The Synonym Management Process in SAREL

Àngels Hernández & Núria Castell
TALP Research Center
Universitat Politècnica de Catalunya
Barcelona, Spain 08034
e-mail: (ahernandez, castell)@talp.upc.es

Abstract

The specification phase is one of the most important and least supported parts of the software development process. The SAREL system has been conceived as a knowledge-based tool to improve the specification phase. The purpose of SAREL (Assistance System for Writing Software Specifications in Natural Language) is to assist engineers in the creation of software specifications written in Natural Language (NL). These documents are divided into several parts. We can distinguish the Introduction and the Overall Description as parts that should be used in the Knowledge Base construction. The information contained in the Specific Requirements Section corresponds to the information represented in the Requirements Base. In order to obtain high-quality software requirements specification the writing norms that define the linguistic restrictions required and the software engineering constraints related to the quality factors have been taken into account. One of the controls performed is the lexical analysis that verifies the words belong to the application domain lexicon which consists of the Required and the Extended lexicon. In this sense a synonym management process is needed in order to get a quality software specification. The aim of this paper is to present the synonym management process performed during the Knowledge Base construction. Such process makes use of the Spanish Wordnet developed inside the Eurowordnet project. This process generates both the Required lexicon and the Extended lexicon that will be used during the Requirements Base construction.

1. Introduction

The preliminary Software Specifications in complex systems are often written in Natural Language. The documentation writing is guided by the norms that define the linguistic restrictions required and also by the software engineering constraints related to the quality factors of the software specifications. However, in general, these norms are not strictly followed. It is very important to control the writing of these Software Requirements Specifications (SRS) in order to detect conceptual mistakes in this first step of the software development process. The correction of errors in the design and implementation phases implies spending more time and effort than in the specification phase. The ambiguity of the documents and the fact of not following the norms can produce an incorrect formal specification. A complete study of the problems arising from NL specifications can be found in [12].

Many projects have tackled the use of Natural Language in the specification phase. Among the CREWS project publications we want to remark [2] where the scenario based approach and linguistic based instrument have been proposed for improving requirement engineering tools and techniques. Within the framework of quality documents the ATTEMPTO approach [9] should be pointed out, whose main goal is to reduce ambiguity and vagueness inherent in NL. The REVERE project [14] should also be mentioned because it integrates a number of techniques to provide a set of tools to help requirements engineer to explore the documentation. The goal in our case is different because we want to help engineer during the writing process. Another important work is QuARS [8] where a tool for the analysis of natural language software requirements based on a quality model is presented. From the lexical point of view [3] is an interesting work where a lexical analysis can be used to identify individual objects which will translate directly into the final implementation.

Within the software specification written in NL area we have designed the SAREL help-system in order to obtain high-quality software requirements specifications. This is a knowledge-based system, where some linguistic engineering tools are applied. In this paper we will focus in some lexical tasks related with the ambiguity reduction and the Knowledge Base construction.

The paper is organized as follows: in section 2, a description of the SAREL system is given. Section 3 explains the lexicon generation process in detail to obtain the Required Lexicon and the Extended Lexicon used in later phases. In section 4 we show how these lexicons are used in order to rewrite the Introduction and the Overall Description without synonyms. The different rules used in Knowledge Base generation process are explained in section 5. Then the Modules used during the Requirements Base construction are described in section 6. Finally, in section 7, the conclusions are presented.

2. Description of the SAREL System.

The main goal of SAREL is to assist a software engineer in the creation of quality preliminary software specifications written in NL. The assistance process validates the SRS introduced by the engineer taking into account the writing norms (for instance [11]) and the quality properties [5]: consistency, completeness, traceability, modifiability, and verifiability (among others stated by IEEE Standards). As a result, a Conceptual Representation which consists of the Knowledge Base (KB) and the Requirements Base (RB) of the software requirement specification is obtained. It should be emphasized that the KB creation process is the first step. This process is shown on the left side on the Figure 1. After that the system will be ready to check the software requirements set. Figure 1 shows that on the right side.

Taking into account [11], there are three essential parts in a SRS: (1) **Introduction** provides an overview of the entire SRS; (2) the **Overall Description** section describes the general factors that affect the product and its requirements; (3) the **Specific Requirements** section contains all the software requirements, going into enough detail so as to enable engineers to design a system that satisfies those requirements.

We can distinguish the Introduction and the Overall Description as parts that should be used in the Knowledge Base construction. Up to a certain point, these two parts contain all the background information needed to understand the problem as a whole. The information contained in the specific Requirements Section corresponds to the information represented in the Requirements Base.

The KB construction process is split into three steps: (1) the ***Lexicon Generator*** extracts from the original text the Required Lexicon and the Extended Lexicon using Wordnet [W]; (2) the ***Lexical Refinement*** will generate a new Introduction and a new Overall Description where all the words belong to the Required Lexicon; (3) the ***KB Generator*** will generate a hierarchy of concepts grouped into two main classes: Objects and Activities.

At this point we can see that the lexicon generation process is important because we can control which words must be used (Required) which words can be substituted by others (Extended) and which words can not be used.

The RB construction process is split into three Modules: (1) the ***Style Refinement Module*** controls the requirement introduced according to the writing norms, the lexical analysis will use the Required and the Extended Lexicon to verify that the words belong to the application domain; (2) the ***Conceptual Refinement Module*** validates the conceptual requirement in relation to the KB and the RB; (3) the ***Software Quality Control Module*** carries out a set of optional analyses to validate the global RB increased with the new requirement.

Finally if the Requirement Conceptual Representation is correct it will be added to the RB.

3. The Lexicon Generator.

In this section we present the process which generates the Required lexicon and the Extended lexicon. The Extended lexicon contains all the words of the application domain and the Required is a subset of the Extended. Each word present in the Required is the representative that has been chosen from the synonyms set.

Firstly, the morphological analyzer Maco+ [1] and the POS tagger Relax [13] will process the sentences contained in the Introduction and the Overall Description of the document. The output of this process is a list of words with its corresponding PAROLE tag.

Secondly from the list of words obtained the system will split the list into three different sub lists corresponding with the names, verbs and adjectives. Each of them will be processed by the Synonym Analysis in order to obtain the Required and Extended names, the Required and Extended verbs and the Required and Extended adjectives. Finally all these three results will be put it all together into the Required Lexicon and Extended Lexicon.

3.1. Synonym Analysis

For each word in the text analyzed the system will find all the synsets using Wordnet and its corresponding label. The synsets associated to a word are the different meanings that has that word in different contexts. For example the word *light* has different meaning depending on the context: "*light up*" or "*light drink*". The result of this search is as follows:

$label_{1\,1}$ $word_1$ $synset_1$
$label_{1\,2}$ $word_1$ $synset_2$
....
$label_{1\,N}$ $word_1$ $synset_N$

After that for all synsets the system will find all the synonyms associated and its corresponding synset:

$label_{1\,1}$ ($word_1$ $synset_1$ $synonym_{1\,1}$ $synset_{1\,1}$
.....
$synonym_{1\,M1}$ $synset_{1\,M1}$)
$label_{1\,2}$ ($word_1$ $synset_2$ $synonym_{2\,1}$ $synset_{2\,1}$
.....
$synonym_{2\,M2}$ $synset_{2\,M2}$)
.....
$label_{1\,N}$ ($word_1$ $synset_N$ $synonym_{N\,1}$ $synset_{N\,1}$
.....
$synonym_{N\,MN}$ $synset_{N\,MN}$)

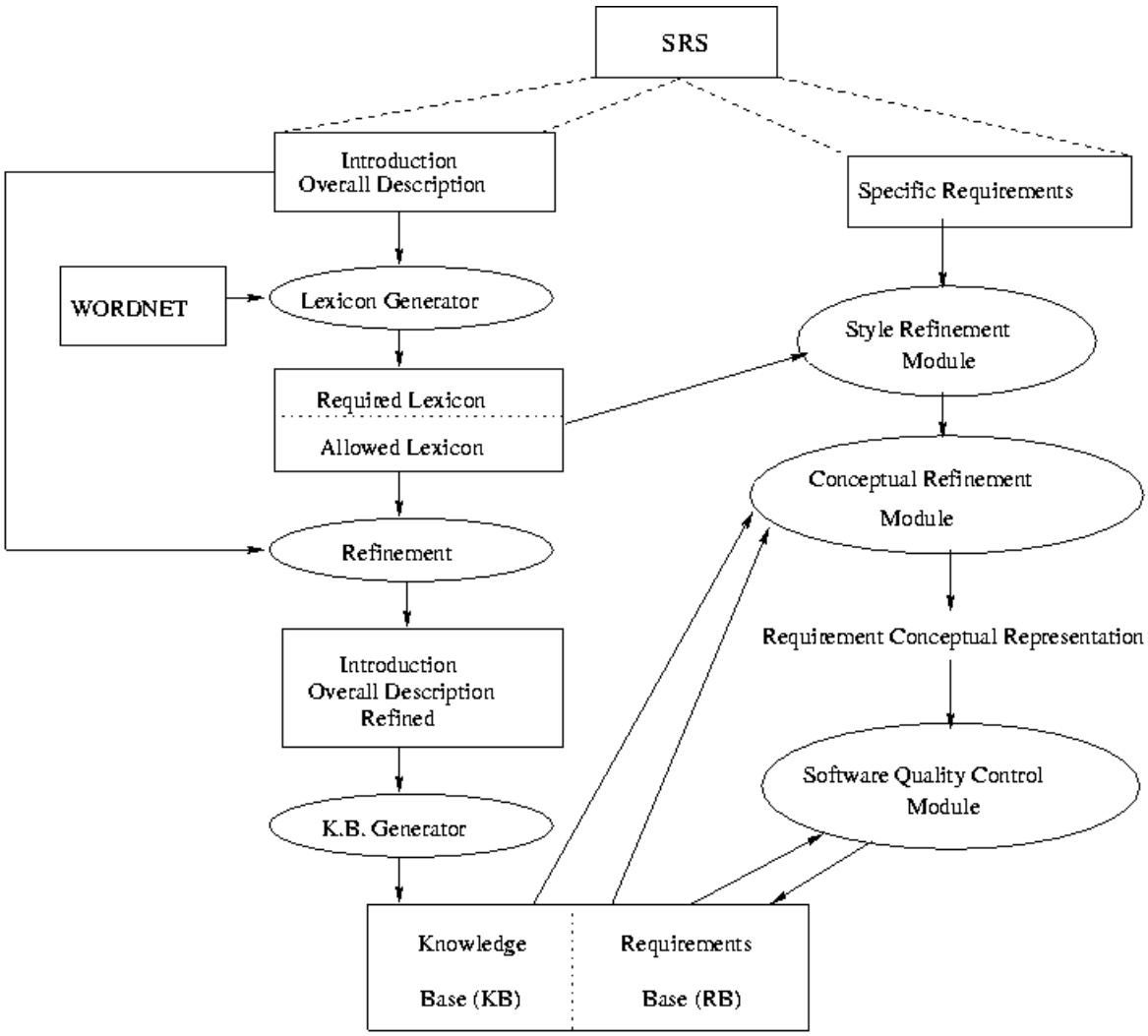

Figure 1: Knowledge Base and Requirements Base Construction

Following the system will find the possible coincidences between the first word that belongs to the original text and the rest of the sets. A coincidence appears when exists the following situation:

$label_{X\,J}$ ($word_X$ $synset_J$ ..... $synonym_{J\,K}$ $synset_{J\,K}$ .....)

$label_{YQ}$ ($word_Y$ $synset_Q$..... $synonym_{Q\,R}$ $synset_{Q\,R}$ .....)

$word_X = synonym_{Q\,R}$

$word_Y = synonym_{J\,K}$

$label_{X\,J} = label_{Y\,Q}$

There are five possible situations:

- (1) The $word_X$ does not appear in any of the others sets, and it has many synsets associated. In this case we do not know the right label, neither the right synset. The information to save is only the $word_X$, so that the output will be:

label? $word_X$ synset? [ ]

- (2) The same situation as before but in this case the $word_X$ has only one synset, then we will keep this information:

$label_{X\,1}$ $word_X$ $synset_1$ [$synonym_{1\,1}$ $synset_{1\,1}$

.....

$synonym_{1\,M1}$ $synset_{1\,M1}$]

- (3) The $word_X$ appears only once in the rest of the sets. That means the coincidence has selected the right synset.

$label_{X\,J}$ ($word_X$ $synset_J$ ..... $synonym_{J\,K}$ $synset_{J\,K}$ .....)

$label_{YQ}$ ($word_Y$ $synset_Q$..... $synonym_{Q\,R}$ $synset_{Q\,R}$ .....)

$word_X = synonym_{Q\,R}$

$word_Y = synonym_{J\,K}$

$label_{X\,J} = label_{Y\,Q}$

At this point it is necessary to decide which word will be the representative of the synonym set. To do that the analyzer will consider the frequency of each word in the original text. If the $word_Y$ has a higher frequency than the $word_X$ the information to hold is:

$label_{YQ}$ $word_Y$ $synset_Q$ [.....$synonym_{Q\,R}$ $synset_{Q\,R}$ .....]

- (4) The $word_X$ comes into the rest of the sets many times, but all the coincidences correspond with the same word in different synsets. That means, there are little differences between the synsets. The system will save all these synsets because all of them are correct.

$label_{YQ}$ $word_Y$ $synset_Q$ [.....$synonym_{Q\,R}$ $synset_{Q\,R}$ .....]

$label_{YP}$ $word_Y$ $synset_P$ [.....$synonym_{P\,R}$ $synset_{P\,R}$ .....]

(5) The last possible situation is the same as the previous one, but now the coincidences correspond with different words. In this situation the Synonym Analysis will select the word with more coincidences with the $word_X$ leading us to the same situation 4 .

4. The Lexical Refinement.

Once the Required Lexicon and the Extended Lexicon have been built the following step is to refine the Introduction and the Overall Description in order to get a text without synonyms. The output is a text where all the names, verbs and adjectives belong to the Required Lexicon. To get that each word will be analyzed:

- if the word belongs to the Required Lexicon it will be added to the text refined.
- if the word belongs to the Extended Lexicon the Lexical Refinement process will substitute this word by the representative of the synonym set. The Morphological Analyzer (maco+) is used in order to obtain the right conjugation. For example if we have to substitute the verb "get" by the verb "obtain" in the following sentence:

"The system A is getting the data"

the result will be:

"The system A is obtaining the data"

Firstly the process transform *getting* to *get*. Secondly the system will substitute *get* by *obtain*. And finally the word *obtain* will be transformed into *obtaining* using the PAROLE tag.

At this point we want to emphasize the input resulted in a refined text (Introduction and Overall Description) without synonyms. The Lexical Refinement is an essential process taking into account this text will be the starting point to the creation of the Knowledge Base.

5. The KB Generator.

The KB Generator starts from a list of words with its corresponding PAROLE tag generated in the previous steps and the goal here is to generate a hierarchy of concepts grouped into two main classes: Objects and Activities. As a consequence of the Lexical Refinement the KB Generator will not produce two synonyms concepts. This process is split into three steps:

1. Creation of the main nodes: Object and Activity.
2. Object nodes generation:

2.1. Creation of nodes corresponding to simple names tagged as NC (Common Nouns)

2.2. Creation of nodes corresponding to noun phrases tagged as: {NC followed by NC} or {NC followed by AQ (Qualifying)} or {NC followed by VMP (Participle)}.

2.3. Creation of nodes corresponding to the following schema: A + De (Of) + B, where A and B are object nodes generated before.

3. Activity nodes generation:

3.1 Creation of nodes corresponding to simple verbs tagged as VMI (intransitive verb) or VMN (infinitive).

3.2 Creation of nodes corresponding to verbal groups tagged as VMI followed by VMN.

After the KB construction process the software engineer can visualize the Conceptual Representation obtained.

6. The RB Construction.

As set out above, the information represented in the RB corresponds to the information contained in the specific requirements section. In order to control the set of software requirements contained in the SRS document, we have considered necessary to establish the different roles associated to each kind of requirement. At this point we should mention the KARAT system [16] where the classification of the software requirements is the same as the one used in our system. Below we present three different kinds of requirements (among others stated by IEEE Standards [11]) and the required semantic roles set for each class. It should be pointed out that these required semantic roles could appear with other optional ones [6]:
(1) **Functional Requirements** define the fundamental actions that must take place in the software when accepting and processing the inputs and when processing and generating the outputs. The required semantic roles are:
{Agent, Action and Patient}.
(2) **Performance Requirements** specify both the static and the dynamic numerical requirements established for the software or on human interaction with the software as a whole. This class needs the following required semantic role sets:
{Patient, Measurement, At-value and Unit} or
{Patient, Measurement, From-Value, To-Value and Unit}.
(3) **Interface Requirements** correspond to a detailed description of all inputs into, and outputs from, the software system. The required semantic role sets are:
{Patient and Qualitative-Feature} or
{Patient, Quantitative-Feature and Units-Feature}.

The creation of the RB is undertaken requirement by requirement. Once the user has introduced the sentence corresponding to the requirement, the controls contained in the Style Refinement Module and the Conceptual Refinement Module are applied.

The **Style Refinement Module** controls the requirement according to the writing norms. This control is split into four steps: (1) the lexical analysis verifies that the words belong to the application domain lexicon, that means to the Required Lexicon. If not the software engineer will be consulted if new words have to be added to the Required Lexicon or not; (2) the syntactic-semantic analysis produces a tree-like semantic representation; (3) the ambiguity control helps the engineer to identify the correct representation between the possible interpretations; (4) the simplicity control detects whether the structure is simple or compound.

At that point a syntactic-semantic representation is produced. Below we present the output corresponding to the sentence :

"La ludoteca suministrará servicios de directorio"
("The play-center will supply directory services")

((La_tdfs0
ludoteca_ncfs000)_sn(suministrará_vmif3s0)_grup-verbal(servicios_ncmp000)_sn(de_sps00
directorio_ncms000)_grup-sp._Fp)_

We want to remark the substitution of the original verb in the sentence. The verb *provide* was the original one and it has been substituted by the verb *supply* during the lexical analysis.

The **Conceptual Refinement Module** validates the requirement in relation to the Requirements Base. At first, it obtains a conceptual representation using the KB and after that, it detects duplicated information. Taking into account that the requirement presented above corresponds to the Functional class, the Conceptual Refinement Module will identify "ludoteca" as the Agent, "suministrar" as the Activity and "servicios de directorio" as Patient. These entities must be present in the KB or else the system will request the software engineer to ask if new concepts have to be added or not. At this point the Conceptual Refinement Module detects duplicated information if the RB contains another representation with the same semantic roles and values. Once the requirement has been checked and is correct, its Conceptual Representation is added to the RB. This process incrementally generates a Conceptual Representation of the specific requirements section.

As in the case of the KB construction section, the Conceptual Representation associated to the requirement introduced can be visualized. In Figure 2 we present an example of Knowledge Base and Requirements Base visualization. This figure is composed of two parts: at the top, we can see the Conceptual Representation of concepts contained in the KB, and, at the bottom, the Conceptual Representation of the requirement presented above. The objects and the activities contained in the requirement correspond with nodes presents in the KB. It is also possible to visualize the representations in a global way (for example, the requirements sets containing "ludoteca" as Agent).

The **Software Quality Control Module** carries out a set of optional analyses to validate the global RB increased with the new requirement. The goal is to offer information about the software quality properties: consistency, completeness, traceability, modifiability, and verifiability.

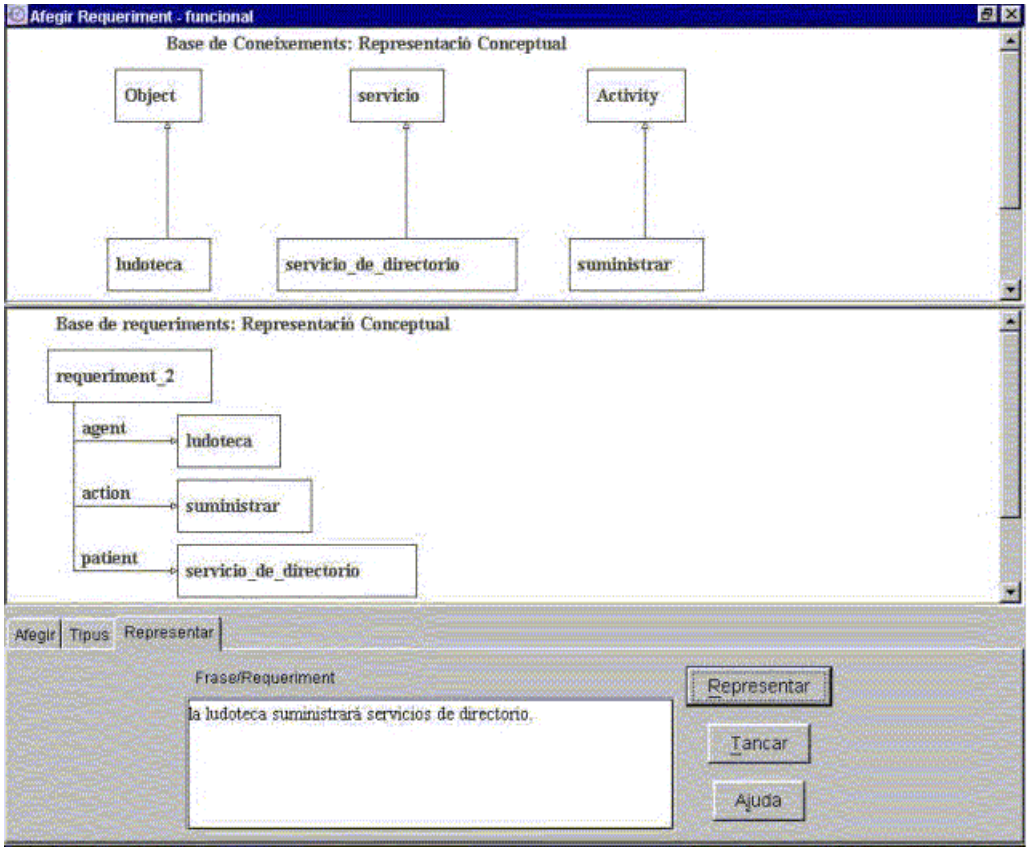

Figure 2: KB and RB Visualization

6.1. The two functionalities of SAREL

The SAREL system has two different functionalities, depending on the user's goal: Vertical Processing and Horizontal Processing.

Vertical Processing basically corresponds to the sequential application of the controls. The input is a software specification written in NL and the output is its associated Conceptual Representation. The Requirements Section is processed, requirement after requirement, by both the Style Refinement Module and the Conceptual Refinement Module in order to obtain the Conceptual Representation that can be optionally validated by the engineer using the Software Quality Control Module. Once a requirement has been checked, its conceptual representation is added to the RB. Using this functionality, it is possible to obtain the Conceptual Representation associated with the preliminary software specification. This means that the information represented can be consulted in a collective or individual way by the engineers in a more reliable format. A more precise description of this functionality can be found in [4].

In **Horizontal Processing** the input is of two different conceptual representations, and the goal here is to offer information about the correspondence between them. Document1 corresponds to the User Company that needs to develop a computer system and, therefore, document2 corresponds to the Software Company that will carry this out. The system will give a correspondence measure based on similarity analysis [15] applied to the components of the requirements. Depending on the value of this measure, the correspondence will be tagged as: Correct, Excess or Excess-Insufficient. See [7] for a more precise description.

7. Conclusions

The main goals of the SAREL system are:

- To obtain a high-quality software requirements specifications. To reach that the writing norms that define the linguistic restrictions required and the software engineering constraints related to the quality factors have been taken into account in the Style Refinement Module, the Conceptual Refinement Module and the Software Quality Control Module.
- To get the Conceptual Representation associated to the software requirements specification. In this sense the SRS

can be manipulated and consulted in a more reliable way using the KB and RB visualization.
- To compare different conceptual representations corresponding with different documents SRS belonging to the same application domain.

All these aims need a previous control over the lexicon used in all these documents. To decrease the ambiguity level the system uses the Required Lexicon and a Extended Lexicon generated at first. In this paper we have presented the synonym management during this generation process using Wordnet.

We are planing to use many different documents (technical documentation, user manuals) as a input in the Knowledge Base generation in order to get a more complete conceptual representation. Another area to exploit in the future is in relation with the lightweight formal methods for the partial validation of natural language requirements documents presented in [10].

8. References

[1] J. Atserias, J. Carmona, I. Castellón, S. Cervell, M. Civit, L. Màrquez, M.A. Martí, L. Padró, R. Placer, H. Rodríguez, M. Taulé, and J. Turmo, "Morphosyntactic Analysis and Parsing of Unrestricted Spanish Text", First International Conference on Language Resources and Evaluation (LREC'98) Granada, Spain, 1998.

[2] C. Ben Achour, "Linguistic Instruments for the Integration of Scenarios in Requirements Engineering", Proceedings of the Third International Workshop on Requirements Engineering: Foundations of Software Quality (REFSQ'97)", Barcelona, Catalonia, Spain, June, 1997, pp. 93-106.

[3] P. Bowden, M. Hargreaves, and C.S. Langensiepen, "Estimation support by lexical analysis of requirements documents", The Journal of Systems and Software, 51 (2000), pp. 87-98.

[4] N. Castell, and A. Hernández, "Filtering Software Specifications Written in Natural Language", Proceedings of the 7th Portuguese Conference on Artificial Intelligence (EPIA'95), LNAI 990, Funchal, Madeira Island, Portugal, 1995, pp. 447-455,

[5] N. Castell, O. Slavkova, Y. Toussaint, and A. Tuells, "Quality Control of Software Specifications written in Natural Language", Proceedings of the 7th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE'94), Austin, Texas, USA, 1994, pp. 37-44.

[6] N. Castell, and A. Hernández, "The Software Requirements Modeling in SAREL", Proceedings of the 4th International Workshop on Requirements Engineering: Foundations of Software Quality (REFSQ'98), Pisa, Italy, 1998, pp. 49-56.

[7] N. Castell, and A. Hernández, "The use of SAREL to control the correspondence between Specification Documents", Proceedings of VII Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA'97), Málaga, Spain, 1997, pp. 529-539.

[8] F. Fabbrini, M. Fusani, S. Gnesi, and G. Lami, "An Automatic Quality Evaluation for Natural Language Requirements", Proceedings of the Seventh International Workshop on Requirements Engineering: Foundations of Software Quality (REFSQ'01), June 4-5, 2001, Interlaken, Switzerland 2001.

[9] E. Fuchs, and R. Schwitter, "Attempto Controlled English (ACE)", First International Workshop On Controlled Language Applications. Katholieke Universiteit Leuven, Belgium 1996.

[10] V. Gervasi and B. Nuseibeh, "Lightweight Validation of natural language Requirements: a case study", Proceedings of the Fourth International Conference on Requirements Engineering (ICRE'2000), June 19-23, Schaumburg, Illinois, 2000, pp.140-148.

[11] IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications, 1998.

[12] R. Melchisedech, "Investigation of Requirements Documents Written in Natural Language", Requirements Engineering (1998) 3:2, pp. 91-97.

[13] L. Padró, "A Hybrid Environment for Syntax-Semantic Tagging", PhD Thesis, Departament de Llenguatges i Sistemes Informàtics. Universitat Politécnica de Catalunya. Barcelona, 1998.

[14] P. Rayson, R. Garside, and P. Sawyer, "Recovery Legacy Requirements", Proceedings of the Fifth International workshop on Requirements Engineering: Foundations of Software Quality (REFSQ'99), June 14-15, Heidelberg, Germany 1999, pp. 49-54.

[15] H.C. Romesburg, "Cluster analysis for researchers", Belmont, Calif.: Lifetime Learning Publications, 1984.

[16] B. Tschaitschian, C. Wenzel, and I. John, "Tunning the quality of informal software requirements with KARAT". Proceedings of the Third International workshop on Requirements Engineering: Foundations of Software Quality (REFSQ'97) Barcelona, Catalonia, Spain, 1997, pp. 81-92.