

# Knowledge Management in the Sarel System

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

## Abstract

*The specification phase is one of the most important and least supported parts of the software development process. SAREL (Assistance System for Writing Software Specifications in Natural Language) has been conceived as a knowledge-based tool to improve the specification phase. The purpose of SAREL is to assist engineers in the creation of software specifications written in Natural Language (NL). The aim of this paper is to present the creation process of the Knowledge Base and the Requirements Base corresponding to a software specification written in NL. This process uses some linguistic engineering tools developed in our department for the pre-processing of the input documents. SAREL is designed to process documents that contain preliminary software specifications. 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. 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.*

## 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 [9].

An important work to be considered is the result obtained in the NATURE Project [8], whose goal is to establish a theory of Requirements Engineering. The KARAT system [11] should also be mentioned, where the classification of the software requirements is the same as the one used in our system. Further, within the framework of quality documents in general, the ATTEMPTO approach [6] should be pointed out, whose main goal is to reduce ambiguity and vagueness inherent in NL.

In order to obtain high-quality software requirements specifications, we have designed the SAREL help-system. This is a knowledge-based system, where some linguistic engineering tools are applied. The controls contained within this system can be grouped into three modules: the Style Refinement Module, the Conceptual Refinement Module and the Software Quality Control Module. The last two modules use the Knowledge Base (KB), which contains the domain knowledge, and the Requirements Base (RB), which contains the requirements representation. The functionality of the system is twofold: Vertical Processing and Horizontal Processing, depending on the user's goal.

The aim of this paper is to present the creation process of the KB and the RB corresponding to a software specification. It should be emphasized that the KB creation process is the first step. After that the system will be ready to check the software requirements set.

In section 2, a description of the modules in SAREL is given, followed by a description of the two functionalities of our system. Section 3 explains the application of the linguistic engineering tools to the documents containing software specifications. This is carried out to construct the Knowledge Base. Then, the incremental construction of the Requirements Base is explained in section 4. Some implementation details are provided in Section 5, followed by some conclusions.

## 2. Description of the SAREL System.

The main goal of SAREL is to assist an engineer in the creation of quality preliminary software specifications written in NL. The assistance process validates every requirement introduced by the engineer taking into account the writing norms (for instance [7]) and the quality properties [3]: consistency, completeness, traceability, modifiability, and verifiability (among others stated by IEEE Standards). As a result, an incremental Conceptual Representation of the software specification is obtained. The SAREL system is split into the following three modules:

- 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; (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.
- 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.
- 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 above mentioned software quality properties.

The SAREL system has two different functionalities, depending on the user's goal:

- The **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 document 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 [2].
- In **Horizontal Processing** the input is of two different conceptual representations, and the goal here is to offer information about the correspondence between them. Document<sub>1</sub> corresponds to the User Company that needs to develop a computer system and, therefore, document<sub>2</sub>

corresponds to the Software Company that will carry this out. The system will give a correspondence measure based on similarity analysis [10] 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 [5] for a more precise description.

## 3. The KB Construction.

Taking into account [7], 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.

Before introducing the different requirements that will be validated by SAREL, the KB corresponding to the software specification application domain must be built.

Firstly, the morphological analyzer (Maco+) and the POS tagger (Relax) will process the sentences contained in the Introduction and the Overall Description of the document. To a certain extent, these two parts contain all the background information needed to understand the problem as a whole. 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 generate a hierarchy of concepts grouped into two main classes: Objects and Activities. This second process is split into five steps:

1. Detection of NOMASK tagged words: It is possible to find words with no tag. In this case, these words correspond with application domain's specialized concepts and the user has to decide if these concepts have to be included in the application domain lexicon or not.
2. Creation of the main nodes: Object and Activity.
3. Object nodes generation:
  - 3.1. Creation of nodes corresponding to simple names tagged as NC (Common Nouns)
  - 3.2. 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)}.
  - 3.3. Creation of nodes corresponding to the following schema: A + De (Of) + B, where A and B are object nodes generated before.
4. Activity nodes generation:
  - 4.1 Creation of nodes corresponding to simple verbs tagged as VMI (intransitive verb) or VMN (infinitive).
  - 4.2 Creation of nodes corresponding to verbal groups tagged as VMI followed by VMN.
5. Creation of relations corresponding to the semantic roles presented in [4].

After the KB construction process the software engineer can visualize the Conceptual Representation obtained. Figure 1 presents an example of that visualization showing the nodes connected with the

object “Entidad” (Entity). The KB Conceptual Representation uses the frame-based formalism YAYA [1] developed in our department.

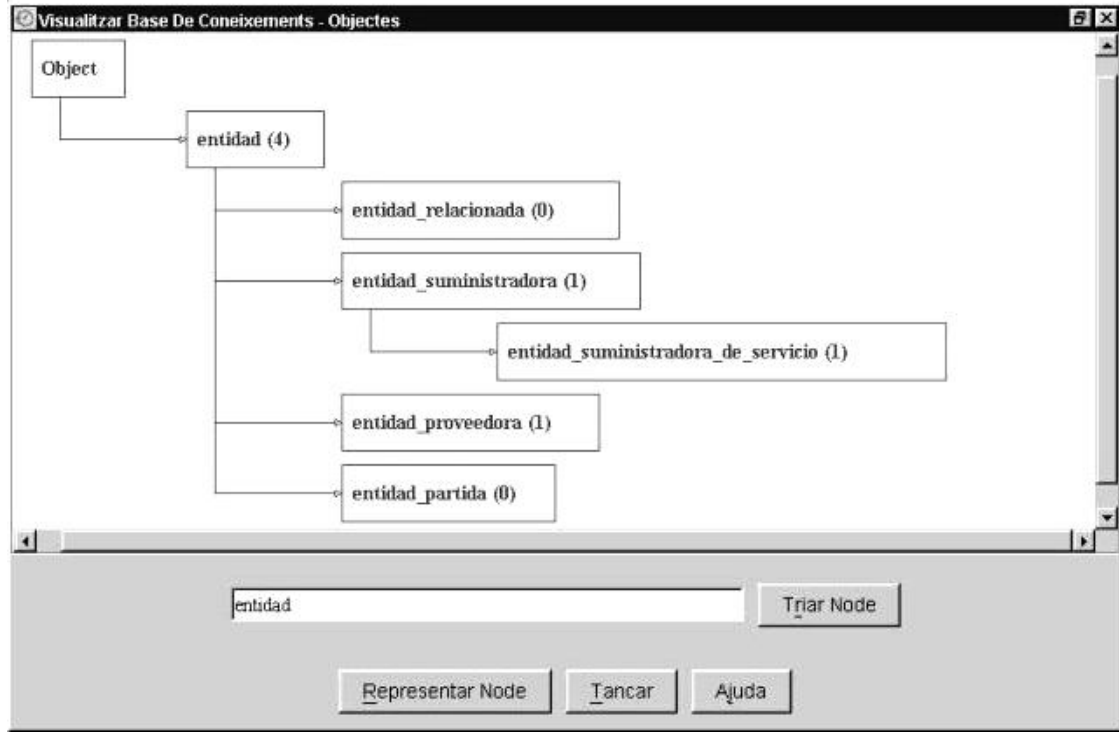


Figure 1. Knowledge Base Conceptual Representation.

#### 4. The RB Construction.

As set out above, the information represented in the Requirements Base 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. Below we present three different kinds of requirements (among others stated by IEEE Standards) 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 [4].

- **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}.

- **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}.

- **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 are applied. 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 )\_
```

Taking into account that this requirement 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 that 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 set of requirements that contain "ludoteca" as Agent).

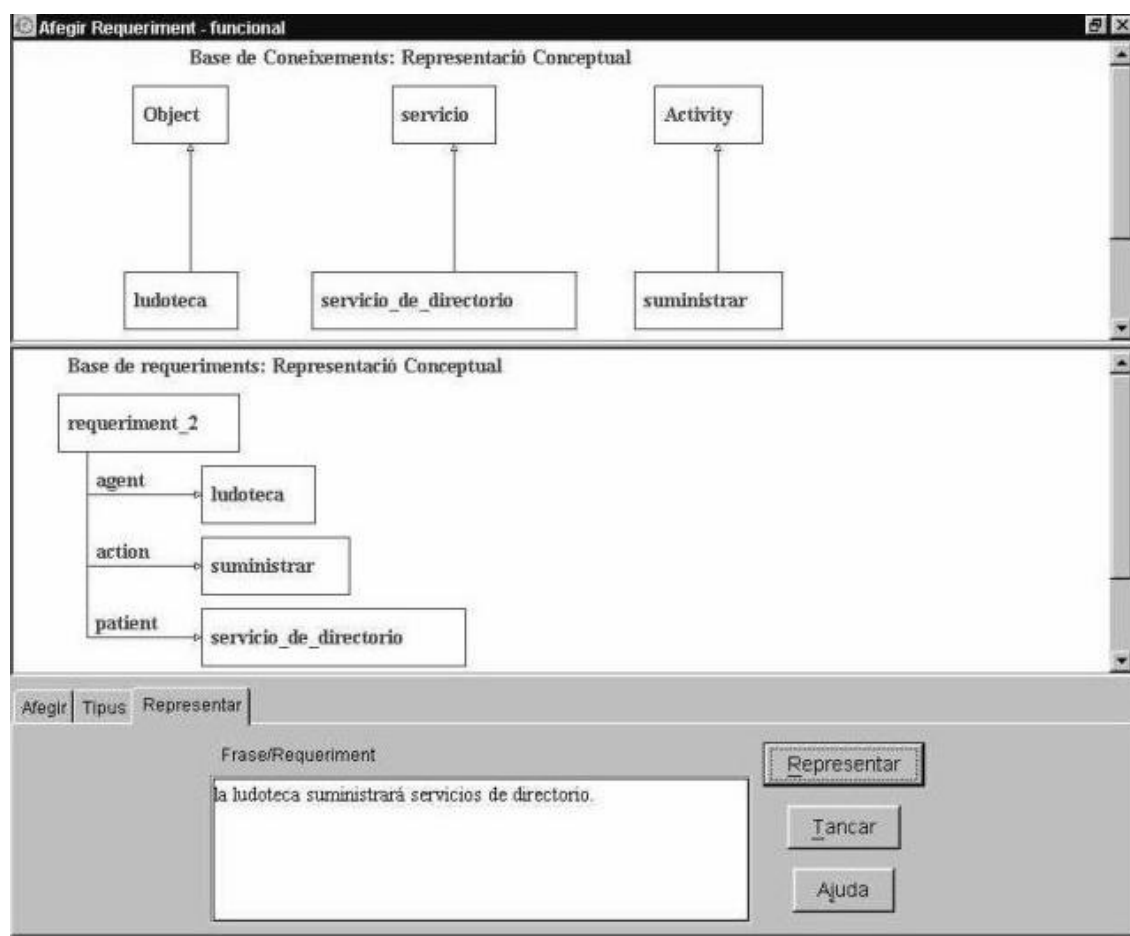


Figure 2. Requirements Base Conceptual Representation

The Conceptual Representation generated uses the frame-based formalism YAYA so that the controls contained in the Quality Control Module need to use commands of this representation system.

From the Vertical Processing point of view, the RB is used by the analysis tasks [3] contained in the Software Quality Control Module. For instance, the traceability analysis provides the engineer with information about the traceability links of one fixed

requirement. In this case, SAREL activates a set of existing algorithms that control the traceability taking into account the relationships between the introduced requirement and a subset of requirements of the RB. The relationships are based on the entities (Objects and Activities) and the relations stated in the requirements. From this information the engineer can evaluate the level of traceability of a set of requirements.

From the Horizontal Processing point of view, the correspondence analysis will search if the concepts contained in the RB1 (obtained from the User Company document) correspond with concepts contained in the RB2 (obtained from the Software Company document). During this process, it is useful to split the corresponding analysis bearing mind the different groups of requirements. The output here is a correspondence measure based on similarity analysis.

## 5. Implementation Details.

We have adapted some linguistic tools developed in our software department such as a morphological analyzer (Maco+), a POS tagger (Relax), and a syntactic chart-based parser (Tacat). It is possible to find our tools at the following web: <http://www.lsi.upc.es/~nlp>

These tools have been used: (1) in the KB construction (totally implemented); and (2) in the RB construction (partially implemented), especially in the syntactic-semantic analysis (the second step in the Style Refinement Module) and in the Conceptual Analysis (the first step in the Conceptual Refinement Module).

In order to construct the KB and the RB, we have used the frame-based formalism YAYA. The documents dealt with are written in Spanish, although the tools mentioned above can be also used for Catalan documents. The generation and visualization processes have been developed in Java 1.2.1.

## 6. Summary and Conclusions.

The main goal of this paper has been to present the creation process of the KB and the RB corresponding to a software specification written in NL. The introduction and the overall description are used in the KB construction and the specific requirements section corresponds to the information represented in the RB. The KB construction will be improved using the Spanish Wordnet developed inside the Eurowordnet project (funded by EU, LE-2 4003).

Using the SAREL system, the information corresponding to the software specification document, which is represented in the KB and in the RB, can be shared by the software engineers in a more reliable format. These kinds of documents within a complex system are bulky. However, our system allows their

handling at a conceptual level, which helps to detect, for instance, inconsistencies or incompleteness.

This research is partially supported by a grant from the Catalan Government (CIRIT 1997SGR51).

## 7. References.

- [1] Alvarez, J., Yet Another Yet Another (YAYA). *Technical report, LSI-96-15-T* Dept. of LSI, Universitat Politècnica de Catalunya, Barcelona, Spain, 1996.
- [2] Castell, N. & Hernández, A., Filtering Software Specifications Written in Natural Language. *Proceedings of the 7th Portuguese Conference on Artificial Intelligence (EPIA'95)*, LNAI 990, pp 447-455, Funchal, Madeira Island, Portugal, 1995.
- [3] Castell, N. & Slavkova, O. & Toussaint, Y. & Tuells, A., 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)*, pp 37-44, Austin, Texas, USA, 1994.
- [4] Castell, N. & Hernández, A., The Software Requirements Modeling in SAREL. *Proceedings of the 4th International Workshop on Requirements Engineering: Foundations of Software Quality (REFSQ'98)*, pp 49-56, Pisa, Italy, 1998.
- [5] Castell, N. & Hernández, A., 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)*, pp 529-539, Málaga, Spain, 1997.
- [6] Fuchs, E., Schwitter, R. Attempto Controlled English (ACE). *First International Workshop On Controlled Language Applications*. Katholieke Universiteit Leuven, Belgium 1996.
- [7] IEEE Std 830-1993: *IEEE Recommended Practice for Software Requirements Specifications*, 1993.
- [8] Jarke, M. et al. Theory Underlying Requirements engineering: An Overview of NATURE at Genesis. *IEEE International Symposium on Requirements Engineering (RE'93)* San Diego, California, USA, 1993.
- [9] Melchisedech, R., Investigation of Requirements Documents Written in Natural Language, *Requirements Engineering* (1998) 3:2, pp 91-97.
- [10] Romesburg, H.C., Cluster analysis for researchers. Belmont, Calif.: *Lifetime Learning Publications*, 1984.
- [11] Tschaischian, B., Wenzel, . and John, I. 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.