Thau, Loo Joseph, M. Hellerstein, and M. Frans Kaashoek. On the feasibility of peer-to-peer web indexing and search. *In 2nd International Workshop on Peer-to-Peer Systems (IPTPS 2003),* 2003.
115. Gleb Skobeltsyn and Karl Aberer. Distributed cache table: efficient query-driven processing of multi-term queries in p2p networks. *In P2PIR '06: Proceedings of the international workshop on Information retrieval in peer-to-peer networks,* 2006.
116. Matthias Bender, Sebastian Michel, Peter Triantaffllou, Gerhard Weikum, and Christian Zimmer. P2P content search: Give the web back to the people. *In 5th International Workshop on Peer-to-Peer Systems (IPTPS 2006),* 2006.
117. Toan Luu, Gleb Skobeltsyn, Fabius Klemm, Maroje Puh, Ivana Podnar ·Zarko, Martin Rajman, and Karl Aberer. AlvisP2P: scalable peer-to-peer text retrieval in a structured p2p network. *In Proc. VLDB Endow.,* 2008.
118. Mark Sanderson. Retrieving with good sense. Inf. Retr., 2(1):49-69, 2000.
119. Bobby Bhattacharjee, Sudarshan Chawathe, Vijay Gopalakrishnan, Pete Keleher, and Bujor Silaghi. Efficient peer-to-peer searches using result-caching. *In Proc. of the 2nd Int. Workshop on Peer-to-Peer Systems,* 2003.

120. Chunqiang Tang, Zhichen Xu, and Sandhya Dwarkadas. Peer-to-peer information retrieval using self-organizing semantic overlay networks. SIGCOMM '03: Proceedings of the 2003 conference on Applications,technologies, architectures, and protocols for computer communications, 2003.
121. Wenhui Ma, Wenbin Fang, Gang Wang, and Jing Liu. Concept index for document retrieval with peer-to-peer network. *In Proc. SNPD '07,* 2007
122. Jiangong Zhang and Torsten Suel. Efficient query evaluation on large textual collections in a peer-to-peer environment. *In P2P '05: Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing,* 2005.
123. F. Giunchiglia, B. Dutta, and V. Maltese. Faceted d lightweight ontologies. DISI Technical Report DISI-09-022, University of Trento. To appear in: "Conceptual Modeling: Foundations and Applications", A. Borgida, V. Chaudhri, P. Giorgini, Eric Yu (Eds.) LNCS 5600 Springer.

Appendix A: Usability Evaluation Tasks

Basic Tasks: Search and Navigation

- Search and navigate Words
- Search and navigate Senses
- Search and navigate Concepts related with Senses
- Discover the relations which are defined between the selected concepts

Search and navigation: Words , Senses, Concepts		
	User Tasks	Task time
1	Run Linguistic console from Firefox	
2	Search the word “Cat” from the Words panel	
3	Navigate the sense result list	
4	Click in a sense to populate the concept hierarch for the above senses and examine the relations.	
6	Find the concept where “Bengal tiger” is-a type of “cat”.	
7	Toggle the concept hierarchy from parent-child to child-parent.	
8	Find the concept where cat is a carnivore.	
9	Find its verb concept where cat has a more general concept “colonization”	
9	Navigate the concept hierarchy by changing the view to “Children to Parents” by clicking on the “Parents /Children” button form the concept panel and do the vice versa	
10	Find the corresponding Italian sense of “cat” from the sense panel by clicking on ‘show in other language’ button attached with senses	
11	Change the language using the language combo and repeat the above task	

Advance Tasks: CRUD on ontology

- Search and navigate the concept for a given Word
- View and edit the exceptional forms of a Word
- Modify the properties (e.g. the rank) of a Word w.r.t. a Sense
- Create/Delete/Update a Sense and its properties (the POS, the gloss, its set of words)
- Associate a Sense to a Concept
- Visualize all the relations for a given Concept
- Create/Remove a relation between two Concepts

CRUD on ontology		
	User Tasks	Task time
1	Run Linguistic console from Firefox	
2	Search the word “Cat” from the Words panel	
3	Create a new Sense for the word “Cat”: 1. Click on the add Sense button in the Sense panel. 2. A message box will appear and will ask the user weather he wants to create a new sense for “Cat” or just create a new Sense?	

	3. Press 'Ok'. 4~~.~~ Type the word in the word input box.~~.~~ 5. Specify the rank by entering any positive numeric value 6. Select the word as “preferred term” by selecting the preferred term checkbox. 7. Enter the natural language description of the sense in the gloss field. 8. Set the the part of speech of the word in the sense. 9. Finally, pressed the ‘Create Sense’ for final commit.	
4	Create a Sense as a new root concept: 1. Click on the add Sense button in the Sense panel. 2. A message box will be appear and will ask the user weather he wants to create a new Sense of “Cat” or just create a new Sense? 3. Press 'Cancel' just to create a new Sense" 4. Type the word in the word input box. 5. Specify the rank by entering any positive numeric value 6. Select the word as “preferred term” by selecting the preferred term checkbox. 7. Enter the natural language description of the Sense in the gloss field. 8. Set the the part of speech of the word in the Sense. 9. Finally, pressed the ‘Create Sense’ for final commit.	
6	Create a Sense to be associated with a concept with no Sense in the language: 1. Select a concept with the label “?” from the concept panel 2. A dialog box will be appear with the message “No concept description available in " Italian"! Press 'OK' to create concept description Or you might try by changing the language." 3. Press OK. 4. Look at the other languages to understand what the concept is about. To do this just change the language from the Word panel. If you find any concept in any other language then use this information to construct the the Sense information in the language you are working with. 5. Fill up the information for word and Sense as for the create Sense operation described in section 3. 6. Finally click on “Create Sense” button of the CUD panel.	
7	Create a Sense to be associated with a concept with no Sense in the language: 7. Select a concept with the label “?” from the concept panel 8. A message box will be appear with the message “No concept description available in " Italian"! Press 'OK' to create concept description Or you might try by changing the language." 9. Press OK. 10. Look at the other languages to understand what the concept is about. The user can do this just by changing the language from the Word panel. When the user selects a different language, all the information in the UI remains the same but it is shown in the selected language. If the user finds any concept in any other language then he can get some idea on the Sense information in the language he is working with. 11. Fill up the information for word and Sense as for the create Sense operation described in task 4. 12. User click on add word button of the CUD panel.	
8	Add a word to an existing Sense:	

	2. Selects a Sense from the Senses panel in which you wanted to add the word. 3. Click on the edit Sense button attached with the selected Sense. As a consequence, the CUD panel will be initialized with the information of the selected Sense. List of existing word(s) will appear in the words list box of the “manage Sense” segment. 4. Click on the add word button from the words list box in the “Manage Sense” panel which will activate the “Manage word” panel on the left of the CUD with blank values. 5. Type the word in the word text box and presses enter. If the word already exists (in any other Sense) the exceptional forms related to that word will be automatically shown. 6. Set the word as preferred term 7. Finally, press the Add Word button.	
9	Remove a word from a Sense: 1. Select the Sense from the Senses panel and click on the update Sense button. 2. The information for the Sense will appear in the CUD panel. The list of words for that Sense will appear in the words list box. 3. To remove a word click on the ❌ button of any word.	
10	Update a Word of a Sense (the properties of a word associated to a Sense): - Select the Sense from the Senses panel and click on the update Sense button. The list of words for that Sense will appear in the words list box. The preferred word will appear in the word input box together with its information. - Select any word from the words list that you wants to modify and change the following information of that word: - Exceptional forms. - Rank (in the Sense) - Is preferred term (in the Sense) - Finally, press the Update word button to commit the changes.	
11	Update Gloss and POS of a Sense: 1. Follow the same procedure as update a word of a Sense, 2. Finally press the Update Sense button after making the changes to gloss and POS.	
12	Delete a Sense: 1.To delete a Sense select the Sense from the Senses panel and click on the delete button ❌ attached with the Sense block	
13	Delete a Sense from one language and then recreate it: 1. Search the word H2o and select the Sense. 2. Check its corresponding Sense in Italian language 3. Delete this English Sense by clicking on the delete button ❌ attached with the Sense. 4. Change the language in Italian and search for the word H2o. 5. Click on the down arrow button so see its English meaning. 6. The English meaning will show “undefined?” and a ➕ button will appear with the description block. 7. Click on the ➕ button and create the sysnet (as described task 4) 8. Add the word “water” with h2o (as described task 8)	
14	Linking the (concept corresponding to the) Sense with an existing concept:	
15	Delete a relation: **1.** Select the Sense from the Senses panel and click on the update Sense button attached	

	with the Sense block. **2.** The list of relation for corresponding concept of that Sense will appear in the “Manage Relation” panel. **3.** To remove a relation click on the ✖ button of any relation from the Relation-Target table.	
16	Update a relation type: **1.** Select the Sense from the Senses panel and click on the update Sense button attached with the Sense. **2.** Select the relation from the Relation-Target table that you wanted to update. **3.** The selected relation will appear in the relation and target concept fields. Change the relation type (e.g., is-a, part-of) from the relation dropdown list to update. **4.** Finally click on “Update Relation” button to commit the change.	
17	Create a sample concept hierarchy: 1. Create three sample concepts Concept1, Concept2 and Concept3. 2. Make the Concept1 parent of Concept2 3. Make the Concept3 child of Concept1	

Appendix B: Heuristic Feedback

The feedback detailed below represents a heuristic review which provides important data to our design team while the development of Linguistic and Conceptual management console. It helps us find out the usability problems in the user interface and contribute as part of the iterative design process.

Introduction

The linguistic knowledge is the part of the UK that handles and provides the storage for the natural language information used to describe the concepts stored in the Concept Core. It defines words, their synonyms, word senses (grouped into synsets), and describe the senses of each word with natural language descriptions (glosses).

The Linguistic and Conceptual management console is the user interface which users access the semantic information. The goal of this user interface is to make user-friendly and effective management console in Sweb browser to perform CRUD operations (including search and navigation) on the Linguistic and Conceptual part for the Universal Knowledge.

Evaluation provided by: ______________________________

Email: ____

Date of evaluation: __________________________________

Three-point scale

• Low	The heuristic is far from satisfaction. Users may make mistake or cannot complete the task because the flaw in the interface
• Moderate	The heuristic is generally achieved. Users can finish their work but there is space for improvement
• High	The heuristic is highly achieved. Users are benefit from the heuristic and work efficiently

User Interface Checklist (a mark in the checkbox i*ndicates applicability)*

Visibility of System Status	How satisfied is this requirement?			Comment
☐ Visual feedback when sense/concepts are selected	___ Low	___ Moderate	___ High	__________________
☐ After the user completes an action, there is feedback that indicate the next action can be started	___ Low	___ Moderate	___ High	__________________
☐ Response times are appropriate to the task (Simple, frequent: < 1s; Common: 2-4s; Complex: 8-12s)	___ Low	___ Moderate	___ High	__________________
☐ A consistent icon design scheme across the system	___ Low	___ Moderate	___ High	__________________
☐ Prompts and Errors are visible to users	___ Low	___ Moderate	___ High	__________________
☐ The interface communicates what is the relation among classifications and nodes	___ Low	___ Moderate	___ High	__________________
☐ It is obvious how to operate the browsing tree	___ Low	___ Moderate	___ High	__________________
☐ The interface communicates what search scope is being searched	___ Low	___ Moderate	___ High	__________________
☐ It is obvious where in the search interface to enter a query	___ Low	___ Moderate	___ High	__________________
☐ Search result info. (search query scope, number of results) is clearly shown	___ Low	___ Moderate	___ High	__________________

Match Between System and the Real World	How satisfied is this requirement?			Comment
☐ Language and phrases familiar to user	___ Low	___ Moderate	___ High	__________________
☐ Icons are concrete and familiar	___ Low	___ Moderate	___ High	__________________
☐ Menu choices are readily understood meaning	___ Low	___ Moderate	___ High	__________________
☐ Search query is simple to define	___ Low	___ Moderate	___ High	__________________
☐ User can narrow the search results in a logical way	___ Low	___ Moderate	___ High	__________________

User Control and Freedom	How satisfied is this requirement?			Comment
☐ Users are facilitated in decision making and task processing	___ Low	___ Moderate	___ High	__________________
☐ Users are prompted to confirm operations that have destructive consequences	___ Low	___ Moderate	___ High	__________________
☐ Users can reverse their actions	___ Low	___ Moderate	___ High	__________________
☐ Users can move forward and backward between classifications/ nodes	___ Low	___ Moderate	___ High	__________________
☐ Query builder in advanced search	___ Low	___ Moderate	___ High	__________________

can be used effectively				
☐ Users easily leave search and start browsing for relevant content	___ Low	___ Moderate	___ High	__________________

Consistency and Standards	**How satisfied is this requirement?**			**Comment**
☐ Integers, real numbers and characters are aligned probably	___ Low	___ Moderate	___ High	__________________
☐ Embedded field-level prompts appear to the right of the field label	___ Low	___ Moderate	___ High	__________________
☐ Menu structure matches the task structure	___ Low	___ Moderate	___ High	__________________
☐ Attention techniques (intensity, size, font and color) are used with care	___ Low	___ Moderate	___ High	__________________
☐ Important information is placed at the beginning of the prompt	___ Low	___ Moderate	___ High	__________________
☐ User actions/system objects are named consistently, in grammatical and style, across the interface	___ Low	___ Moderate	___ High	__________________
☐ Interfaces and operations follow the design of offline(desktop) application paradigm	___ Low	___ Moderate	___ High	__________________

Help Users Recognize, Diagnose, and Recover from Errors	**How satisfied is this requirement?**			**Comment**
☐ Prompts are brief and unambiguous	___ Low	___ Moderate	___ High	__________________
☐ Messages place users in control of the interface	___ Low	___ Moderate	___ High	__________________
☐ Error messages suggest the cause of the problem	___ Low	___ Moderate	___ High	__________________
☐ Error messages indicate what action the user needs to take to correct the error	___ Low	___ Moderate	___ High	__________________

Error Prevention	**How satisfied is this requirement?**			**Comment**
☐ Menu choices are logical, distinctive and mutually exclusive	___ Low	___ Moderate	___ High	__________________
☐ The interface prevents users from making errors whenever possible	___ Low	___ Moderate	___ High	__________________
☐ Data entry boxes indicate the number of character spaces left	___ Low	___ Moderate	___ High	__________________
☐ Search box is long enough to handle common query lengths	___ Low	___ Moderate	___ High	__________________

Recognition Rather Than Recall	**How satisfied is this requirement?**			**Comment**
☐ Current processing's information is placed where the eye is likely to be				

looking in the screen	___ Low ___ Moderate ___ High	__________________
☐ Classifications/nodes have been grouped into logical and distinguish zones	___ Low ___ Moderate ___ High	__________________
☐ Browsing and searching zones have been separated clearly, however, they are recognized to be interdependent	___ Low ___ Moderate ___ High	__________________
☐ The search interface is located where users would expect it to be	___ Low ___ Moderate ___ High	__________________

Flexibility and Minimalist Design	**How satisfied is this requirement?**	**Comment**
☐ Multiple levels of error message detail are available for novice and expert users	___ Low ___ Moderate ___ High	__________________
☐ Users can jump across the classifications/nodes in the tree by using keyboard shortcut	___ Low ___ Moderate ___ High	__________________
☐ In the search keyword field, search is triggered by multiple actions	___ Low ___ Moderate ___ High	__________________
☐ Sorting options are helpful which make search results listed in a useful way	___ Low ___ Moderate ___ High	__________________

Aesthetic and Minimalist Design	**How satisfied is this requirement?**	**Comment**
☐ Information essential to decision making is displayed on screen	___ Low ___ Moderate ___ High	__________________
☐ All icons are visually and conceptually distinct	___ Low ___ Moderate ___ High	__________________
☐ Field labels are brief, familiar and descriptive	___ Low ___ Moderate ___ High	__________________

Help and Documentation	**How satisfied is this requirement?**	**Comment**
☐ Relevant information is enough for novice users to complete their tasks	___ Low ___ Moderate ___ High	__________________

Skills	**How satisfied is this requirement?**	**Comment**
☐ Operations are easy to learn and use	___ Low ___ Moderate ___ High	__________________
☐ Users are the initiators of actions rather than responders	___ Low ___ Moderate ___ High	__________________
☐ The method of moving the cursor to the next or previous object both are simple and visible	___ Low ___ Moderate ___ High	__________________

Pleasurable and Respectful Interaction

with the User	**How satisfied is this requirement?**	**Comment**
☐ Amount of necessary information has been kept to a minimum	___ Low ___ Moderate ___ High	__________________
☐ Search query field automatically submit search keyword after a time-out is appropriate	___ Low ___ Moderate ___ High	__________________

Privacy	**How satisfied is this requirement?**	**Comment**
☐ Protected or confidential areas can be accessed with certain password	___ Low ___ Moderate ___ High	__________________

Summary comments & enhancement suggestions: ______________________________
____
____