An Experience in Using a Tool for Evaluating a Large Set of Natural Language Requirements

Antonio Bucchiarone
Service Oriented Applications Group, Fondazione Bruno Kessler, Trento, Italy
bucchiarone@fbk.eu

Stefania Gnesi
Istituto di Scienze e Tecnologie dell'Informazione, CNR, Pisa, Italy
stefania.gnesi@isti.cnr.it

Alessandro Fantechi
Dipartimento di Sistemi e Informatica, Universita' degli studi di Firenze, Italy
fantechi@di.unifi.it

Gianluca Trentanni
Istituto di Scienze e Tecnologie dell'Informazione, CNR, Pisa, Italy
gianluca.trentanni@isti.cnr.it

ABSTRACT

Requirements analysis is an important phase in a software project. It is often performed in an informal way by specialists who review documents looking for ambiguities, technical inconsistencies and incompleteness. Automatic evaluation of Natural Language (NL) requirements documents has been proposed as a means to improve the quality of the system under development. We show how the tool QUARS EXPRESS, introduced in a quality analysis process, is able to manage complex and structured requirement documents containing metadata, and to produce an analysis report rich of categorized information that points out linguistic defects and indications about the writing style of NL requirements. In this paper we report our experience using this tool in the automatic analysis of a large collection of natural language requirements, produced inside the MODCONTROL project.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements/Specifications

Keywords

Requirements Analysis, Natural Language Automated Analysis, Natural Language Processing

1. INTRODUCTION

An analysis of the natural language requirements by means of automatic tools has been considered as an added value for guaranteeing the successful outcome of the EU/IP MODTRAIN project, subproject MODCONTROL [23], addressing the standardization of an innovative Train Control and Monitoring System (TCMS) system, due to the capability to point out potential sources of ambiguity and other weaknesses. Particularly, the availability of automatic tools for the quality analysis of Natural Language (NL) requirements [2] is recognized as a key factor. Following the project choices, each TCMS requirement has been stored in a common repository, using RequisitePro [21], TCMS requirements have been stored in a single repository, associating to each requirement several metadata attributes providing several notions of traceability (to the author, to the package, and so on). In order to analyze a so large amount of requirements with respect of their metadata attributes, a modified version of the QUARS (Quality Analyzer for Requirements Specifications) [4] tool has been developed. In particular QUARS EXPRESS is able to handle a more complex and structured data format containing metadata and produces an analysis report rich of categorized information. The information grows as a function of the number of metadata items available (e.g. as a function of the number of authors, the number of packages and so on) and the size of the report grows consequently and can be composed of several pages. As an improvement of the simple text based report made by QUARS, the new report exploits the HTML technology to produce structured hypertextual pages. We have analyzed using QUARS EXPRESS the Functional and System Requirements of TCMS including more than 5700 requirements. The results of the analysis have shown that the analysis process based on QUARS EXPRESS not only can be able to point out linguistic defects, but can provide also some indications on the writing style of different NL requirements authors (from different partners) giving them the opportunity to become aware of defects and of potential improvements.

In the next section we briefly present the MODCONTROL case study. In section 3 we introduce QUARS, and in section 4 we show how it has been modified to cope with the needs of the MODCONTROL project. In section 5 we present the quality analysis process used in the project, in which QUARS EXPRESS is used together with other tools (i.e., IBM RequisitePro and SoDA). In section 6 we discuss the experience made in MODCONTROL while conclusions and future work are presented in section 7.

2. MODCONTROL TCMS CASE STUDY

The MODCONTROL approach has provided Functional Requirements Specification (FRS) and a System Requirements Specification (SRS) for the new generation of TCMS. These specifications will aim at the standardization of essential interfaces of the TCMS with other major subsystems of the train, such as Traction Con-

SAC'10 March 22-26, 2010, Sierre, Switzerland.

trol, Air Conditioning, Doors, Brakes or Auxiliary Power Distribution. During MODCONTROL's specification phase, project partners have gathered requirements from different sources such as specifications of existing trains, standards or drafted specifications from other EU projects. These requirements have then been consolidated, harmonized and refined among the project partners in several review sessions.

The SRD (System Requirements Document) has been generated from the common server of the MODTRAIN project and it is the result of the input provided by the project partners. The SRD expressed as Natural Language sentences, in its current status, is composed of more than 5700 requirements categorized as :Functional Requirements (**FREQ**) , System Requirements (**SREQ**) , Glossary Items (**TERM** and Use Cases (**UC**).

Each requirement is constituted by several attributes, that are the *Text* of a single requirement, its *Source* that is any document from which the requirement derives, the *Responsibility* that is the person who has actually inserted the requirement in the repository, the *Package* which indicates the part of the system the requirement refers to, the *Type* (i.e., Functional, Architectural, Performance, Realtime, etc.). Indeed, both FREQ and SREQ include what are normally called "functional" as well "non-functional" requirements, the latter referring to requirements about Performance, Reliability, and so on. The attribute Type is actually the information that distinguishes these classes.

3. NL REQUIRMENTS ANALYSIS

A NL requirements document, composed by different sources, may suffer differences in style and accuracy producing an unbalanced and ambiguous final requirements document. Several approaches can be followed to ensure a good quality requirements document. One approach is the linguistic analysis of a NL requirements document aimed to remove as many readability and ambiguity issues as possible. Several studies dealing with the evaluation and the achievement of quality in NL requirement documents can be found in the literature and natural language processing (NLP) tools have been recently applied to NL requirements documents for checking the consistency and completeness. Among such tools, and ARM [10] and TIGER PRO [19] perform a lexical analysis of documents detecting and possibly correcting ambiguous terms or wordings, while tools such as LOLITA [6] and Circe-Cico [1] exploit syntactic analyzers to detect ambiguous sentences having different interpretations. QUARS [4], exploits both approaches, the lexical and the syntactical one.

4. QUARS EXPRESS

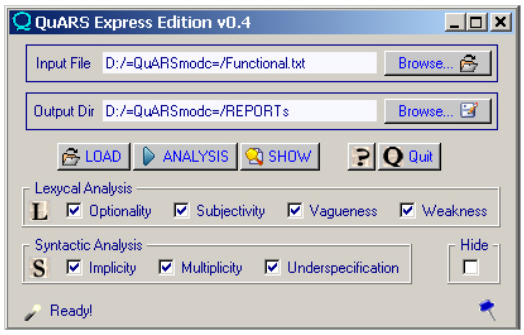

Figure 1: QUARS EXPRESS **Graphical User Interface**

The huge number of requirements to be analyzed, together with the project partners request to manage a minimum metadata set, led us to the development of a new tool (Figure 1) with a more effective interface, able to process large sets of requirements in a batch-like way. Though QUARS EXPRESS shares the same QUARS quality model [4], the five metadata fields (*requirement ID, Responsibility, Type, Source, Package*) newly handled enable the production of an analysis report enriched by more detailed metrics and statistics.

In order to handle metadata, manage the requirements sets and provide requirement traceability, the tool has been interfaced with the RequisitePro based repository by means of the SoDA plug-in [22] and by the definition of an exchange plain text format (see an example in Figure 2).

The produced analysis report is tailored to be used for both analysis and correction purposes, or for productiveness investigations.

Moreover, several readability index calculations have been introduced allowing the requirement authors to improve their writing style.

In the following subsections, the QUARS EXPRESS functionalities and features are described in more detail.

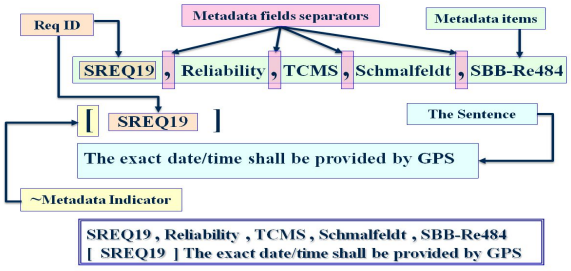

Figure 2: QUARS EXPRESS **exchange data format example**

4.1 Defects Identification

QUARS EXPRESS performs a linguistic analysis of a requirements document and points out the sentences that are defective according to the expressiveness of the quality model described in [4]. This analysis process is split in the two parts, the "lexical" one and the "syntactical" one, described in the following.

Lexical Analysis.

The QUARS EXPRESS lexical analysis domain is based on four linguistic ambiguity classes: *optionality, subjectivity, vagueness* and *weakness*. Each class is identified by a corresponding dictionary that can be tailored according to the application domain of the requirements document. The engine captures defective requirements by identifying candidate defective words belonging to these four defects classes.

Syntactical Analysis.

Similarly, the QUARS EXPRESS syntactical analysis engine captures defective wording belonging to two other linguistic ambiguity classes: *implicity* and *under-specification*. The engine exploits the third-party syntactical parser MINIPAR [18].

Also the identification of *implicity* and *under-specification* defects refers to dictionaries that can be tailored as well, according to the application domain the requirements belong to.

A further analysis is performed taking advantage of the parser output and looking for intrinsic ambiguity relaying on the wrong phrase construction (e.g. the sentence has more than one main verb, subject or object). The errors found belong to the *multiplicity* class that clearly does not have a related dictionary.

False Defects masking/hiding.

Sometimes, detected defects may however be *false defects*. This may occur mainly for three reasons:

- a correct usage of a candidate defective word
- a usage of a candidate defective wording which is not usually considered a defect in the specific system or domain
- a possible source of ambiguity inserted on purpose to give more freedom to implementors.

QUARS EXPRESS provides a simple mechanism to mask to the analysis engines false defective wording. Due to the possibility of handling metadata included in QUARS EXPRESS, the management of *false positive* defects can be done with the granularity of the classification given by metadata. For this reason we have not maintained in QUARS EXPRESS the more refined false positive management implemented in QUARS.

Moreover, defective sentences and their errors are stored for the further metric calculation and to be presented in a well organized form in the final report.

4.2 Metrics and Statistics derivation

With respect to QUARS, QUARS EXPRESS calculates several more metrics (readability indexes and error rates), explained in detail in the following paragraphs.

Readability.

In QUARS EXPRESS, seven readability indexes have been introduced exploiting the GNU program called "Diction/Style" [20]. The Style program analyzes the surface characteristics of the writing style of a document and calculates the values of seven readability indexes (*Kincaid* [15], *ARI* [11], *Coleman-Liau* [13], *Flesh* [14], *FOG* [16], *LIX* [17], *SMOG* [12]), well known in the related research field. These indexes are a mathematical attempt, based on word and syllables count, to point out the minimum US school grade the reader needs to understand the text. As a consequence, there is not an actually *good* value for any of them, but we can assume that technical writings, as requirements documents are, present an unavoidable reading difficulty that leads to scores higher than those presented by common popular writings such as newspapers, novels etc.

The readability analysis scores are shown in each report file for each defective sentence such as the lexical analysis and the syntactic analysis. Moreover the readability scores calculated for every single sentence, even the not defective ones, and for the whole document are reported as well but in separate files.

The purpose of the readability analysis and the underlined techniques are orthogonal with respect to those of the defectivity analysis and hence the produced results are not correlated.

Error Rates.

In QUARS EXPRESS the reckoning of some error rates, basically percentages has been introduced: the *Defect Rate*, the *Analysis Defect Rate* and the *Error Defect Rate*. The same rates are calculated with respect to requirements subsets catalogued by means of metadata fields and their values. Moreover, all the defect rates are calculated with respect to both general analysis results and to any single chosen kind of analysis.

The *Defect Rate* is the ratio between the number of requirements with at least one defect and the total number of analyzed requirements.

Analysis Statistics

Analyzed Requirements	Defective Requirements	Errors	Defect Rate*
793	292	573	37%

* The number of sentences found in the document with at least an error (defective sentences) **divided by** the number of the analyzed sentences (e.g. all the requirements found in the document)

Defect Rates

Analysis -->	Optionality	Subjectivity	Vagueness	Weakness	Implicity	Multiplicity	Underspecification
Analysis Defect Rate**	1% (5/793)	1% (8/793)	13% (106/793)	1% (11/793)	3% (24/793)	25% (198/793)	3% (21/793)
Error Defect Rate***	1% (5/573)	1% (8/573)	22% (124/573)	2% (14/573)	5% (29/573)	65% (372/573)	4% (21/573)

** The number of sentences with at least an error of the kind of the analysis related item (Optionality, Subjectivity,...) **divided by** the number of defective sentences found in the document

*** The number of related analysis item errors **divided by** the total number of errors found in the document

Figure 3: Analysis Statistics

The *Analysis Defect Rate* is the ratio between the number of requirements with at least one defect of a certain class and by the number of defective requirements found in the document.

The *Error Defect Rate* is the ratio of defects of the certain class to the total number of defects found.

These rates single out the amount of a certain class of errors with respect to the whole requirements document (Defect Rate), the performed analysis (Analysis Defect Rate) and the subset of errors only (Error Defect Rate).

Figure 3 is an example of the rates as they are shown in the Report pages.

4.3 QuARS Express Report Structure

As an evolution of the simple text-based report made by QUARS, the new report format exploits the HTML technology to produce well organized and structured hypertextual pages by means of the values of the metadata fields . Unfortunately, as the number of values assumed by the metadata fields increases, the information grow consequently, as well the size of the report and its number of pages.

The report structure is formed by a main directory with five subdirectories and several pages. The main directory contains general report files showing the analysis performed on the whole document and giving a general idea of the defects distribution by means of concise overview tables and global statistics. Explanations about how the tool works and about the meaning of the various analysis performed, the statistics calculated and the readability indexes formulas utilized are provided as well. There is a subdirectory for every metadata field with inside an html page for every of its value. Each one of these html pages gives a projection of the performed analysis over the subset of the requirements filtered by means of the related metadata field. All the HTML pages are dynamically produced following a common structure: the header, the *Table of Contents* and the analysis results are organized in tables providing hypertext links in order to easily jump from a detailed point of view to a more general one and vice versa. Where possible, there are even links redirecting to other pages of the same report and their subsections.

5. QUALITY ANALYSIS PROCESS

The overall Quality Analysis Process adopted in the project is depicted in Figure 4 and is summarized in the following:

- ***(a)*** The partners of the project create a new file project in RequisitePro [21] and insert the requirements with all the required attributes (Name, Text, Responsibility, Package, etc.).
- ***(b)*** The different requirements are stored in a Requirements File, one for each requirement class.

- ***(c)*** At this point, in an automatic way, the tool SoDA [22] generates a text document containing the requirements and the relevant attributes, and saves it in **text** format (alternative formats are *DOC*, *HTML* and *XML*). A specific template has been defined for SoDA in order to allow QUARS EXPRESS to properly interpret the information contained in the generated document.

- ***(d)*** The obtained **text** file is input to QUARS EXPRESS that analyzes the sentences (requirements) and gives as output the Defects Requirement Reports (DRR), for both FREQ and SREQ documents, together with the calculation of relevant metrics.

- ***(e)*** In the case QUARS EXPRESS points to some defects, a refinement activity is needed, possibly followed by another quality analysis step. The DRR should be filtered by experts, in a "*false defect survey*"

- ***(f)***, in order to establish whether a refinement is really necessary or not.

- ***(g)*** Otherwise, the approved requirements document is released.

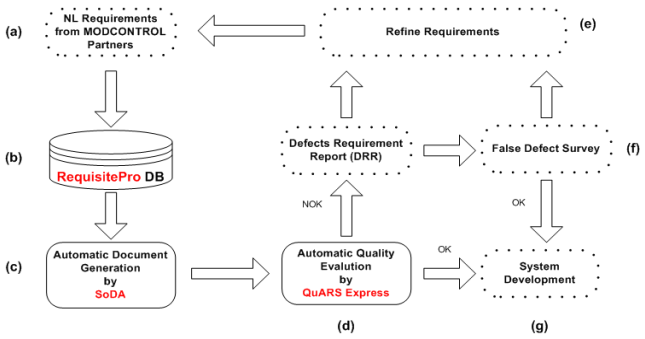

Figure 4: Evaluation Process

6. THE RESULTS OF THE ANALYSIS OF MODCONTROL REQUIREMENTS

In the MODCONTROL project, we have analyzed by means of QUARS EXPRESS the whole set of produced requirements, that is the SREQ and FREQ documents. The results of the analysis have shown that the underlying process not only can be able to point out linguistic defects, but can provide also some indications on the writing style of different requirements authors (from different partners), giving them the opportunity to become aware of defects and of potential improvements. In particular, it has been noted that a requirement author is inclined to repeat the same type of mistakes, unless becoming aware of them. In Figure 5 we can see the number of requirements (SREQ or FREQ) written by the partners (**A**,**B**,**C**, and **Others** for the requirements that have been recorded without the author indication): by the way, the figures tell that project partner **B** has had apparently more responsibility on system requirements, while **C** has had more responsibility on functional requirements.

In Table 1 and 2 we can see the number of defective requirements and the *"Defect Rate"* associated to each partner of the project after the QUARS EXPRESS application on SREQ and FREQ documents. These numbers, once false defects have been filtered out,

Partners	SREQ	(%)	FREQ	(%)
A	710	28	388	12
B	1153	45	596	19
C	208	8	1803	56
Others	497	19	422	13

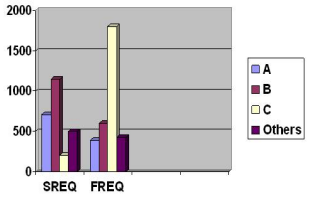

Figure 5: Requirements for each partner

can give an indication on which partner can be considered less accurate in the process of writing requirements. Another important information is on what type of defects is more often introduced in the writing.

Partners	Analyzed	Defective	Errors	Defect Rate(%)
A	388	238	585	61
B	596	296	558	50
C	1803	1046	2516	58
Others	422	67	136	15
Total	**3209**	**1647**	**3795**	**51**

Table 1: FREQ: Defect Rate and Errors

Partners	Analyzed	Defective	Errors	Defect Rate(%)
A	710	353	900	61
B	1153	524	998	45
C	208	46	88	22
Others	497	356	836	72
Total	**2568**	**1282**	**2822**	**50**

Table 2: SREQ: Defect Rate and Errors

In Table 3 and 4 we can notice that *multiplicity* and *vagueness* are more frequent. Table 5 gives some results about the execution time needed to perform the described analysis over such large documents. The differences in the execution speed between FREQ and SREQ depend on the text length of each requirement. SREQ requirements tend to be more concise than FREQ ones: apparently, describing functions requires more verbosity.
The last analysis performed is the Readability Analysis. Table 6 shows the readability average scores of the two documents, FREQ and SREQ.
Note that the SREQ document results to be more readable than the FREQ one. In fact, the indexes values of the SREQ document stand in reasonable ranges according to their technical nature, whereas the scores of the FREQ document are higher than we expected.
Indeed, values of the Kincaid, ARI, Coleman-Liau, FOG, SMOG indexes higher than 15, of the LIX index higher than 58, and of the Flesh index lower than 60 give the indication of a hardly readable document. In our case FREQ exceeds most of such indexes, and it is close to the limits for the other ones: though this is not a dramatic defect, it is advisable to improve the readability of functional requirements, for example shortening phrases and splitting paragraphs.

6.1 Review Process

In MODCONTROL, after the first evaluation process execution, the partners have been invited not only to correct defects, but also to

Analysis	Defects	%	Errors	%
Optionality	35	2	47	1
Subjectivity	39	2	54	1
Vagueness	353	22	652	18
Weakness	128	8	164	4
Implicitly	116	7	251	7
Multiplicity	847	51	2437	64
Underspecification	129	8	190	5

Table 3: FREQ: Defects for Type

Analysis	Defects	%	Errors	%
Optionality	23	2	29	1
Subjectivity	39	3	61	2
Vagueness	396	31	613	22
Weakness	54	4	61	2
Implicitly	66	5	129	5
Multiplicity	633	49	1809	64
Underspecification	68	6	120	4

Table 4: SREQ: Defects for Type

return knowledge about false defects. We have hence indeed identified some typical sources of false defects (*false positives*), such as:

- Words usually indicating vagueness are used to allow for implementation freedom by the manufacturers, in order not to dictate early implementation choices.
- Sometimes the use of passive voice, in verbs, can be deliberately chosen by authors not to address a specific subject for a specific requirement. But in such cases, a discussion of that requirement among experts is useful to clarify the intended meaning of the requirement.
- Some defects are originating from previous guidelines as norms, which are taken as they are.

Consider these examples of false defects, taken from the requirements related to the lighting systems:

- **FREQ2349:***... lighting shall provide a comfortable and pleasing visual environment*.

  In this case the judgment about a "comfortable" and "pleasing" (two vague words) lighting level for passengers is left to the manufacturers, which will follow also marketing criteria. Anyway, this requirement is derived from European guidelines, and hence it has been imported as it was.

- **FREQ2351:** *The emergency lighting shall be sufficient to enable continued occupation or safe egress from the vehicle*.

  In this case the vague word "sufficient" is indeed weak, since a standard is expected to predicate more precisely about emergency issues. However, this text is taken as it is from the same European guidelines.

- **FREQ1760:** *The emergency lighting system shall provide a suitable lighting level in the passenger and in the service areas of at least 5 lux at floor level* .

  In this case, the vague word "suitable" is indeed a vagueness defect, but we can note that the lighting level is specified in the next line: this is actually a redundant requirement, that should be better written as:

  The emergency lighting system shall provide, in the passenger and in the service areas, a lighting level of at least 5 lux at floor level.

Doc.	N. Req.	Time(min)	Speed(Req/min)
FREQ	3209	210	15.28
SREQ	2568	52.8	48.63

Table 5: Execution Time of Analysis

Readability Index	FREQ Scores	SREQ Scores
Kincaid	13.5	7.4
ARI	15.6	7.6
Coleman Liau	14.2	13
Flesch Index	44.8/100	63.4/100
Fog Index	16.8	10.4
LIX	56.5	40.7
SMOG-Grading	14.2	10.1

Table 6: Readability Analysis Results

These examples show that for the detection of most false defects the domain knowledge of the experts who have written the requirements is needed. Collecting the feedback from experts on false defects, we will be able to tune the tool in order to diminish the false defects percentage. Actually, in MODCONTROL this collection has been performed point-wise, and no systematic means to collect feedbacks, and hence to measure the false defect rate, was established. No systematic analysis of *false negatives*, that is, actual defects in the requirements which have not been detected by the tool, has been attempted either. This is a touchy issue: actually, it is obvious that, for the same very nature of the tool, QuARS is not able to detect many categories of defects, and in particular those that require, to be detected, a complex syntactic analysis, rather than the mostly lexical one performed by the tool. In [5, 7] it has been estimated that only 37% of defects in requirements are in categories detectable by QuARS, but the tool is able to detect all the defects in such categories. This implies that developers should not assume an over-confidence on the results of the tools. Although a more systematic analysis of these issues has not been conducted within the MODCONTROL project, the application of QUARS EXPRESS to check the quality of the TCMS requirements has been appreciated as an added means to consolidate the results of the project.

7. LESSONS LEARNED AND FUTURE DIRECTIONS

The concurrent and distributed nature of the MODCONTROL project, which include both physical meetings and meeting by means of groupware systems like electronic email, has demanded the necessity of a quality evaluation process of the requirements document. In fact in MODCONTROL, the objective has been the quality evaluation of the merged requirements document from different partners in order to have the requirements document of the system no ambiguous and readable. At the end of our experience in this project we can present the key points that have emerged after the application of the evaluation process.

- **Process automation and learning phase**: the evaluation

process introduced in figure 4 is very simple to use and has a high degree of automation. The unique things that are demanded to the customer are the learning of the tools, the insertion of the requirements in the database in a well defined way and a definition of a SoDA template in order to generate in automatic way the requirements document that must be analyze from QuARS.

- **Scalability:** Previous experiences with the QUARS tool reported in the literature [3] appear to have been limited to documents with a few hundreds of requirements. Our experience has shown that QuARS EXPRESS easily scales up of an order of magnitude, due to the linear complexity of the analysis algorithms.

The results of the discussed experience are promising. The possibility to have requirements evaluated has been very well appreciated inside the project, thus enabling requirements authors to improve style and precision in the next step of refinement of the collection of requirements. The availability of analysis tools based on NL processing techniques has been crucial for the achievement of this goal.

Several directions for further developing the technique and the tool can be considered, starting form the most ambitious ones to the technical evolution of the tool.

In the first category, we consider the important issue of *semantic consistency* among requirements coming from different sources. This issue has been for example discussed in [8], but is currently not addressed by QUARS EXPRESS. On the other hand, in the context of MODCONTROL project semantic consistency has been addressed by separation of concerns, giving responsibility of each different function or subsystem to one partner.

A useful evolution of the tool is the round-trip integration with RequisitePro (or other requirements management tools), that is, the ability to modify the requirements directly on the output report and having them automatically updated in the database.

Although the metadata defined for MODCONTROL are quite generic and not strictly domain-related, different projects may need different metadata definition. Making the tool easily customizable on the reference metadata is another feature that would contribute to increase its industrial application.

8. ACKNOWLEDGMENTS

This work has been partially supported by the European project MODTRAIN (FP6-PLT-506652/TIP3-CT-2003-506652) subproject MODCONTROL and by the Italian project TOCAI.it.

Moreover, authors thank Andreas Winzen of Siemens AG, Erlangen, Germany, for his valuable insights about this work and Isabella Biscoglio and Giuseppe Lami for their support.

9. REFERENCES

[1] V. Ambriola and V. Gervasi. *Experiences with Domain-Based Parsing of Natural Language Requirements*, Proc. 4 th International Conference NLDB '99, Klagenfurt, Austria, 1999.

[2] D. Berry, E. Kamsties, *Ambiguity in requirements specification*, In: Perspectives on Requirements Engineering, Kluwer (2004) 7-44

[3] A. Bucchiarone, S. Gnesi and P. Pierini. *A Quality Analysis of NL Requirements: An Industrial Case Study*, Proc. of the 13th IEEE International Requirements Engineering Conference, Paris, France, (RE 2005).

[4] S. Gnesi, G. Lami, G. Trentanni, F. Fabbrini and M. Fusani. *An automatic tool for the analysis of application of Natural Language Requirements*, Computer Systems Science and Engineering Vol. 20, N. 1, pp 53-62, CRL Publishing 2005.

[5] G. Lami and R. W. Ferguson. *An empirical study on the impact of automation on the requirements analysis process*, In: Journal of Computer Science and Technology, vol. 22 (3) pp. 338 - 347. Springer, 2007.

[6] L. Mich and R. Garigliano. *Ambiguity Measures in Requirement Engineering*, International Conference on Software Theory and Practice. ICS 2000, Beijing, China.

[7] D. M. Raffo, R. Ferguson, S. Setamanit, B. D. Sethanandha Evaluating the Impact of the QuARS Requirements Analysis Tool Using Simulation in LNCS vol. 4470 "Software Process Dynamics and Agility" pages 307-319 - Springer, July 2007

[8] M. Sabetzadeh and S. M. Easterbrook. *An Algebraic Framework for Merging Incomplete and Inconsistent Views*. Proc. of the 13th IEEE International Requirements Engineering Conference, Paris, France, (RE 2005).

[9] R. H. Thayer and M. Dorfman. *IEEE Software Requirements Engineering*, Second Edition, IEEE Computer Society, New York, NY. 1997.

[10] W. M. Wilson , L. H. Rosemberg and L. E. Hyatt. *Automated Analysys of Requirement Specifications*, Proceedings of the 19th International Conference on Software Engineering (ICSE-97), Boston, MA, May 1997.

[11] E. A. Smith, R. J. Senter. *Automated Readability Index (ARI)* Wright-Patterson AFB, OH: Aerospace Medical Division. AMRL-TR, 66-22, 1967.

[12] H. McLaughlin. *SMOG grading - a new readability formula*, Journal of Reading, 22, 639-646, 1969.

[13] M. Coleman and T.L. Liau. *A Computer readability formula designed for machine scoring*, Journal of Application Psychology, 60, 283-284, 1975.

[14] R. Flesch. *How to Write Plain English*, HarperCollins - 1st edition (August 1979)

[15] J. P. Kincaid, R. P. Fishburne, R. L. Rogers and B. S. Chissom. *Derivation of new readability formulas for navy enlisted personnel.* (1975).

[16] R. Gunning. *The Technique of Clear Writing.* McGraw-Hill New York, 1952

[17] J. Anderson. *Analysing the Readability of English and Non-English Texts in the Classroom with Lix* Paper presented at the Australian Reading Association Conference, Darwin, August (ED 207 022).

[18] http://www.cs.ualberta.ca/ lindek/minipar.htm

[19] X. Tran, J. Kasser. Improving the Recognition and Correction of Poor Requirements. Systems Engineering , Test And Evaluation Conference, SETE 2005 Ű A Decade of Growth and Beyond - Brisbane, Queensland.

[20] Diction/Style reference site. See: http://www.gnu.org/software/diction

[21] IBM Rational RequisitePro. http://www-306.ibm.com/software/awdtools/reqpro/.

[22] IBM Rational SoDA. http://www-306.ibm.com/software/awdtools/soda/.

[23] MODTRAIN: Innovative Modular Vehicle Concepts for an Integrated European Railway System. See: http://www.modtrain.com/.