An Approach for Indexation, Classification and Retrieval of Knowledge Sources in Collaborative Environments

Ruben Costa
Centre of Technology and Systems
UNINOVA
Quinta da Torre, Portugal
rddc@uninova.pt

Celson Lima
Federal University of Western Pará
UFOPA / IEG / PSI
Santarém, Brasil
celsonlima@ufpa.br

***Abstract*—This work introduces a conceptual framework and its current implementation to support the semantic enrichment of knowledge sources. It improves the ability for indexing and searching of knowledge sources, enabled by a reference ontology and a set of services which implement the searching and indexing capabilities. Essentially, our approach defines an appropriate knowledge representation based on semantic vectors which are created using three different but complementary algorithms for each knowledge source, using respectively the concepts and their equivalent terms, the taxonomical relations, and ontological relations. We introduce the conceptual framework, its technical architecture (and respective implementation) supporting a modular set of semantic services based on individual collaboration in a project-based environment (for Building & Construction sector). The main elements defined by the architecture are an ontology (to encapsulate human knowledge), a set of web services to support the management of the ontology and adequate handling of knowledge providing search/indexing capabilities (through statistical/semantically calculus). This paper also provides some examples detailing the indexation process of knowledge sources, adopting two distinct algorithms: "Lexical Entries-based" and "Taxonomy-based". Results achieved so far and future goals pursued here are also presented.**

Keywords-Knowledge Engineering; Ontologies; Indexation; Classification; Retrieval

I. INTRODUCTION

Over the last two decades, the adoption of the Internet as the primary communication channel for business purposes brought new requirements especially considering the collaboration centred on engineering projects. Engineering companies are project oriented and successful projects are their way to keep market share as well as to conquer new ones. From the organisation point of view, knowledge goes through a spiral cycle, as presented by Nonaka and Takeuchi in the SECI model [1]. It is created and nurtured in a continuous flow of conversion, sharing, combination, and dissemination, where all the aspects and dimensions of a given organisation, are considered, such as individuals, communities, and projects.

Knowledge is considered the key asset of modern organisations and, as such, industry and academia have been working to provide the appropriate support to leverage on this asset [2]. Few examples of this work are: the extensive work on knowledge models and knowledge management tools, the rise of the so-called knowledge engineering area, the myriad of projects around 'controlled vocabularies' (e.g., ontology, taxonomies, thesaurus), and the academic offer of knowledge-centred courses (graduation, master, doctoral).

As relevant literature shows [3]; [4]; [5]; [6], knowledge management (KM) does not only comprise creation, sharing, and acquisition of knowledge, but also classification, indexation, and retrieval mechanisms (see Figure 1). Knowledge may be classified by its semantic relevance and context within a given environment (such as the organisation itself or a collaborative workspace). This is particularly useful to: (i) improve collaboration between different parties at different stages of a given project life cycle; and (ii) assure that relevant knowledge is properly capitalised in similar situations. For example, similar projects can be conducted in a continuously improved way if lessons learned from previous are promptly known when a new (and similar to some previous one) project is about to begin.

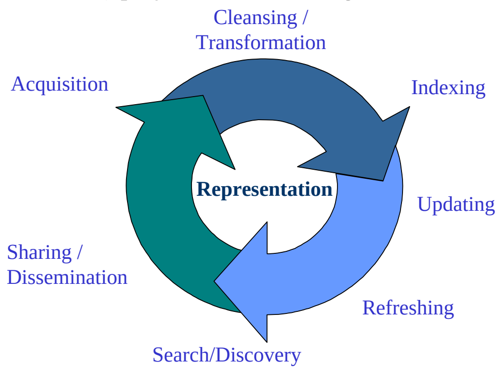

Figure 1. KM Lifecycle

Semantic systems utilize an ontology (or a set of ontologies) to encapsulate and manage the collection and representation of relevant knowledge, hence giving information a human-relevant meaning. Semantic description of project resources enhances collaboration through better understanding of document contents (supporting better understanding and extraction of knowledge) [7]. In addition, by introducing ontological reasoning, semantic techniques enable discovery of knowledge and information that was not part of the original use case or purpose of the ontology itself [8].

The work presented here provides project teams with semantic-enabled services, targeting the improvement of the semantic richness of knowledge sources (KS) used/created, during the execution of an engineering project. The work conceptually covers two dimensions, namely collaboration and knowledge engineering, focused on ontology development and knowledge sharing activities [9]. Knowledge, the dimension particularly explored in this paper, relates to the 'currency' being exchanged during a collaborative process, in this case a collaborative engineering process. Technical documents, lessons learned, and expertise, are some examples of such currency.

This paper is structured as follows. Section 2 defines the objectives and addresses the problem to be tackled. Section 3 introduces the software components handling the knowledge related matters previously introduced. Section 4 gives illustrative examples of the software operation. Section 5 explains the need for conducting more empirical results. Finally, section 6 concludes the paper and points out the future work to be carried out.

II. Related work

Index terms are traditionally used to characterize and describe the semantics of a document. Such approach attempts to summarize a whole document with a set of terms that are relevant in the context of the document. While this approach has given some satisfactory results in the area of Information Retrieval (IR), it still has some limitations as it proceeds by oversimplifying the summarization process by relying on a subset of relevant terms that occur in a document, and uses these as a mean to convey the semantics of the document. The most commonly used IR models are: Boolean, Vector and Probabilistic [14]. In the Boolean model, documents are represented as a set of index terms. This model is said to be set theoretic [15]. In the Vector model, documents are represented as vectors in a t-dimensional space. The model is therefore said to be algebraic. In the probabilistic model, the modelling of documents is based on probability theory. The model is therefore said to be probabilistic. Alternative models that extend some of these classical models have been developed recently. The Fuzzy and the Extended Boolean Model have been proposed as alternatives to the set theoretic model. The Generalized Vector, the Latent Semantic Indexing, and the Neural Network models have been proposed as alternatives to the Algebraic Model. The Inference Network, and the Belief Network models have been proposed as an alternative to the Probabilistic Model.

It is also worth mentioning that models that reference the structure, as opposed to the text, of a document do exist. Two models have emerged in this area: the Non-Overlapping Lists model [16] and the Proximal Node model [17]. Our approach enhances the vector-space model for IR, by adopting an ontology based implementation. It implements the notion of semantic vectors, which takes into account the taxonomical and ontological relations between concepts, which is an aspect that is neglected by most of IR approaches nowadays.

The e-cognos project [12] addressed this issue, but its major outputs remain only at a first level of IR, described in this work as lexical entries based indexation. A more recent work also addresses this theme, by enhancing the vector space-model [13], but it does not take into account the ontological and taxonomical relations of ontology concepts, adopting a different approach as the one presented in this work.

III. Relevance of the Work

The key question guiding the development of this work is: How to augment the relevance of knowledge sources in collaborative engineering projects in order to support users within problem-solving interactions?

The traditional method of turning data into knowledge relies on manual analysis and interpretation. For example, in the building & construction domain, it is common for specialists to periodically conduct several simulations before start building, on a regular basis. The specialists then provide a report detailing the analysis to the building owners and building contractors organizations; this report becomes the basis for future decision making and planning for building & construction.

This form of manual probing of a data set is slow, expensive, and highly subjective. In fact, as data volumes grow dramatically, this type of manual data analysis is becoming completely impractical in many domains. Who could be expected to digest millions of records, each having tens or hundreds of fields? We believe that this job is certainly not one for humans; hence, analysis work needs to be automated, at least partially.

On the other hand, systems are normally focused on the management of structured information, but they also include a wide range of unstructured information in the form of documents, drawings, images, etc.. Thus, although there might be an understood relationship between a document and a part of the product structure, there are still concerns about how to more effectively make the information and knowledge stored in such systems available to and useful for a wide range of actors in collaborative environments.

In comparison to structured information, the unstructured information lacks context, and since there are no predetermined data types or established relationships between dispersed pieces of information, it is often difficult to find such information if you do not know exactly what you are looking for. For example, when searching for documentation of a certain decision, it might be needed to browse through a vast amount of e-mail, meeting notes, spreadsheets, or blog posts, and the only help available is usually a free-text search that does not always return relevant results. In the specific case of documents, it is often to find metadata in the form of the file name, the date it was created, the version history, the name of the person who created the document, but this information usually says little about the relevance and usefulness of the actual content.

It is important to highlight that a document, or any other kind of unstructured piece of information that has been stored in a database, does not mean that the content is easily retrieved or analysed beyond the individual or team that took part in the creation of the document.

A. *Objectives*

The main objective pursued here is related with capturing and reuse of knowledge, by adopting an ontology-based approach using semantic and statistical/proximity based algorithms to better augment the relevance of knowledge sources created/used within collaborative engineering projects. In this sense, the key capabilities to be provided are the following:

- Knowledge documentation and storage: support a consistent approach for documenting lessons learned in ontology-based system that allows semantic retrieval of documents.
- Knowledge classification: knowledge classification is a highly desirable functionality and one having a high priority. Existing tools only allow for the categorisation of knowledge. It is more important to support knowledge item clustering (finding similarities between knowledge items).
- Search for knowledge items: the search, discovery, and ranking of knowledge items are issues of high priority with respect to both the manner in which these are done and in terms of the different types of knowledge items considered (full text search; searches on the basis; and discovery of experts and communities).

This work aims to provide the best ontological representation for a given knowledge source within a given context, when adding/searching for knowledge. When adding a new knowledge into the knowledge repository, the approach being implemented will extract the best relevant keywords from the KS and calculates their statistical weights. This set of keywords/weights forms the basis of the so-called Semantic Vector (SV), which is analysed against the ontology in order to get the ontological representation of the KS, which is defined by concepts from the ontology. A knowledge representation is then built for the KS and stored into the repository. This is going to be explained more clearly in the following sections.

When searching for knowledge, the system analyses the queries in order to get the appropriate ontological representation. Effectively, the system finds the knowledge representations that best match the concepts in order to get the relevant KS from the knowledge repository for a given query.

IV. Technical Architecture & Choices

The technical architecture supporting the software infrastructure conceived here as our proof of concept is structured in three main layers: Knowledge Repository, Knowledge Services, and User Interface.

Knowledge repository layer holds the domain knowledge, creating a sort of knowledge space, which is organised around three key entities: Knowledge Sources, their respective Knowledge Representations, and the Ontology itself, which comes with its ambassador, the Ontology Server.

Knowledge Sources are elements which represent the corporate memory of an organization, i.e., documents, spread sheets, media files, and similar sources that can be used to support the acquisition or creation of knowledge. The KS repository represents, then, the collection of all KS currently available in the knowledge space.

When a new KS is added into the knowledge space, its respective knowledge representation is created by the system in order to characterise such KS. The knowledge representation includes some basic information about the KS and adds its specific semantic vector. Broadly speaking, a semantic vector (which will be described in detailed in further sections) gives the best ontological representation to index the KS just added into the space. Therefore, the knowledge representations repository is a container that aggregates all knowledge representations currently available in the knowledge space.

The ontology holds concepts, axioms and relations used to represent knowledge in the domain of work. In our case, the ontology is structured as a pair of taxonomies, as follows: (i) taxonomy of concepts connected via pure taxonomical relations (e.g., as is); and (ii) taxonomy of relations, which contains ontological relations (other than the pure taxonomical ones) also used to improve the semantic links of ontological concepts. These taxonomies are used in different phases of the semantic vector creation, which is also described in detail in further sections. The ontology server is then a software component acting as the ontology ambassador, which means, it provides the way to access to any ontological data.

The knowledge services layer offers the key semantic services used in the knowledge space, namely indexing, discovery, and maintenance, which are respectively provided by the following components: Indexer, Discover, and Maintener. From interoperability point of view, it is worth mentioning that knowledge services are provided as a set of web-services.

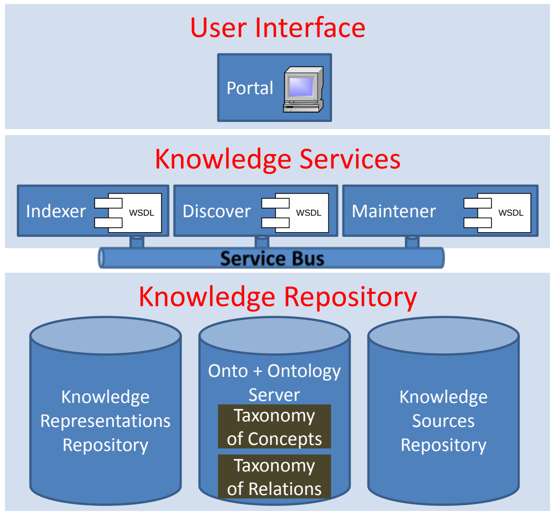

Figure 2. Technical architecture

The User Interface layer offers the front-end with the user via web portal, enabling users to interact with the knowledge space.

In terms of technical choices, two points are highlighted. Firstly, the adoption of the Web services model also plays a very strategic role regarding openness, interoperability, and integration of the system. We use Web Services Description Language (WSDL) [18] to specify the knowledge services, which can also be used to integrate any additional service deemed necessary to our system and which can be provided by third party. Having the WSDL file describing a given web service it is easy to produce the web client able to invoke that service. Thanks to this mechanism, all knowledge services currently provided are available to any web application in the same way that the system interoperates with any other web application.

Secondly, the Java language was chosen due to its key features, which are platform independence and open source model.

A. *The Ontology*

Knowledge sources strongly rely on ontological concepts, as a way to reinforce their semantic links. The ontology uses a taxonomy of concepts holding two dimensions: on one hand, the knowledge sources themselves are represented in a tree of concepts and, on the other hand, the industrial domain being considered. Instances of concepts (also called individuals) are used to extend the semantic range of a given concept. For instance, the ontological concept of 'Design_Actor' has two instances to represent architect and engineer as roles that can be considered when dealing with knowledge sources (see Figure 3) related to design (experts, design-related issues/solutions, etc.). Moreover, each ontological concept also includes a list of terms and expressions, called equivalent terms, which may represent synonyms or expressions that can lead to that concept. Ontology support is particularly useful in terms of indexation and classification towards future search, share and reuse.

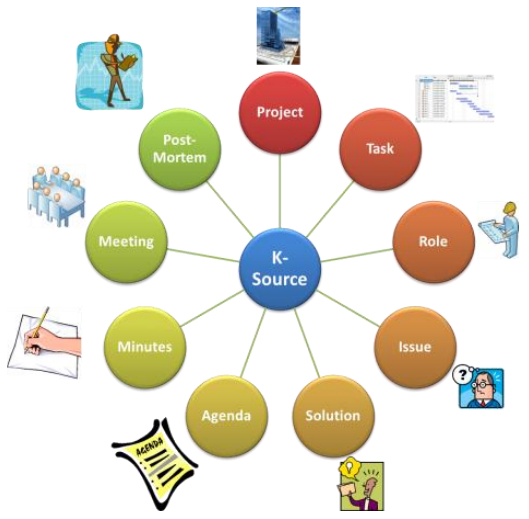

Figure 3. Instances of Knowledge Sources.

The ontology is developed to support and manage the use of expressions which contextualize a KS within the knowledge repository. The ontology adds a semantic weight to relations among KS stored into the knowledge repository. Every ontological concept has a list of 'equivalent terms' that can be used to semantically represent such concept. These terms are, then, treated in both statistical and semantic way to create the semantic vector that properly indexes a given KS.

The ontology was not developed from scratch; rather, it has been developed taking into account relevant sources of inspiration, such as the buildingsmart IFD model [10], [11], and the e-cognos project [12].

The basic ontological definition is as follows: a group of Actors uses a set of Resources to produce a set of Products following certain Processes within a work environment (Related Domains) and according to certain conditions (Technical Topics). Other domains define all relevant process attributes. For example, the Technical Topics domain defines the concepts of productivity, quality standard and duration.

B. *The Services*

The semantic support services that compose the API layer can globally be described as the following:

- Indexing: The service is designed to accept a list of keywords, compare the keywords to ontological concepts, and produce a ranked list of ontological concepts that best matches that list of keywords. For each keyword, it calculates a corresponding weight reflecting its relevance. The set of keyword-weight pairs is the semantic vector of the knowledge source. This vector is then used to assign a hierarchy of relevant metadata to each knowledge source.
- Discover: The service enables the user to perform searches across knowledge elements, is invoked whenever a user requests a search for a set of keywords. The service produces a matching ontological concept for these keywords, and then matches the resulting concept to the metadata of target knowledge source. This ontology-centred search is the essence of semantic systems, where search phrases and semantic vectors are matched through ontological concepts.
- Maintener: This service is responsible for managing the domain ontology enabling the following capabilities: Browse the concepts/relations(allows navigation through the ontology, showing the description of both concepts and relations); Create new concept(allows the addition of a new concept into the ontology); Create new relation(allows the addition of a new relation into the ontology); Create new attribute(allows the addition of a new attribute to a concept); Import OWL ontology; and Remove concept(allows removal of a concept from the ontology).

V. Indexation Process

To better understand the indexation process through semantic vectors comparison (Figure 4), it is necessary to understand how and where these are created and used.

Each semantic vector contains the necessary ontological concepts that best represent a given knowledge source when it is stored into the knowledge repository. These concepts are ordered by their semantic relevance regarding the KS. KS are compared and matched based on their semantic vectors and the degree of resemblance between semantic vectors directly represents the similarity between KS.

Semantic vectors are automatically created using project-related knowledge, using a process which collects words and expressions, to be matched against the equivalent terms which represent the ontological concepts. This produces an inventory of: (i) the number of equivalent terms matched at each ontological concept; and (ii) the total number of equivalent terms necessary to represent the harvested knowledge. This inventory provides the statistical percentage of equivalent terms belonging to each ontological concept represented in the universe of harvested knowledge. This step represents, the calculus of the 'absolute' semantic vector of a given KS, taking into account the equivalent terms-based percentages.

However, the approach presented here also considers a configurable hierarchy of KS relevance, as part of the creation of semantic vectors. This hierarchy is defined using 'relative' semantic factors to all types of KS, which ranges respectively from low relevance (0) to high relevance (1) for the context creation. Both hierarchy and relative semantic factors can be changed if necessary, depending on what KS are considered most relevant for the indexation process.

The final step, which comprehends the semantic evaluation, also includes ontological concepts that are not linked to the knowledge gathered, but have a semantic relationship of proximity with a relevant (heavy) ontological concept. This is done through the definition of a secondary semantic factor to ontological concepts based on their relative distances, inside the ontology tree.

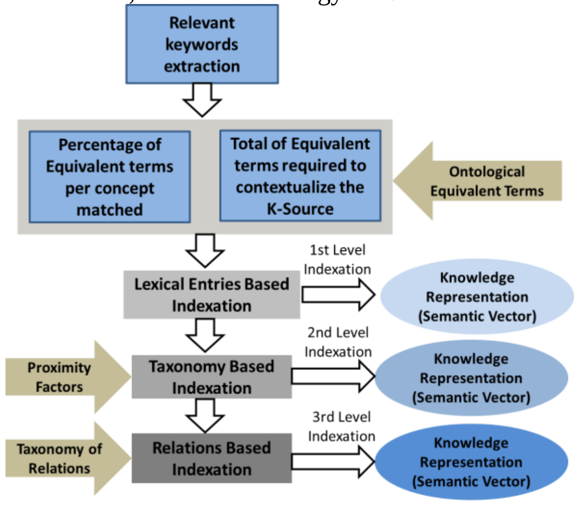
Relevant keywords extraction
Percentage of Equivalent terms per concept matched
Total of Equivalent terms required to contextualize the K-Source
Ontological Equivalent Terms
1st Level Indexation
Lexical Entries Based Indexation
Knowledge Representation (Semantic Vector)
2nd Level Indexation
Proximity Factors
Taxonomy Based Indexation
Knowledge Representation (Semantic Vector)
3rd Level Indexation
Taxonomy of Relations
Relations Based Indexation
Knowledge Representation (Semantic Vector)

Figure 4. Semantic Vector creation process

Summing up, the final calculation of the semantic vector includes: statistical percentages based on the equivalent terms, the hierarchy of relevance for KS, and the weight assigned to the proximity level.

As referred previously, semantic vectors are continuously updated through the project's life cycle, and even in project's post-mortem. This is done in order to maintain the semantic vector's coherence with the level of knowledge available. Semantic vectors are automatically created: (i) whenever a new KS is gathered; and (ii) to help answering queries issued by the users.

Our approach provides three algorithms to perform the process of retrieving the best ontological representation and weight for both the KS and the query. Those algorithms, namely "Lexical Entries based", "Taxonomy based" and "Relation based" work as follows:

- Lexical Entries: each concept is defined with a list of lexical entries in a different language. The algorithm gets all the lexical entries of all the concepts of the ontology and matches them with all the keywords in the semantic vector. Therefore at the end of the first step, a list of concepts (Lc) matching the semantic vector is built. Further, the weight of the concept C (Wc) is calculated for all the concepts in the list applying the following formula:

$$Wc=NKm \div NKsm \tag{1}$$

where:
Wc: weight calculated to the concept.
NKm: number of keywords that match the concept C.
NKsm: number of keywords in the semantic vector.

- Taxonomy: The algorithm starts from the list "Lc" built in the "Lexical Entries based" algorithm and provides a different way to arrive at the weights. The aim is to try to increase the weight of the concepts which may have received a poor weight in the first stage trying to see if they are close in the taxonomy to a concept that received a good weight in the first stage. The "Lc" list gets the best concepts that match the keywords. A concept is considered a best concept when its weight exceeds the value "best-concept-range" defined in the parameters table. The others are named "worth concepts". For each best concept, the algorithm checks if there are worth concepts nearby concepts of the Lc list in the taxonomy. If this is the case, their respective weights are augmented according the following formula:

$$Wc=Wbc \times Vp \tag{2}$$

where:
Wc: weight performed for the concept
Wbc : weight of the best concept
Vp: value got in the parameters table depending on the level and the way.

The Vp is a value between 0 and 1 and depends on the distance between the best concept and the worth concept in the hierarchy of concepts. The weight of the new concept is only updated in case the weight given to the preformed concept is greater than the old one. This step is implemented as a way to promote concepts that are strongly taxonomically-related with the best matched concept. Analogously, other concepts that are not so strongly taxonomically-related with the best concept match are penalized.

- Relation: this algorithm will be available in the next version of the system. It aims to integrate the richness of the relations among the concepts in order to provide a more powerful way to represent a KS.

A. *Example*

The list of ontological concepts that best represents a knowledge source is ranked according to the ontological weights assigned to each concept. As stated, there are three ways to calculate such a weight, namely: equivalent terms-based, taxonomy-based, and fully ontology-based. The equivalent terms represent the keywords related to each concept (synonyms or words that can be associated to that concept). They are then used as "indexes" to access the concepts, therefore using purely "statistics" (the greater the number of equivalent terms of a given concept found in the KS representation, the heavier the concept becomes). The taxonomy-based way takes the previous weight and refines it using the "is a" relation to navigate around the heaviest concepts and augment the weight of neighbouring concepts (this augmentation is based on a configurable table of factors guiding generalization/specialization of the taxonomy). The fully ontology-based method exploits all the relations that start from the heaviest concepts to augment the neighbouring concepts (augmentation process is similar to the taxonomy-based one).

Figure 5 and Figure 6, present the results of calculating weights using the equivalent term- and the taxonomy-based methods. The KS representation is given by such concepts: "Heat Pump; Product; Cooling Tower; Solar Collector; Climate Control; Central Heat Generator; Waste Management; Transformation and Conversion; Fan; Extractor; Air Ductwork; Steam Treatment" (column Keywords) and the respective concepts found that match it (column Concepts). The column “Lexical” show the first weight calculated, that is the ratio between number of keywords related to one concept and the total number of keywords in the query (e.g., for the first concept, Transformation and Conversion, the value 0.417 comes from 5 divided by 12). This is a very straight forward calculation.

TABLE I. LEXICAL TERMS VS TAXONOMY BASED

Concepts(7)	Keywords(12)	Keywords(#)	Lexical	Taxonomy
Transformation and Conversion	Heat Pump; Cooling Tower; Solar Collector; Central Heat Generator; Transformation and Conversion	5	0,417	0,417
Product	Product	1	0,083	0,055
Climate Control	Climate Control	1	0,083	0,25
Waste Management	Waste Management	1	0,083	0,055
Impelling Equipment	Fan, Extractor	2	0,167	0,111
HVAC Distribution Device	Air Ductwork	1	0,083	0,055
Energy Treatment	Steam Treatment	1	0,083	0,055

Figure 5 shows the comparative results of the two methods. It is possible to detect immediately that some concepts have had their weights increased. After calculating the first weight, the taxonomy-based method is applied, where it is evaluated the neighbourhood of the heaviest concept(s) and, by following their taxonomical relations, raises the weight of the neighbouring concepts.

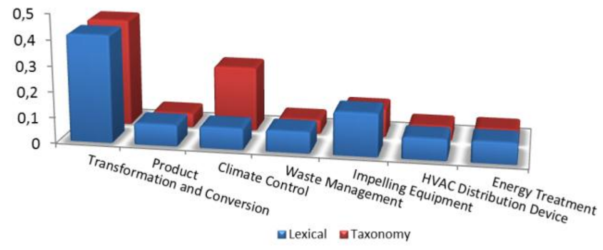

Figure 5. Comparison between equivalent term- and taxonomy-based weights

The initialization of such weights is done manually using a table of values that expresses the factors to be used when augmenting a super/sub concept. This table is configured by the user. It is worth noticing that the taxonomy-based method always keeps the higher weight if the taxonomy-based weight is going to be smaller than the equivalent term-based weight.

Finally, Figure 6 illustrates how the taxonomical relations are used to raise the weight of neighbouring concepts.

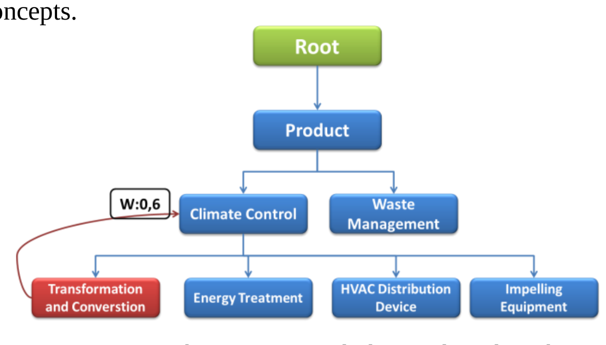

Figure 6. Using the taxonomy to calculate ontological weights.

In this example "Transformation and Conversion" is the heaviest concept with a weight of 0.417, having the neighbour "Climate Control". Therefore, using the factor configured by the user (0.6) the weights of the neighbours are recalculated. As consequence, "Climate Control" is augmented whilst "Product", "Waste Management", "Impelling Equipment", "HVAC Distribution Device", "Energy Treatment" are penalized proportionally.

VI. Results and Achievements

This research work is still an on-going process, where relevant empirical data and conclusions aren't not yet matured. An assessment which compares our solution with already existing ones is not yet developed, due to the fact the empirical data and conclusions can't be drawn yet. Work developed so far includes the establishment of the lexical entries and taxonomy based algorithms and the improvement of the taxonomy based algorithm with the inclusion of heuristics which enable the weights associated with the taxonomy relations to change dynamically.

VII. Conclusions

This work brings a contribution focused on collaborative engineering projects where knowledge engineering plays the central role in the decision making process.
Key focus of the paper is the indexation and retrieval of knowledge sources provided by semantic services enabled by a domain ontology. This work specifically addresses collaborative engineering projects from the Construction industry, adopting a conceptual approach supported by knowledge-based services. The knowledge sources indexation process is supported using a semantic vector holding a classification based on ontological concepts.

When addressing collaborative working environments, there is a need to adopt a semantic description of the preferences of the users and the relevant knowledge elements (tasks, documents, roles, etc.). In this context, we foresee that knowledge sources can be semantically enriched when adopting the indexation process described within this work

Ontologies which support semantic compatibility for specific domains should be adaptive and evolving within a particular context. Ontologies ability to adapt to different environments and different context of collaboration is of extremely importance, when addressing collaborative engineering projects at the organizational level.

As future work regarding this research topic, there is a need to further analyse into what extent neighbours concepts can influence the calculus of the semantic vector as well as how ontological relations can contribute also to the better representations of knowledge given by the semantic vector.

References

[1] I. Nonaka and H. Takeuchi, The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation. New York: Oxford University Press, 1995

[2] J. Firestone and M. McElroy, Key Issues in the New Knowledge Management. Burlington: Butterworth-Heinemann, 2003.

[3] M. Koenig, The third stage of KM emerges. KMWorld, March 2002.

[4] Y. Malhotra and Y, Beyond "Hi-Tech Hidebound" Knowledge Management: Strategic Information Systems for the New World of Business. BRINT Research Institute, 1999.

[5] M. McElroy, "The Second Generation of Knowledge Management," Knowledge Management Magazine, Oct 1999, pp. 86-88.

[6] K. Dalkir, Knowledge Management in Theory and Practice. Oxford: Elsevier, 2005.

[7] A. Zeeshan, A. Chimay, R. Darshan, P. Carrillo, and D. Bouchlaghem, "Semantic web based services for intelligent mobile construction collaboration," ITcon, 367-379, 2004.

[8] O. Lassila and M. Adler, "Semantic Gadgets: Device and information interoperability". Cleveland: Ubiquitous Computing Environment workshop, 2003.

[9] R. Costa, C. Lima, J. Antunes, P. Figueiras, and V. Parada, "Knowledge management capabilities supporting collaborative working environments in a project oriented context," European Conference on Intellectual Capital (pp. 208-216). Lisbon: ACI, 2010.

[10] buildingSMART. Retrieved: September, 2011 from IFD Library: http://www.ifd-library.org/

[11] OCCS. Retrieved: September, 2011 from OmniClass: http://www.omniclass.org/

[12] C. Lima, B. Fies, T. Diraby, and G. Lefrancois, "The challenge of using a domain Ontology in KM solutions: the e-COGNOS experience," International Conference on Concurrent Engineering: Research and Applications, (pp. 771-77). Funchal, 2003.

[13] P. Castells, M. Fernandez, and D. Vallet. An Adaptation of the Vector-Space Model for Ontology-Based Information Retrieval. IEEE Trans. on Knowl. and Data Eng. 19, 2 (February 2007), pp. 261-272.

[14] R. Baeza-Yates and B. Ribeiro-Neto. Modern Information Retrieval, Addison Wesley, 1999.

[15] V. Gudivada, V. Raghavan, W. Grosky, and R. Kasanagottu. Information retrieval on the World Wide Web. Internet Computing, IEEE , vol. 1, no.5, pp. 58-68, Sep/Oct 1997

[16] F. Burkowski, "An algebra for hierarchically organized text-dominated databases", Information Processing & Management, 28(3), 333–348, 1992

[17] G. Navarro and R. Baeza-Yates. Proximal Nodes: A Model to query document databases by contents and structure, ACM Transactions on Information Systems 15 (4), October 1997, pp. 401-435.

[18] Web Services Description Language (WSDL) Retrieved: September, 2011 from World Wide Web Consortium: http://www.w3.org/TR/wsdl