Towards Unification of Requirements Engineering Approaches using Semantics-based Process

Imed Eddine Saidi[1,2], Taoufiq Dkaki[1], Nacer Eddine Zarour[2] and Pierre-Jean Charrel[1]
[1]*IRIT Laboratory, University Toulouse 2 Mirail Maison de la Recherche, 5 Allée Antonio Machado, 31058 Toulouse, Cedex 9, France*
[2]*LIRE Laboratory, University Constantine 2, BP 325, Route Ain El Bey 25017 Constantine, Algeria*

Keywords: Requirements Engineering, WordNet, Semantic Similarity, Concepts Abstraction.

Abstract: In Requirements Engineering, there exist different kinds of approaches such as goal-oriented, viewpoint-oriented and scenario-oriented approaches to specify companies' needs. These companies use these different approaches to elicit, specify, analyse and validate their requirements in different contexts. The globalization and the rapid development of information technologies sometimes require companies to work together in order to achieve common objectives as quickly as possible. In this paper, we propose a unified requirements engineering meta-model which allows cooperation in the requirements engineering process between heterogeneous systems. This meta-model is based on the abstraction of different kinds of approaches to benefit from all advantages that already exist in the other requirements engineering approaches while taking into account interoperability.

1 INTRODUCTION

"Requirements engineering (RE) is the process of discovering, documenting and managing the requirements for a computer-based system" (Sommerville and Sawyer, 1997a).

In Requirements engineering, companies have different cultures and use different kinds of tools and approaches to describe and manage upstream phases of software projects such as goal-oriented, viewpoint-oriented and scenario-oriented approaches. The globalization and the rapid development of Information Technologies sometimes require companies to work together in various fields including RE in order to achieve common objectives as quickly as possible. Another thing, these companies are not ready to agree on a unique RE approach to cooperate because of the time and the cost that result from the migration. Our goal in this paper is not to impose a new way of companies working around one approach but to propose a unified RE meta-model (UREM) as an intermediary of communication between different types of meta-models of RE approaches in order to allow cooperation between these approaches.

Bendjenna et al. (2010) have proposed an integrated approach MAMIE which combines different kinds of concepts: goal, scenario and viewpoint in order to allow cooperation between companies. In i* approach, there exists different variations for particulars usages. Carlos and Xavier (2011) have defined super meta-model hosting identified variations of i* and implementing a translation algorithm between these different variations oriented to semantic preservation. Our work intends to be a combination between the two works. We propose an abstract meta-model which allows cooperation and translation of information between different kinds of RE approaches.

This paper is organized in six sections. In section two we present an overview of the idea behind UREM. In section three, we present our three-steps unification process. In section four, we draw UREM. In section five, we discuss the performance of the implementation of the unification process. Finally, we conclude and draw perspectives of this work.

2 TOWARDS UNIFIED RE METAMODEL

The aim of this paper is to propose a unified meta-model (UREM) that makes companies cooperate with each other to achieve common objectives.

Eddine Saidi I., Dkaki T., Eddine Zarour N. and Charrel P..
Towards Unification of Requirements Engineering Approaches using Semantics-based Process.
DOI: 10.5220/0004623804430450
In *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval and the International Conference on Knowledge Management and Information Sharing* (KMIS-2013), pages 443-450
ISBN: 978-989-8565-75-4

UREM is an intermediary of communication and information translation between different types of RE Meta-models in order to allow cooperation between them.

Each RE Meta-model is composed of a set of concepts. Thus, communication between two RE meta-models is a communication between different concepts of these two meta-models. We start by a simple example to understand the idea behind our work. Suppose that two persons want to work together on a common objective to achieve it as quickly as possible. These persons speak different Languages *LangA* and *LangB*. To make these persons work together, we should find a common ground that brings the two persons to understand each other. The two persons speak languages but these languages are different. So, to make these persons communicate together we need a *Translator* which knows the two languages (*Language concept*) *LangA* and *LangB*. The concept *Language* is the common ground of *LangA* and *LangB* and through this ground there exist a translation rules to perform two-way translation between *LangA* and *LangB*. We say that *LangA* and *LangB* are similar in the context that the two are languages and share *Language Concept*. Thus, the concept *Language* is an abstraction of *LangA* and *LangB*.

From this idea, we are looking to create a new meta-model which is composed of a set of classes where each class is an abstraction of a set of concepts (similar concepts) that exist in different RE Meta-models. From the example, we have created an *Abstract Class* (*language*) which is an abstraction of *ClassA* (*LangA*) and *ClassB* (*LangB*).

3 ABSTRACTION OF RE METAMODELS STEP BY STEP

In this section, we present our approach to unify existing RE meta-models. The principle of this approach as mentioned in the previous section is to find sets of new concepts that are abstractions (merging) of different concepts from different RE meta-models. In other words, a group of similar concepts from different approaches represents one abstract concept. Finding similarities between RE concepts is then a key issue in our process. There exist different methods to find similarities between objects such as structural similarities (Vincent et al., 2004) as used in a previous paper (Saidi et al., 2012), syntactic similarities and semantic similarities.

In this paper, we adopt a more rigorous process that is more concerned with the meaning of RE concepts (semantic process). Our process is based on WordNet (George, 1995) to find semantic relationships and similarities between words which represent RE concepts (words are the only thing that we get to apprehend RE concepts).

WordNet is a large lexical database which is available online and provides a large repository of English lexical items. The smallest unit in WordNet called *synset*, which represents a specific meaning of a word. It includes the word, its explanation, and its synonyms. Each sense of a word is in a different *synset*. Synsets are equivalent to senses = structures containing sets of terms with synonymous meanings. Each *synset* has a *gloss* that defines the concept it represents. For example, the words night, nighttime, and dark constitute a single synset that has the following gloss: the time after sunset and before sunrise while it is dark outside. Synsets are connected to one another through explicit semantic relations. Some of these relations (hypernym, hyponym for nouns, and hypernym and troponym for verbs) constitute is-a-kind-of (holonymy) and is-a-part-of (meronymy for nouns) hierarchies.

For example, tree is a kind of plant, tree is a hyponym of plant, and plant is a hypernym (abstraction) of tree. Analogously, trunk is a part of a tree, and we have trunk as a meronym of tree, and tree is a holonym of trunk. If there is more than one sense, WordNet organizes them in the order of the most frequently used to the least frequently used.

Our aim is to perform cooperation between different types of approaches. In this paper, we choose one approach from each type of RE approaches in order to achieve our goal, regardless of the RE approach chosen, our unification process is applicable to various other approaches. In this paper, we deal with approaches that are widely used: i* (Yu, 1995) as goal oriented approach, CREWS (Sutcliffe et al., 1998) as scenario oriented approach and PREview (Sommerville and Sawyer, 1997) as viewpoint oriented approach. We denote:

$$A = \{A_1 = i^*, A_2 = \text{CREWS}, A_3 = \text{PREview}\} \quad (1)$$

Each approach A_i has a set of concepts $A_i = \{c_{i1}, c_{i2}, \ldots, c_{in}\}$ and each concept c_{ij} has a name and a definition def_{cij}. We distinguish two categories of concepts:

- Concepts of category one: these concepts (names of concepts) are represented in WordNet as *synsets* and we can get directly the definition and the different semantic relationships between them. The most of these concepts are represented

as one word, but it is not always the case, for example: viewpoint concept.

- Concepts of category two: these concepts are not represented as synsets in wordnet. These concepts are mostly concepts which are composed of more than one word.

We adopt an incremental process in order to create UREM, we start with concepts of category one. Next, we use results of category one to complete the unification process with the concepts of the second category and conclude UREM. Knowing that it was possible to deal with the two categories in the same way as described in section 3.3 for category two by finding similarities between texts definitions. But regarding category one, there exists a simpler way to group similar concepts in one abstract concept. In addition, WordNet guides us to find appropriate names for abstract concepts (hypernyms) of category one. We keep using these names resulting from category one in our meta-model as extensions (regarding meaning) of WordNet concepts (words). The choice of an incremental process simplifies the unification process.

The following sub-sections describe how to deal with each category of concepts to conclude new abstract concepts of UREM.

3.1 Tokenization

The first step of the unification process is to set up initial data to be treated as follows:

List of tokens T which is a list of all concepts names of the three approaches i*, CREWS and PREview.

$$T = \{\text{Actor, Task, Actor, Goal, SoftGoal, UseCase, Scenario, Agent, Object, Action, Event, State, Name, StructureObject, StateTransition, Viewpoint, Concern, Requirement, Source, History, Focus}\} \quad (2)$$

From the definition of each concept of category two, we create a list of tokens (words) that are composing concepts definitions by removing stop words (is, a, the…). There are four concepts of category two.

$$T' = \{T_1, T_2, T_3, T_4\} \quad (3)$$

Where T_1, T_2, T_3, T_4 are respectively tokens lists for SoftGoal, UseCase, StructureObject and StateTransition.

These lists will be used to find the appropriate sense for each token in a list when the overall tokens in the list are used together. The following sub-sections describe the remaining steps of the process.

3.2 Dealing with Concepts of Category One

The algorithm of abstraction of this category of concepts is composed of two steps.

3.2.1 Semantic Relatedness and Word Sense Disambiguation (WSD)

In English language, a word can have more than one sense that can lead to ambiguity. Disambiguation is the process of finding out the most appropriate sense of a word (concept) that is used in a given context.

The Lesk algorithm (Lesk, 1986) uses dictionary definitions (gloss) to disambiguate a polysemous word in a sentence context. The idea of the algorithm is to count the number of words that are shared between the two glosses. The more overlapping (overlap scoring) the words, the more related the senses are. For example: In performing disambiguation for the "pine cone" phrasal, according to the Oxford Advanced Learner's Dictionary, the word "pine" has two senses:

- Sense 1: kind of evergreen tree with needle-shaped leaves,
- Sense 2: waste away through sorrow or illness.

The word "cone" has three senses:

- Sense 1: solid body which narrows to a point,
- Sense 2: something of this shape, whether solid or hollow,
- Sense 3: fruit of a certain evergreen tree.

By comparing each of the two gloss senses of the word "pine" with each of the three senses of the word "cone", it is found that the words "evergreen tree" occurs in one sense in each of the two words. So, these two senses are then declared to be the most appropriate senses when the words "pine" and "cone" are used together.

The original Lesk algorithm only uses the gloss of a word, and a simple overlap scoring mechanism. An adapted version of the algorithm has been proposed (Satanjeev and Ted, 2002) which uses WordNet to access a dictionary with senses arranged in a hierarchical order. This extended version uses not only the gloss/definition of the synset, but also considers the meaning of related words.

When trying to guess the appropriate sense of a word in a sentence, the original Lesk algorithm does not utilize the senses it previously assigned to the previous words. Our implementation of the algorithm takes into account senses that are already assigned.

The aim of this step is to find the appropriate sense for each concept of category one, these concepts are the tokens $t_1=c_1 \ldots c_n=t_n$ of T. Thus, to disambiguate each pair (t_i, t_j) of tokens in T. The algorithm is to find the context $Context(t_i,t_j) = (s_{ik},s_{jl})$ where s_{ik} and s_{jl} are the most appropriate senses for concepts c_i (t_i) and $c_j(t_j)$ respectively. We denote:

$$t_i = \{s_{i1},s_{i2},\ldots,s_{io}\},\ t_j=\{s_{j1},s_{j2},\ldots,s_{jp}\} \quad (4)$$

$$context(t_i,t_j) = (s_i,s_j) \quad (5)$$

$$Score(s_i,s_j) = \max(\{score(s_{ik},s_{jl})\}_{1\le k\le o,\ 1\le l\le p}) \quad (6)$$

The algorithm is described in the following steps:

1. For each pair of tokens t_i and t_j in the list, we look up and list all possible senses $\{s_{i1},s_{i2},\ldots,s_{io}\},\{s_{j1},s_{j2},\ldots,s_{jp}\}$
2. For each sense s_{ik} of the two tokens we list the three following relations $def(s_{ik})$:
   a. The definition of synset which represents synonyms $def_{syno}(s_{ik})$
   b. The definition of synset which represents the hypernym. $def_{hype}(s_{ik})$
   c. Definitions of synsets that represent hyponyms. $def_{hypo}(s_{ik})$
3. Combine all possible gloss pairs that are archived in the previous steps, and compute the relatedness by searching for overlap. The overall score is the sum of the scores for each relation pair. We denote:

$$Def(s_{ik}) = \{Def_{syno}(s_{ik}) = x_1,\ Def_{hype}(s_{ik}) = x_2,\ Def_{hypo}(s_{ik}) = x_3\} \quad (7)$$

$$Def(s_{jl}) = \{\ Def_{syno}(s_{jl}) = y_1,\ Def_{hype}(s_{jl}) = y_2,\ Def_{hypo}(s_{jl}) = y_3\ \} \quad (8)$$

$$Score(s_{ik},s_{jl}) = \sum_{1<s<4,1<t<4} Overlap(x_s,y_t) \quad (9)$$

Where $Overlap(x_s\ y_t)$ is a function which counts the number of shared words between the two definitions x_s and y_t.

4. Once each combination has been scored, we pick up the sense that has the highest score to be the most appropriate sense for the target concept in the selected context (T). We denote:

$$Score(s_i,s_j) = \max(\{score(s_{ik},s_{jl})\}_{1\le k\le o,\ 1\le l\le p}) \quad (10)$$

3.2.2 Least Common Hypernym and Semantic Similarity between Two Senses

The above method allows us to find the most appropriate sense for each concept of category one in T. In this step we look up the least common hypernym (LCH) for each pair of this category of concepts using appropriate senses. We treat the taxonomy of hyponymy as a tree. The following steps describe how to get the overall tree which is composed of a set of LCH of different concepts of category one:

1. For each sense s_i, we build the tree $Tree(s_i)$ from the node s_i to all hypernyms of all levels. A $Tree(s_i)$ is defined as follows:

$$Tree(s_i) = Level(Hype(s_i)) \quad (11)$$

$$Hype(s_i) = \{h^1_i, h^2_i,\ldots, h^n_i\} \quad (12)$$

$$Level(x^l_i) = Level(x^{l-1}_i) + 1 \mid x^l_i \in Hype(s_i) \quad (13)$$

$$Level(s_i) = 1 \quad (14)$$

Where $Hype(s_i)$ is the set of nodes which represent hypernyms of different levels. Level is a function which associates a level value for each node in the tree.

2. Once all trees are built. We establish connections between LCH. The least common hypernym between two senses is the common hypernym which have the minimum value of level between all other common hypernyms.
3. We build a new tree Tree(T) which is composed of the set of concepts of the first category and all least common hypernym archived in the previous step. Figure 1 illustrates this tree.

For example, Work is a hypernym of the two concepts Task and Action. Task and Action are kinds of Work and Work is an abstraction of the two concepts. Any abstraction between two concepts means that exist some kind of similarity between them. P. Resnik (1999) said: "The shorter the path from one node to another, the more similar they are". We use this idea to compute similarity scores in sub-section 3.3 (The greater value of similarity score between two concepts, the shorter path it is).

Least common hypernyms LCH_s will be used as concepts of the UREM. This step gives us a first look to choose appropriate names for new abstract concepts.

$$LCH_s = \{Work, Content, Event, Knowledge, PsychologicalFeature, Quality, AbstractEntity, PhysicalEntity, Entity\} \quad (15)$$

We observe that State, Name and Source have not common hypernyms and are not present in the tree illustrated in Figure 1. We move these concepts to category two to deal with them in another way. We modify the set T' by adding three tokens lists T_5, T_6 and T_7 for State, Name and Source.

$$T' = \{T_1, T_2, T_3, T_4, T_5, T_6, T_7\} \quad (16)$$

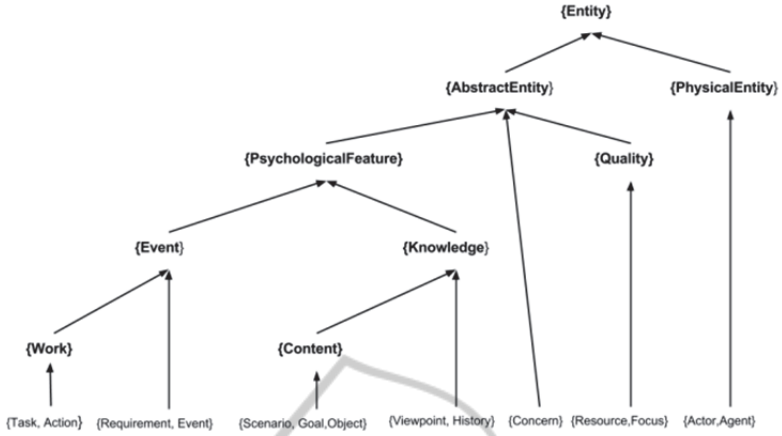

Figure 1: Tree of least common hypernyms.

3.3 Dealing with Concepts of Category Two

In the previous section 3.2 we have concluded the set of our abstract concepts for UREM. We are aware that where exist a common hypernym between two concepts, there exist a path between them. The shorter path from the first concept to the second, the more similar they are. For example: Task is similar to Requirement because they share Event as hypernym but task is more similar to Work because the distance between the two nodes is shorter. So, the similarity score is proportional to the path-length. There are many proposals for measuring semantic similarity between two synsets (synsets are concepts in our case): Wu and Palmer (1994), Leacock and Chodorow (1998), Resnik (1995b). In this paper, we experiment with a simple formula.

$$Sim(s_1,s_2) = 1 / distance(s_1,s_2) \quad (17)$$

In other words, for concepts of category one, we have found a distance between two concepts through a common hypernym and from this distance we can compute similarity score. So, if there exists a similarity score between two concepts, we can create a path between them through a common hypernym (as an extension of this hypernym in WordNet). In this step, we compute similarity scores between concepts of category two by comparing text definitions for each pair of them with LCH_s elements. Any *synset* in WordNet is substitutable of its *hypernym* then if we say Work for example, it means Work, Task, Action...and so on (Task is a Work). Each definition is composed of a set of words.

The following steps describe the algorithm:

1. For each list of tokens T_i (definition of a concept in category two) in T', we disambiguate each token and find its most appropriate sense as described for concepts of the first category.
2. For each pair of concepts definition (tokens lists) T_k and T_m in T', we build similarity relative matrix of appropriate senses R[n,m] where n is the number of tokens of T_k and m is the number of tokens of T_m, and the cell R[i,j] is the similarity score between the appropriate sense for the token of T_k at position i and the appropriate sense for the token of T_m at position j. we compute the similarity score between senses by finding the distance between the two senses as described for category one concepts, and applying the formula 17.
3. For each concept definition T_i in T' with each concept of the first category. We build similarity relative vector R[n,1] in the same way of the previous step 2.
4. For each matrix archived in the previous steps 2 and 3 we compute the overall similarity score between the two concepts concerned. There are many strategies to acquire an overall combined similarity value for sets of matching pairs, we apply an appropriate strategy (Average Matching) to compute the overall score. This similarity is computed by dividing the sum of similarity values of all match candidates (senses) of both concepts by the total number of set tokens. An important point is that it is based on each of the individual similarity values, so that the overall similarity always reflects the influence of them.

$$OverallSim(x,y) = 2*Match(x,y) / |x|+|y| \quad (18)$$

Once we have all similarity scores values. We can build the total similarity matrix for concepts of category two and abstract concepts that are resulted

from the previous step (sub-section 3.2).

3.3.1 Clustering

In this step, we classify our concepts in a set of classes, each class or in other words each cluster groups a set of similar concepts. There exist many strategies and methods to perform clustering from a similarity matrix. We use hierarchical agglomerative clustering (HAC) (Hastie et al., 2009) which seeks to build a hierarchy of clusters. This method starts by considering that each concept is a cluster, and step by step, pairs of clusters are merged as one moves up the hierarchy.

The results of hierarchical clustering are usually presented in a dendrogram. Figure 2 illustrates the dendrogram of concepts clustering.

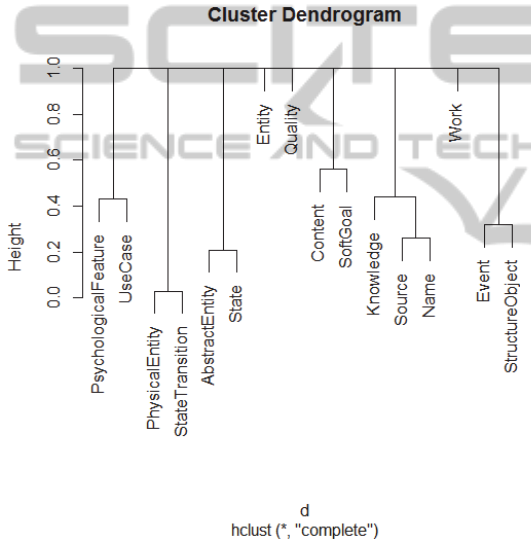

Figure 2: Cluster Dendrogram of concepts.

By cutting the dendrogram at the top level we get the followings clusters:

C_1 = {PsychologicalFeature, UseCase}

C_2 = {PhysicalEntity, StateTransition}

C_3 = {AbstractEntity, State}

C_4 = {Entity}

C_5 = {Quality}

C_6 = {Content, SoftGoal}

C_7 = {Knowledge, Source, Name}

C_8 = {Work}

C_9 = {Event, StructureObject}

In the next section we draw the unified requirements engineering Meta-model (UREM) from these clusters.

4 UREM CONSTRUCTION

In this section we conclude UREM Meta-model. Clusters from C_1 to C_9 are concepts that are archived in 3.2.2 by adding concepts of category two: UseCase, StateTransition, State, SoftGoal, Source and Name, StructureObject to PsychologicalFeature, PhysicalEntity, AbstractEntity, Content, Knowledge and Event respectively.

We keep using same names of abstract concepts of category one for our meta-model. Each concept will be an extension of WordNet concept. The concept Event in UREM for example covers the concept event as described in WordNet in addition to the concept StructureObject as described in CREWS approach. We denote:

$$Event_{UREM} = Event_{WordNet} \cup StructureObject_{CREWS}$$

Relationships between the abstract concepts obtained are Hyponymy relationship or is-kind of relationship. This relationship is modelled as *Generalization* relationship in UML. We add an abstract class *Concept* at the top of all these concepts to add a specific meaning to our context that all these concepts are concepts in Requirements Engineering.

Figure 3 illustrates the Unified Requirements Engineering Meta-model (UREM). A list of concepts is written near their abstract class.

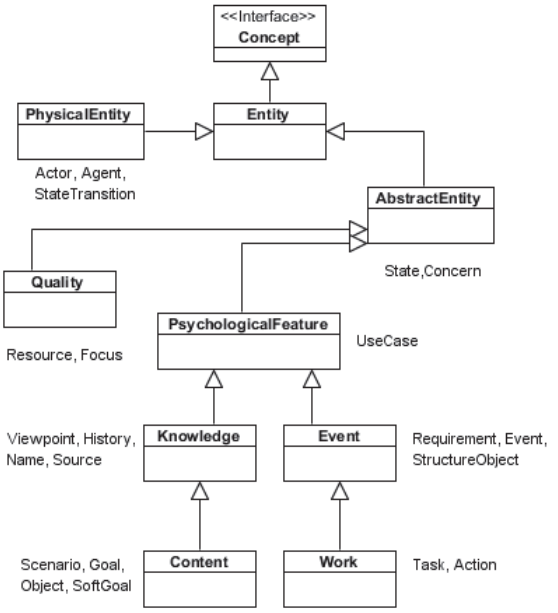

Figure 3: The Unified Meta-Model (UREM).

The goal of UREM is to ensure interoperability and information translation between three different types of RE meta-models. For example, an Action in a CREWS model will be translated into a Task in an i* model via the concept Work of UREM.

5 IMPLEMENTATION

The process described in the previous sections was implemented using C# 4.5 with Visual Studio 2012. Regarding WordNet, Bernard Bou has developed a MySql database that unifies many versions and extensions of WordNet: WordNet 3.0, WordNet 2.0-2.1, 2.1-3.0, 2.0-3.0 sensemaps, VerbNet 2.3, XWordNet 1.1.

Unfortunately, the algorithm takes too long to execute. For example: it takes more than 10 minutes to disambiguate and build trees for all concepts of category one. The implementation needs optimization and code cleaning.

Figure 4 shows some non-functional issues in the algorithm code.

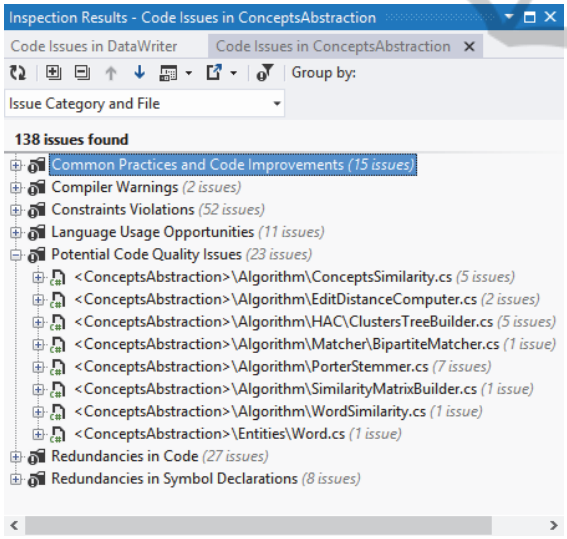

Figure 4: Inspection of Code Issues in the algorithm.

Hierarchical clustering is performed using R programming language and its library igraph. Igraph includes implementation to perform clustering, generate dendrogram, graphs and so on. Figure 2 shows a dendrogram using this library.

6 CONCLUSIONS

This paper has presented a semantic process using WordNet to merge and group similar concepts of different requirements engineering approaches into abstract concepts. These abstract concepts compose the new requirements engineering meta-model (UREM). For a given concept, WordNet allows us to find its abstract concepts (hypernyms). Unfortunately, these concepts are not all simultaneously presented in WordNet. We have proposed an incremental process, we have started the unification process by abstracting concepts that are presents in WordNet (category one). This gives us a first look about abstract concepts naming. By computing semantic similarities in category two, we have merged remaining concepts to abstract concepts archived in the first step. By keeping the same names that are used in the first step, the resulted clusters in the second step present an extension meaning of abstract concepts of WordNet.

UREM will be used as translator of concepts between different RE meta-models in order to allow cooperation between companies which use different types of RE approaches. As a next step, we plan to implement a translation rules between concepts using UREM. We are looking also to implement an interactive tool to enrich requirements visualization and communication between different types of approaches.

ACKNOWLEDGEMENTS

This work is supported in part by the project PHC CMEP Tassili n° 10MDU817.

REFERENCES

Bendjenna, H., Zarour, N. E., Charrel, P. J., 2010. Eliciting Requirements for an inter-company cooperative information System. *Journal of Systems and Information Technology (JSIT)*, Vol 12 n°4, Emerald Group Publishing, pp. 305-333.

Carlos, C., Xavier, F., 2011. A Metamodelling Approach for i* Model Translations. *23rd International Conference, CAiSE 2011*. London, UK, June 20-24. Proceedings pp 337-351.

Castro, J., 2011. Goal Oriented Requirements Engineering i*, Fifth International Conference on Research Challenges in Information Science.

George, A. M., 1995. WordNet: A Lexical Database for English, *Communications of the ACM.* VOL 38, PAGE 39-41.

Hastie, T., Tibshirani, R., Friedman, J., 2009. *Hierarchical clustering (PDF). The Elements of Statistical Learning (2nd ed.).* New York: Springer. pp. 520–528. ISBN 0-

387-84857-6. Retrieved 2009-10-20.

Leacock, C., Chodorow, M., 1998. Combining Local Context and WordNet Similarity for Word Sense Identification, *chapter 11, pages 265–283. MIT Press, Cambridge, MA.*

Lesk, M., 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. *In SIGDOC '86: Proceedings of the 5th annual international conference on Systems documentation, pages 24-26,* New York, NY, USA. ACM.

Resnik, P., 1999. Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language, *Journal of Articial Intelligence Research 11 95-130.*

Resnik, P., 1995b. Using information Content to evaluate semantic similarity in a taxonomy. *In Chris Mellish, editor, IJCAI-95, pages 448–453*, Montreal, Canada.

Saidi, I.E., Dkaki, T., Zarour, N.E., Charrel, P.J., 2012. Towards Unifying Existing Requirements Engineering Approaches into a Unified Model. *KMIS 2012: 311-315*.

Satanjeev B.,Ted P., 2002. An Adapted Lesk Algorithm for Word Sense Disambiguation Using WordNet, *Computational Linguistics and Intelligent Text Processing Lecture Notes in Computer Science Volume 2276, 2002, pp 136-14*.

Sommerville, I., Sawyer, P., 1997a. Requirements Engineering: A Good Practice Guide. *John Wiley & Sons, Inc. New York, NY, USA*. ISBN:0471974447.

Sommerville, I., Sawyer, P., 1997. Viewpoints: principles, problems and a practical approach to requirements engineering. *Computing Department, Lancaster University, Lancaster LA1 4YR, UK,* Annals of Software Engineering 3.

Sutcliffe, A. G, Maiden N., Shailey, M., Darrel, M., 1998. Supporting Scenario-Based Requirements Engineering, *IEEE Transactions on software engineering,* VOL. 24, NO. 12.

Vincent, D. B., Anahi, G., Maureen, H., Pierre, S., Paul, V. D. 2004. A Measure of Similarity between Graph Vertices: Applications to Synonym Extraction and Web Searching. *Society for Industrial and Applied Mathematics.*

Yu, E., 1995. Modelling strategic Relationships for Process Reengineering. *PhD thesis, university of Toronto Canada*.

Wu, Z., Palmer M., 1994. Verb semantics and lexical selection. In *32nd Annual Meeting of the Association for Computational Linguistics*, pages 133–138.