

The use of SAREL to control the correspondence between Specification Documents

Núria Castell Àngels Hernández
Departament de Llenguatges i Sistemes Informàtics
Jordi Girona 1-3. Mòdul C6.
Barcelona, Spain
e-mail: {castell,ahernandez}@lsi.upc.es

Keywords: Software Specifications, Software Quality Control, Intelligent Assistance, Natural Language

ABSTRACT. *This paper aims to describe an extension of the SAREL system (Assistance System for Writing Software Specification in Natural Language) whose main goal is to improve the specification phase. The initial purpose of SAREL was to assist engineers in the creation of software specification written in natural language. At present the functionality of the system is twofold: vertical processing and horizontal processing. In vertical processing the input is a software specification written in natural language and the output is the conceptual representation associated. In this case the system validates every requirement taking into account the writing norms and the quality properties. The conceptual representation can be used to check the software quality factors. Working in horizontal processing the input is two different conceptual representations and the output is the information about the correspondence between them.*

1 INTRODUCTION.

The specification phase is one of the most important and least supported parts of the software development process. 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 in the early phases of software development. During the initial phase, the software specifications are often written in natural language. Therefore it is important to control the writing of these documents because many mistakes can be detected if the writing is clear and concise. In this work frame there are other systems which tackle the problems associated with the specification phase like RA [Reubenstein&Waters91], NATURE [Jarke et al.,93] or with the quality of documents in general like EasyEnglish [Bernth96], ACE [Fuchs&Schwitter96].

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, [ANSI83] and [AECMA89]), 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. 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 spite of these considerations it is difficult to find software specifications that follow and respect these writing norms. The documents in general correspond to different styles of writing and have different structures depending on the document's goal. From that it seems difficult to control the writing without taking into account what kind of document the engineer wants to write.

In section 2 we present briefly the different Modules contained in the SAREL system. In section 3 we discuss the different types of documents that make up the specification phase. Section 4 contains a description of the different ways of working of SAREL. The conclusions are presented in section 5.

2 THE SAREL SYSTEM.

The main goal of SAREL is to assist an engineer in the creation of software specifications written in natural language. A preliminary version of this system was described in [Hernandez94]. The assistance process, split into several steps, validates every requirement introduced by the engineer taking into account the writing norms (for instance [ANSI83]), [AECMA89]) and the quality properties [Castell et al.,94]. This process incrementally constructs a conceptual representation of the specification. The controls, shown in figure 1, can be grouped into three modules: the Style Refinement Module, the Conceptual Refinement Module and the Software Quality Control Module.

The first module controls the requirement according to the writing norms and it is split into four steps: (1) the lexical analysis verifies that the words belong to the application domain lexicon; (2) the syntactic-semantic analysis is produced using ALVEY tools [Briscoe et al.,87] 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 of the sentence 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 Knowledge Base and after that detects duplicated information.

The Software Quality Control Module carries out a series of optional analyses which validate [Alvarez&Castell94], [Castell&Slavkova93], [Castell&Slavkova95] the global Requirements Base incremented with the new requirement. The goal is to offer information about the software quality properties which have been considered most relevant in LESD [Borillo et al.,91], [Borillo et al.,92](completeness, traceability, consistency, verifiability and modifiability).

Once a requirement has been checked and seen to be correct, its conceptual representation

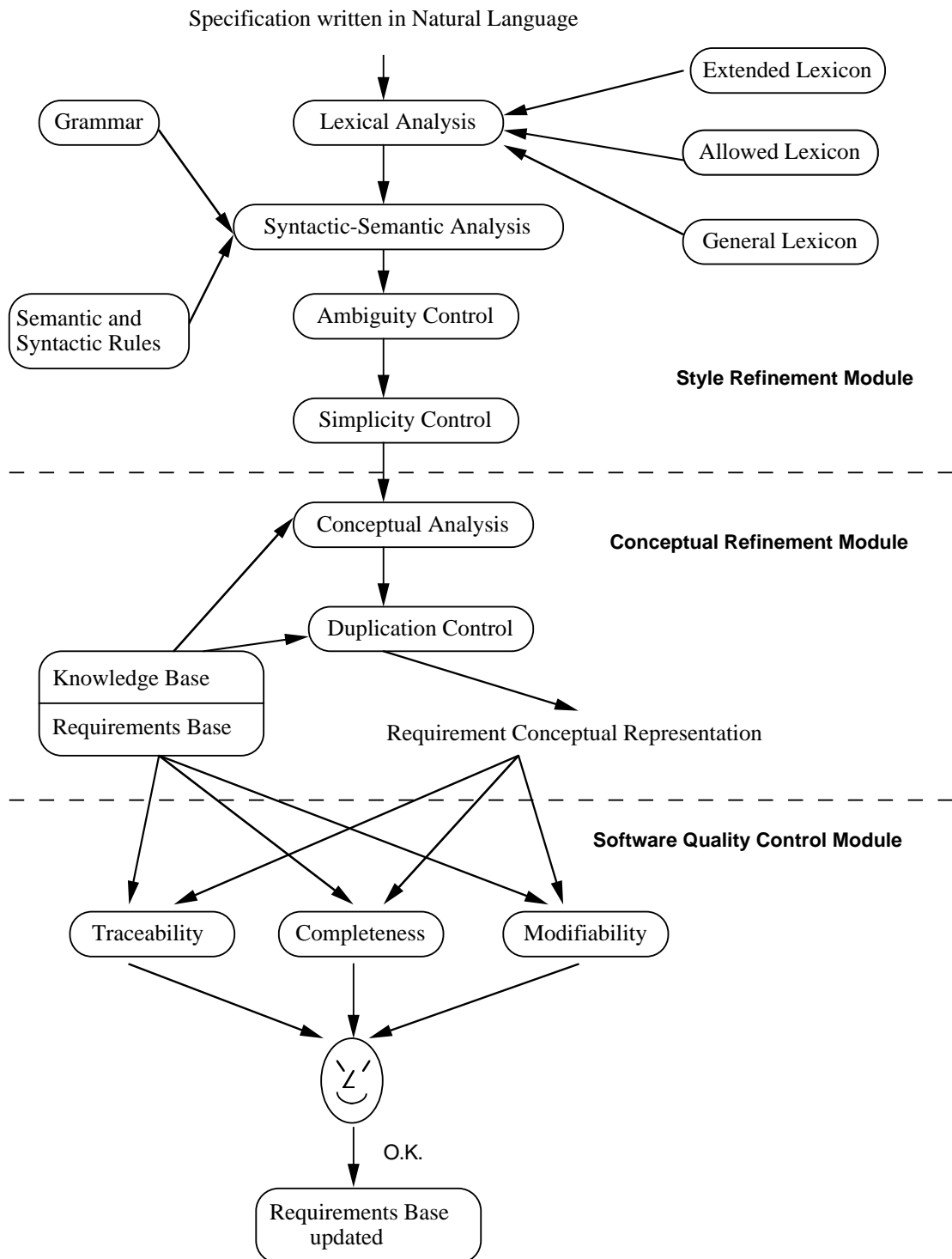


Figure 1. Modules of SAREL

is added to the Requirements Base. This base uses a frame-based formalism [Toussaint92]. A more precise description of this system can be found in [Castell&Hernández95].

3 THE SOFTWARE SPECIFICATIONS.

The first question is what kind of preliminary specifications can be found in the software specification phase. At present we have identified two different situations depending on the participants in the specifications phase.

In the first situation the set of software specifications belongs to a unique work frame. In most cases they correspond to complex systems software specifications (such as those in the aerospace, nuclear and telecommunications fields). They result in bulky documents that may also serve during later phases, and during the functioning and maintenance of the developed computer system.

Due to the complexity of these software specifications normally they are written by several engineers that cooperate during the specification phase. This means it is important that the information introduced by different people would not be contradictory. In order to control the writing of these documents the different engineers should take into account the writing norms (they are not followed completely) and the issue of this specification process has to accomplish the software quality properties mentioned above.

In the second situation during the specification phase it is possible to find different documents that contain in theory the same software specification belonging to different work frames. In this case the supply-demand system leads to at least two different kind of documents: one corresponding to the user company that needs to develop a computer system and the other one corresponding to the software company that will do this. The first document corresponds to a description of the user's requirements, and the latter has to contain all the information needed for the design phase. Frequently in this situation there are a lot of misunderstandings so on one hand the writing norms do not exist and on the other hand there are different views of the same problem.

An example of that situation occurs in telecommunications companies, where the specification phase starts with a preliminary software specification made by the client. This first document reflects the client's requirements and after an analysis of it, the telecommunications company makes another one describing how these requirements can be accomplished with its technology. This second document is shown to the client in order to get his acknowledgment about the conformity to his needs.

The main problem here is to clarify or resolve as early as possible the misunderstandings between the company software engineers and the client in order to get this revision process as short as possible.

Another example of the second situation is the case when the specification phase is divided in fixed steps. For instance in public institutions the usual steps are :

- Publication of the Technical Conditions Description
- Presentation of offers
- Awarding

In the first step the public institution makes one document that can be of two types: Technical Conditions Description to develop a new computer system (TCDD), or Technical Conditions Description to maintain a developed computer system (TCDM). During the second

step the public institution receives many documents corresponding to the different offers (OF), and in the third the public institution generates only one, the Awarding Report (AR). Every document is composed of a set of sections, many of which are obligatory. Focusing our attention on the Technical Conditions Description Document it can be structured in the follow sections:

- Purpose
- General Description
- Functional Characteristics of Software
- Implementation
- Integration with other applications
- Security
- Documentation
- Technical Management
- Deadlines
- Guarantee and maintenance
- Start up
- Experience
- Deliberation Criteria

In a similar way, as with the telecommunications area, it is necessary to control how the offer documents correspond with the original Technical Conditions Description. This information about the correspondence between them also could be used during the awarding phase.

4 THE TWO FUNCTIONALITIES OF SAREL.

In this section we present the different ways that SAREL can work depending on the user's goal. There are two principal ways of working showed in figure 2: vertical processing and horizontal processing.

4.1 Vertical Processing

In vertical processing the input is a software specification written in natural language and the output is the Conceptual Representation associated that optionally can be manipulated by the engineer using the Software Quality Control Module.

If only one document is used in the specification phase (the first situation described in section 3) the SAREL system will assist the engineers in the creation of this software specification. This process validates every requirement introduced by the engineer using the Style Refinement Module and the Conceptual Refinement Module. Once a requirement has been checked and it is correct, its conceptual representation is added to the Requirements Base. At the end the issue is the Conceptual Representation corresponding to all the information contained in the software specification. Using the SAREL system the information contained in the Knowledge-Base and in the Requirements Base can be shared by the engineers in a more reliable format.

If the specification phase is composed of several documents (the second situation described in section 3), the SAREL system first has to identify what kind of document the input is. At this point it is necessary to know the information associated with the structure

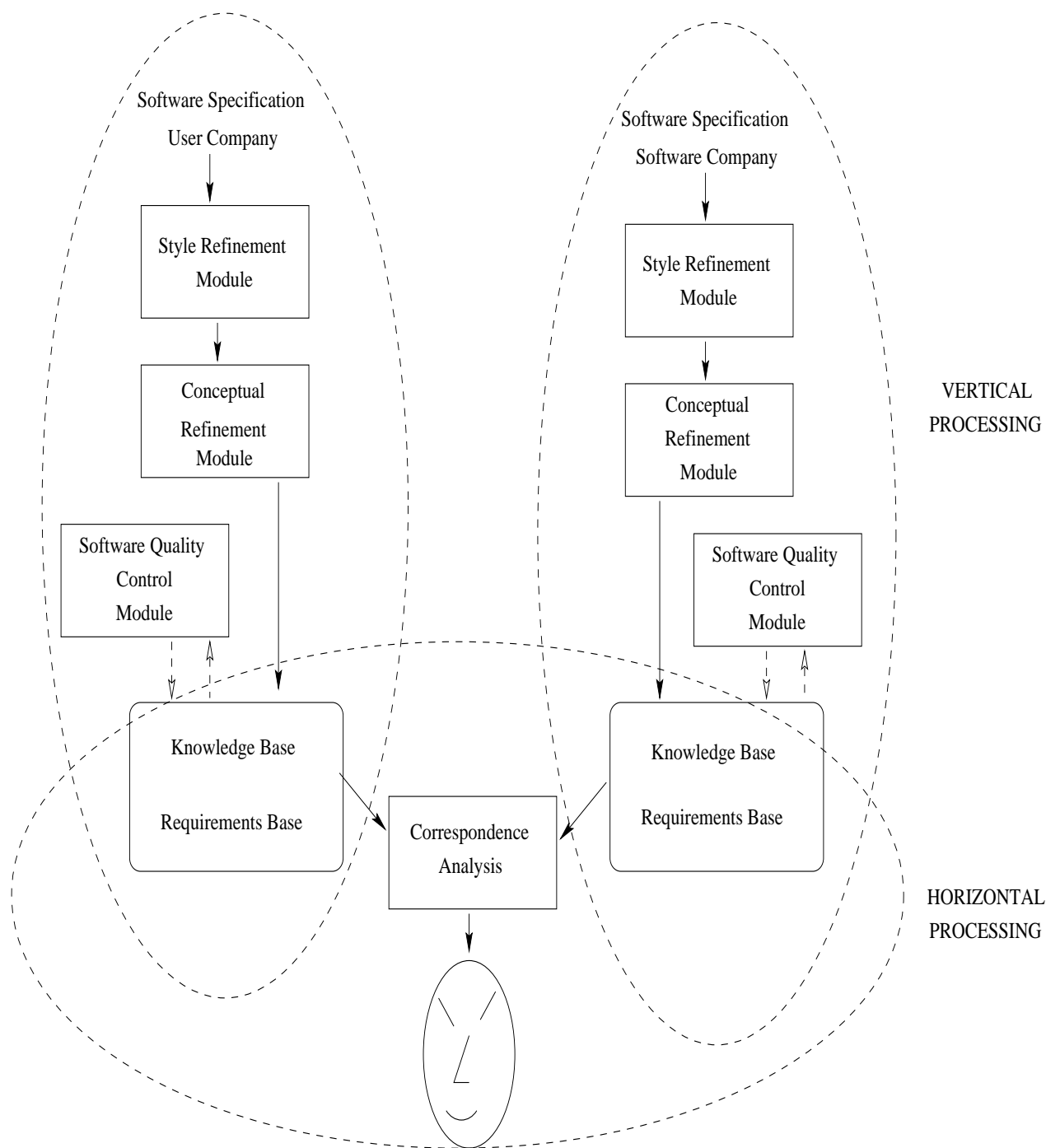


Figure 2. Vertical Processing and Horizontal Processing

of these documents. For this reason the system will use a Document-Knowledge Base. In case the system does not find any structure matching with the document there is the possibility to label the document as unknown or to ask the engineer what kind it is. If the document is identified with one type on the Document-Knowledge Base, the SAREL system has to make sure that every section required is present on the document. Even so, as the document has been labeled as unknown or with a specific structure it will be processed by the Style Refinement Module and in the Conceptual Refinement Module in order to obtain the Conceptual Representation.

In both cases, once the conceptual representation has been obtained it can be globally validated using the optional analysis contained in the Software Quality Control Module.

4.2 Horizontal Processing

In horizontal processing the input is two different conceptual representations (associated to two software specification documents), and the goal here is to offer information about the correspondence between them. So the work frame here is the second situation described in section 3.

At this point it is necessary to fix the conceptual representation corresponding to the client's requirements (*document₁*), the one corresponding to the software company (*document₂*) and which of the two conceptual representations is of more value.

Now we must define the correspondence relationship at different levels: entity, relation, requirement and document. Let us take the case in which *document₁* should be more relevant.

- Entity correspondence: an *entity₂* corresponds with an *entity₁* if the values associated to the mandatory attributes on the *entity₁* are the same or compatible with *entity₂*. Taking into account the taxonomy structure if an *entity₂* corresponds with an *entity₁*, it also corresponds with the more general entities than the *entity₁*. Sometimes the correspondence between two entities requires a preliminar transformation depending on the entity's role.
- Relation correspondence: a *relation₂* corresponds with a *relation₁* if they are the same or if it is more specific than the *relation₁*. Taking into account the taxonomy structure if a *relation₂* corresponds with a *relation₁*, it also corresponds with the more general relations than the *relation₁*.
- Requirement correspondence: a *requirement₂* has a total correspondence with a *requirement₁* if all the entities in *requirement₁* correspond with entities in *requirement₂* and they are linked with correspondence relations. In any case the system will give a correspondence measure based on similarity analyses [Romesburg84] applied over the components (entities and relations) of the requirements.
- Document correspondence: the correspondence between documents will be defined in terms of the correspondence between the requirements contained into the documents.

The correspondence analysis search for every *requirement₁* in the *document₁* its correspondent *requirement₂* in the *document₂*. The result can be as follows:

- Correct correspondence: all the entities and all the relations in *requirements₁* have correspondence with entities and relations in *requirement₂* and vice-versa.
- Excess correspondence: all the entities and all the relations in *requirements₁* have correspondence with entities and relations in *requirement₂* but there are entities or relations in *requirement₂* that have no correspondence. The following exemple corres-

ponds with this situation where we have considered the entity *communications-server* has correspondence with *communications-server-software* (a more specific element) and where *requirement₂* adds the *modality* component.

```
(define-node requeriment1
(instance mandatory-requeriment)
(agent server-transfer-system)
(activity transfer)
(object clock-in)
(destination communications-server))
```

```
(define-node requeriment2
(instance mandatory-requeriment)
(agent server-transfer-system)
(activity transfer)
(modality point-to-point)
(object clock-in)
(destination communications-server-software))
```

- Insufficient correspondence: all the entities and all the relations in *requirements₂* have correspondence with entities and relations in *requirement₁* but there are entities or relations in *requirement₁* that have no correspondence. In this exemple *requirements₂* does not contain the *agent* component.

```
(define-node requeriment1
(instance mandatory-requeriment)
(agent server-transfer-system)
(activity transfer)
(object clock-in)
(destination communications-server))
```

```
(define-node requeriment2
(instance mandatory-requeriment)
(activity transfer)
(object clock-in)
(destination communications-server-software))
```

- Excess-Insufficient correspondence: a set of entities and relations in *requirements₁* have correspondence with entities and relations in *requirement₂*. But there are entities or relations in *requirement₁* that have no correspondence and there are entities or relations in *requirement₂* that have no correspondence. In this exemple, *requirement₁* contains the *object* component, not present in *requirement₂*, and *requirement₂* contains the *modality* component, not present in *requirement₁*

```
(define-node requeriment1
(instance mandatory-requeriment)
```



```

(agent server-transfer-system)
(activity transfer)
(object clock-in)
(destination communications-server))

(define-node requeriment2
(instance mandatory-requeriment)
(agent server-transfer-system)
(activity transfer)
(modality point-to-point)
(destination communications-server-software))

```

Taking into account this information the engineer can identify if the excess correspondence is caused by a document that adds more details/restrictions to the original document. On the other hand the insufficient correspondence could be the result of a more general document than that of original.

5 CONCLUSIONS.

At the begining, the SAREL system was conceived to assist engineers in the creation of software specifications using a knowledge-based approach. Due to the complexity of the specification phase we have found in many situations there are more than one document used in this phase. In this sense the system has been extended for helping engineers in two ways. On one hand in vertical processing only one document at time is considered, and the system assists the engineer in the creation of this document (taking into account the linguistics controls and software quality controls). In this case the main goal is to improve the quality of the preliminary specification. And on the other hand in horizontal processing the system analyses the correspondence between two software specifications to evaluate if the client's requirements will be satisfied by the software developed according to the engineer's requirements.

References

- [Alvarez&Castell94] Alvarez, J. & Castell, N. *An Approach to the Control of Completeness Based on MetaKnowledge*, Technical report, LSI-94-50-R Dept. of LSI, Universitat Politècnica de Catalunya, 1994.
- [ANSI83] ANSI/IEEE Std 729-1983: *IEEE Guide to Software Requirements Specifications* 1983.
- [AECMA89] Association Européene 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.

- [Bernth96] Bernth, A. *EasyEnglish: A Tool for Improving Document Quality*. In *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP'96)*. pages 159-165.
- [Borillo et al.,91] Borillo, M. & Toussaint, Y. & Borillo, A. *Motivations du project LESD*. In *Conference on Linguistic Engineering'91*, Versailles, France, January 1991.
- [Borillo et al.,92] Borillo, M. & Borillo, A. & Castell, N. & Latour, D. & Toussaint, Y. & Verdejo, M.F. *Applying Linguistic Engineering to Software Engineering: The traceability problem*. In *Proceedings of the European Conference on Artificial Intelligence (ECAI92)*, pages 593-595, Viena, Austria, August 1992.
- [Briscoe et al.,87] 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.
- [Castell&Hernández95] Castell, N. & Hernández, A. *Filtering Software Specifications Written in Natural Language*, In *Proceedings of the 7th Portuguese Conference on Artificial Intelligence (EPIA '95)*, pages 447-455, Funchal, Madeira Island, Portugal 1995.
- [Castell et al.,94] Castell, N. & Slavkova, O. & Toussaint, Y. & 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.
- [Castell&Slavkova93] Castell, N. & Slavkova, O. *The Modifiability Factor in the LESD Project: Definition and Practical Results*, Technical report, LSI-95-7-R Dept. of LSI, Universitat Politècnica de Catalunya, 1993.
- [Castell&Slavkova95] Castell, N. & Slavkova, O. *Metrics for Quality Factors in the LESD Project*. In *5th European Software Engineering Conference (ESEC'95)*, Sitges, Spain, 1995.
- [Fuchs&Schwitter96] Fuchs, E. & Schwitter, R. *Attempto Controlled English (ACE)*. In *Proceedings of The First International Workshop On Controlled Language Applications*. Katholieke Universiteit Leuven, pages 124-136, Belgium, 1996.
- [Hernandez94] Hernández A. *SAREL: An assistance system for writing software specifications in natural language*. In *Proceedings of the IBERAMIA '94*, ISBN 980-6168-16-X Caracas, Venezuela, 1994.
- [Jarke et al.,93] 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.
- [Reubenstein&Waters91] Reubenstein, H.B. & Waters, R.C. *The Requirements Apprentice: Automated Assistance for Requirements Acquisition*. *IEEE Transactions on Software Engineering* 17:226-240 1991.

- [Romesburg84] Romesburg, H. C. *Cluster analysis for researchers*. Belmont, Calif.:Lifetime Learning Publications, 1984.
- [Toussaint92] Toussaint, Y. *Méthodes Informatiques et Linguistiques pour l'Aide a la Spécification de Logiciel*. PhD thesis, Universidad Paul Sabatier, Toulouse, 1992.