

# **SAREL : An Assistance System for Writing Software Specifications in Natural Language**

**Angels Hernández**

Departament de Llenguatges i Sistemes Informàtics

Pau Gargallo, 5. Barcelona 08028. SPAIN

telephone number : +34-3-401 73 31

fax number : +34-3-401 70 14

The specification phase is one of the most important and least supported part of software development process. In order to improve the quality control of software at this earlier phase, it is important to identify the quality properties that would be accomplished by a set of requirements forming a specification. Among the well-known quality properties, we are interested in five of them that we consider the most important : consistency, completeness, traceability, verifiability and modifiability.

We have conceived SAREL as a tool to improve the specification phase. The purpose of SAREL is to assist engineers in the creation of software specifications written in natural language. The assistance process takes into account the writing norms and the above mentioned quality properties. To do so, the SAREL system relies on a variety of controls that validate the set of requirements using lexical, syntactic, semantic and domain information. SAREL distinguishes two kinds of knowledge: the Knowledge Base, which contains domain representation, and the Requirements Base, which contains requirements representation. Once a requirement has been validated and is correct, its conceptual representation is added to the Requirements Base.

## **1. Introduction**

The software development process starts generally with the specification phase. In this stage it is very important to control the quality of the specifications in order to detect possible mistakes as early as possible. The correction of errors in the development and implementation phases implies spending more time and effort than in the specification phase. This is the reason why the developers increasingly try to identify the possible mistakes on the early phases of software development.

During the initial phase, the software specifications for complex systems (such as those in the aerospace and nuclear fields) result in bulky documents since they are often written in natural language. Even after being formalized, this original documentation may also serve during later phases, and during the functioning and maintenance of the developed computer system. Documentation writing is guided by the norms which define the linguistic restrictions required to satisfy the specifications. These norms are of two types: those relating to the use of natural language in general (for example, [2] and [1]); and those that are based on terminological

restrictions related to a particular domain (for example, the ESA - European Space Agency - norms). Both of them restrict the use of natural language through a set of rules which limit various irregularities (polysemy, paraphrase, ambiguity, vagueness..) which occur during the interpretation of natural language. Even though the norms define linguistically precise restrictions, the frequent failure to observe them makes it difficult for the consequence of such breaches to be detected afterwards. In addition to linguistic restrictions, the norms also include Software Engineering constraints related to the quality factors of the specifications, such as consistency, completeness, traceability, modifiability and verifiability.

In section 2, we describe our system (SAREL). Its purpose is to assist engineers in writing specifications in natural language taking into account the writing norms and the software quality properties. In section 3, we analyse the common aspects between SAREL and others systems which tackle the problems associated with the specification phase. This paper concludes with a discussion of the future goals of our research.

## **2. SAREL: an assistance system for writing software specifications in natural language**

The SAREL system is part of a more ambitious program of research and development called LESD (Linguistic Engineering for Software Design). This project was instigated by the ARAMIIHS center in Toulouse (France). This center was established through a joint agreement between the CNRS and MATRA MARCONI SPACE company. The project is carried out by researchers from IRIT (Institute de Recherche en Informatique de Toulouse), from the Paul Sabatier University of Sciences, from Le Mirail University of Humanities, from the MATRA company, and, since 1991, by several researchers from the Technical Univeristy of Catalonia through a Spanish-French grantfunded joint initiative.

LESD aims [3], [12] to develop computational tools which will (1) allow conceptual interpretation of functional or preliminary software aerospace specifications written in English; (2) permit evaluation of quality factors by means of reasoning algorithms applied to the conceptual representation; and (3) help the engineers handle documentation.

This project involves an in-depth study of the language and, thus, integrates lexical, grammatical, and semantic components. In addition the knowledge domain has been considered and represented as it is vital to conceptual interpretation and the reasoning process [11].

In this context, the main goal of SAREL is to assist an engineer in the creation of software specifications written in natural language. More precisely, specifications are written in English because this is the common language in the aerospace domain. The assistance process, decomposed in several steps (see figure 1), validates every requirement introduced by the engineer taking into account the writing norms (for instance [1], [2]) and the quality properties [5]. This process incrementally constructs a conceptual representation of the specification. The controls showed in figure 1 can be grouped into two modules: the Style Refinement Module and the Conceptual Refinement Module. The first module analyses the quality of the requirement itself,

and the second one analyses the quality of the requirement taking into account the set of requirements represented in the Requirements Base. Once a requirement has been checked and is correct, its conceptual representation is added to the Requirements Base. Next, we describe in detail these two modules as well as the different analysis and controls that are being implemented.

Figure 1. Validation process by SAREL

## **2.1. Style refinement module**

This module controls the requirement according to the writing norms and is composed of four steps: lexical analysis, syntactic-semantic analysis, ambiguity control and simplicity control. The style refinement module controls at first the lexicon used in a requirement, then analyses its coherence and finally validates its surface form. The examples we present in this paper are taken from an aerospace system documentation. At the moment the syntactic-semantic analyzer has been imple-

mented and we are working on the implementation of the rest of controls.

## Lexical analysis

This analysis carries out the control of lexicon used in the specification. Given a requirement, the lexical analysis verifies that the words belong to the application domain lexicon. To do so, SAREL uses three different lexicons:

- The extended lexicon includes all words related to the application domain.
- The allowed lexicon is an extraction of the previous one. Every word in the allowed lexicon represents a set of synonymous on the extended one.
- The ground lexicon contains the basic words in the English language (prepositions, articles, ...).

The input of this analysis is a sentence composed of words that belong to the extended and ground lexicons. If the requirement contains at least one word not belonging neither to the extended nor to the ground lexicons, it will be refused. The output is the same requirement described using the allowed and ground lexicons. Next we present an example of lexical refinement:

**Req.:** *“The IOIGS shall manage the computer on-board the space-vehicle”*

**Ans.:** *“The word manage is not allowed. The suggested expression for this requirement is: The IOIGS shall monitor the computer on-board the space-vehicle”*

The lexicon division offers the engineer some kind of writing flexibility, because he can use an extended lexicon which will be refined on to the allowed lexicon. Moreover, the change of application domain implies only the change of extended and allowed lexicon, because the lexical validation process is the same.

## Syntactic-semantic analysis

In this step, the ALVEY parser [4] is used. This parser is based on Montague’s logic. It maintains that the composition rules are associated with syntactic rules, in the same way the syntactic rules are associated with semantic composition rules. The ALVEY parser offer us a lexicon with 65.000 entries related to 28.000 words and a set of syntactic rules covering a significant part of normal English. This parser has been modified and adapted to aerospace domain and it has a satisfactory outcome.

Although semantic rules are associated to syntactic ones, the GDE (Grammar Development Environment) is not able to capture the whole meaning of a requirement and a conceptual analysis must be further performed. The output is a tree-like semantic representation. Figure 2, from [11], shows the semantic representation for

the requirement: “*The IOIGS shall monitor the status of the space vehicle*”

Figure 2. Semantic Representation

### Ambiguity control

It is possible to obtain more than one semantic representation from a requirement. This situation appears when there exists some kind of ambiguity in the requirement. The function of the ambiguity controller is to distinguish the representation which corresponds to the engineer idea. For example from the requirement “*The IOIGS shall monitor the computer on-board and the status of the space-vehicle*” we can get two possible interpretations:

- a) “*The IOIGS shall monitor the computer on-board of the space-vehicle and the IOIGS shall monitor the status of the space vehicle*”
- b) “*The IOIGS shall monitor the computer on-board and the IOIGS shall monitor the status of the space vehicle*”

The ambiguity controller applies some rules to the set of semantic representations in order to qualify them. It then cooperates with the engineer to decide which the correct semantic representation is. An example of a rule is the one which asserts that the *of* preposition is always related with the last nominal group.

### Simplicity control

The function of this control is to analyse the simplicity of the requirement. This is one of the properties expressed by the writing norms. From the semantic

representation of a requirement, the simplicity controller detects whether the structure of the sentence is simple or compound. The following requirement does not accomplish this property.

**Req.:** *“The IOIGS shall control the computer on-board of the space-vehicle and the IOIGS shall control the automatic systems of the flight configuration”*

**Ans.:** *“The requirement is a conjunction of two simple requirements”*

If the requirement is composed of two simple requirements, there exists the possibility that the process continues with these two requirements in a sequential way.

## 2.2. Conceptual refinement module

This module carries out a series of controls that validate the requirement in relation to the Requirements Base (RB). At first it obtains a conceptual representation using the Knowledge Base (KB). Both RB and KB use a frame-based formalism [11]. From the conceptual representation this module controls the duplicity of information and finally it validates the requirement activating the reasoning tools developed in LESD to control the quality properties (*completeness, traceability, consistency, verifiability* and *modifiability*).

At this moment we have already defined the Knowledge Base, and the Requirements Base is manually incremented when requirements have been analysed. The automatic process to integrate requirements is in a development phase.

### Conceptual analysis

This analysis identifies in the Knowledge Base the concepts involved on the semantic representation. From this information it constructs a conceptual representation of the requirement. Figure 3, from [11], shows the conceptual representation of two requirements:

**Req-1:** *“The IOIGS shall monitor the system of the space-vehicle”*

**Req-2:** *“The IOIGS shall control the automatic systems of the space-vehicle”*

Figure 3. Conceptual Representation

### Duplicity control

The function of this control is to validate that the requirement introduced by the engineer contains new information. To do so, the Duplicity Controller matches the Conceptual Representation of a requirement to the Requirements Base in order to discover a possible duplicities. The following example shows two equivalent requirements since the system obtains the same conceptual representation for both.

**Req-1:** *“The IOIGS shall monitor the computer on-board the space-vehicle”*

**Req-2:** *“The computer on-board the space-vehicle shall be monitorized by the IOIGS”*

The activity in both is *monitor*, the agent which performs the action is the *IOIGS* and the object, which suffer the actions in both, is the *computer on-board the space-vehicle*. If this Controller detects duplicity in the introduced requirement,

it will offer the analyst the possibility to refine it.

## Analysis of traceability

The goal here is to offer information to the engineer about the traceability links of the requirement. To do so, the system activates a set of algorithms already developed which control the traceability in order to show the relationships between the introduced requirement and a subset of requirements of the RB, either more specific or more general. From this information the engineer knows the relationships between requirements introduced up to this point. From the next requirement :

**Req-1:** “*The IOIGS shall receive the data of the space vehicle*”

the system displays, among others, the following more general requirement:

**Req-2:** “*The system shall process the data of the space vehicle*”

The second requirement is more general than the first because IOIGS is an instance of system and receive is a subclass of process.

## Analysis of completeness

Also in this step the system activates the reasoning mechanisms of LESD which are being developed taking into account several aspects of the concept *completeness*. In order to control the completeness, a general hierarchy of actions-subactions that can be associated to any set of specifications related with an aerospace system [14] is necessary. An exemple of this hierarchy is present in the activity *monitor*, which is composed of three subactivities: *receive*, *analyse* and *display*. Given the next requirement:

**Req.:** “*During the launch phase, the IOIGS shall analyse and display the status of the space vehicle*”

Taking into account the hierarchy described above, the system analyses the relationships among requirements in the Requirements Base. If there is no requirement containing the *monitor* activity, it will inform the engineer. In the same way, if the system notices that there is a requirement that contains the *monitor* activity but there is no requirement containing the subactivities on the hierarchy, it will suggest to the engineer that the current specification is incomplete.

Once a requirement has been validated and is correct, it will be added to the Requirements Base. This process incrementally constructs the conceptual representation of the specification written in natural language.



### 3. Related work

In this section we present a comparative analysis between SAREL and three systems (RA, FRORL development system and ISLET) supporting the specification phase in order to place our research work.

The Requirements Apprentice (RA) assists a human analyst in the creation and modification of software requirements. The FRORL development system facilitates the specification, analysis and development of a software system. The ISLET system supports the construction of formal specifications and their incremental refinement into verified implementations. Although the above system goals are even different from SAREL system goal, they form a good set of references for our research work. A short description of these systems is offered in the Appendix.

#### 3.1. RA & SAREL

The Requirement Apprentice (RA) is the closest system to SAREL. The input for RA is a set of commands while SAREL accepts specifications in natural language, therefore RA does not consider linguistic problems as SAREL does. Nevertheless, we have identified a set of common elements between these two systems:

- a) The Library Cliché corresponds to the Knowledge Base since the library contains information about the application domain.
- b) The Requirements Knowledge Base in RA is related to the Requirements Base in SAREL.
- c) The Cake module corresponds to the Conceptual Refinement Module, because it provides basic facilities in order to validate the set of requirements.

According to quality control, the RA checks the requirement for consistency, completeness, ambiguity and modifiability. On the other hand, the conceptual validation in SAREL is more extensive and is based on writing norms and the five mentioned properties: traceability, completeness, consistency, verifiability and modifiability.

#### 3.2. FRORL Development System & SAREL

In comparing these systems, we have kept in mind that their goals are different. The FRORL Development System aims at supporting the software development. Starting from a set of informal requirements, the system achieves its corresponding Prolog code. The goal of SAREL is to obtain a validated conceptual representation from a specification written in natural language. In spite of that, we have detected some similarities between SAREL and the early stage of the architecture of FRORL Development System:

- a) The starting point in both systems is a set of informal requirements. FRORL development system begins from a requirement document while the input of SAREL is a specification written in natural language. However, in the first system the developer translates the requirements into FRORL sentences, while SAREL supports a really true natural language processing.
- b) The Requirements Analysis in FRORL Development System corresponds to the Style Refinement Module and the construction of the Knowledge Base in SAREL. On the one hand, the developers using the FRORL Development System identify the concepts that characterize the application domain. On the other hand, they solve the problems related to the writing style (ambiguity, simplicity, ...).
- c) The FRORL development methodology is related with the conceptual analysis, because it describes the specifications in a fixed form susceptible of being checked.
- d) The Specification Analysis corresponds to the set of controls performed on the Conceptual Refinement Module. The output in both systems is a report which shows the properties accomplished by the specification.

The set of properties to be validated is different from one system to the other. The Specification Analysis verifies *software properties* and the Conceptual Refinement Module validates *requirements properties*.

### 3.3. ISLET & SAREL

In spite of the differences between these two systems, the idea of refinement is implicit in both. The design derivation process in ISLET uses stepwise refinement to transform pre and post-condition specifications into verified programs. The assistance system SAREL uses stepwise refinement to transform a specification written in natural language into its validated conceptual representation.

We also want to mention the NATURE project [6]. It is a more ambitious project which goal is to develop a set of interacting theories that relate knowledge representation, domain analysis and process support aspects of requirements engineering. We would notice that NATURE has several common aspects with LESD project, which contains the SAREL system, in spite of LESD not being so ambitious as NATURE.

## 4. Conclusion and future research

Our research deals with the control of quality in specifications written in natural language. Since the specification phase is performed by a human analyst, we have designed an assistance system for writing software specifications in natural language. The quality of the specification is studied from two points of view: writing norms (style refinement) and quality properties (conceptual refinement). The

quality properties we are interested in are : traceability, completeness, verifiability, consistency and modifiability.

The assistance process is divided into two modules: the Style Refinement Module and the Conceptual Refinement Module. Once a requirement has been validated by these two modules, it is added to the Requirements Base. During this process we incrementally obtain a conceptual representation of the specification.

Some of our future research directions are the global implementation of controls contained in both modules and the development of new controls related to the verifiability, consistency and modifiability properties.

## 5. References

- [1] ANSI/IEEE Std 729-1983. *IEEE Guide to Software Requirements Specifications* 1983.
- [2] Association Européenne des Constructeurs de Matériel Aéronautique. *AECMA Simplified English, A Guide for the preparation of aircraft maintenance documentation in the international aerospace maintenance language*, December 1989.
- [3] Borillo M., Toussaint Y., and Borillo “A. Motivations du project LESD”. In *Conference on Linguistic Engineering’91*, Versailles, France, January 1991.
- [4] Briscoe T., Grover C., Boguraev B., Carroll J. “The ALVEY Natural Language Tools Project Grammar: A Large Computational Grammar”. Technical report, ALVEY Documents, Cambridge Univ., Computer Laboratory, UK, 1987.
- [5] Castell N., Slavkova O., Toussaint Y. and Tuells A. “Quality Control of Software Specifications written in Natural Language”. In *Proceedings of the Seventh International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE’94)*, Austin, Texas, USA, 1994.
- [6] Jarke M., Bubenko J., Rolland C., Sutcliffe A. and Vassiliou J. “Theory Underlying Requirement Engineering: An Overview of NATURE at Genesis”. In *Proceedings of the IEEE International Symposium on Requirements Engineering (RE’93)*, San Diego, California, USA, 1993.
- [7] Reubenstein H.B. and Waters R.C. “The Requirements Apprentice: Automated Assistance for Requirements Acquisition”. *IEEE Transactions on Software Engineering*, 17:226-240, 1991.
- [8] Rich C. and Waters R.C. “The Programmer’s Apprentice: A research overview” *Computer*, 21:10-25, November 1988.
- [9] Rich C. and Waters R.C. *The Programmer’s Apprentice*. Reading, MA: Addison-Wesley, and Baltimore, MD. ACM Press, 1990.
- [10] Terwilliger R.B. “Towards Tools to Support the Gries/Dijkstra Design Process”. *ACM SIGSOFT Software Engineering Notes*, 18:50-59, 1993.
- [11] Toussaint Y. *Méthodes Informatiques et Linguistiques pour l’Aide a la Spécification de Logiciel*. PhD thesis, Universidad Paul Sabatier, Toulouse, 1992.

- [12] Toussaint Y., Borillo M., Borillo A., Castell N. and Latour D. "Applying Linguistic Engineering to Software Engineering". 26th Linguistic Colloquium. Poznan, Poland, September 1991. (Published in *Linguistische Arbeiten*, n 293, pp 209-217, Max Niemeyer Verlag, 1993)
- [13] Tsai J.P., Weigert T. and Jang H.C. "A Hybrid Knowledge Representation as a Basis of Requirement Specification and Specification Analysis". *IEEE Transactions on Software Engineering*, 18:1076-1100, 1992.
- [14] Tuells A., Castell N. "The completeness problem in LESD", Technical report, LSI-93-51-R Dept. of LSI, Universitat Politècnica de Catalunya, 1993.