EVALUATION OF NATURAL LANGUAGE REQUIREMENTS IN AN INDUSTRIAL CASE STUDY

ANTONIO BUCCHIARONE,[*] STEFANIA GNESI AND GIANLUCA TRENTANNI

Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo" (ISTI-CNR)
Area della Ricerca CNR di Pisa, Via G. Moruzzi 1
Pisa, Italy 56124
[*antonio.bucchiarone,stefania.gnesi,gianluca.trentanni*]*@isti.cnr.it*

ALESSANDRO FANTECHI

Dipartimento di Sistemi e Informatica, Universita' degli studi di Firenze
Firenze, Italy 50139
fantechi@di.unifi.it

Automatic evaluation of natural language requirements documents has been proposed as a means to improve the quality of the system under development. QuARS (Quality Analyzer for Requirements Specifications) was introduced as an automatic analyzer of such requirement documents, developed by ISTI-CNR. In this paper we show how this tool has been applied to a large collection of requirements produced inside the EU/IP MODTRAIN project. In this project several industrial partners have contributed to the collection by proposing requirements coming from their own products, with the purpose to define an European standard for the Train Control and Monitoring Systems (TCMS). To better fit the project needs, QuARS has been extended to be interfaced with the requirement management tool adopted in the project, and adapted to specific and domain dependent aspects of the MODCONTROL requirements. In particular the tool is now able to handle a more complex and structured data format containing metadata. The calculation of readability metrics and other statistical indexes has also been improved.

1. INTRODUCTION

The achievement of the quality of system and software requirements is the first step towards the development of a reliable and dependable product. It is well known that inaccuracies in requirements documents could determine serious problems at all the subsequent phases of system and software development. The availability of methods and tools for the analysis of requirements may improve the effectiveness of the requirement process and the quality of the final product. In particular the availability of automatic tools for the quality analysis of Natural Language (NL) requirements [12] is recognized as a key factor.
QuARS (Quality Analyzer for Requirements Specifications) [23] was introduced as an automatic analyzer of such requirement documents, developed by ISTI-CNR. It performs a lexical analysis of documents detecting and possibly correcting ambigu-

[*]IMT Graduate School, Piazza S. Ponziano, 6 - 55100 Lucca , Italy.

ous terms or wordings. In this paper we show how this tool has been extended to be applied to the analysis of a large collection of NL requirements produced inside the EU/IP MODTRAIN project, subproject MODCONTROL [11].
MODCONTROL addresses the standardization of an innovative Train Control and Monitoring System (TCMS) for the future interoperable European trains. The TCMS controls and monitors the various subsystems of a train, providing the necessary information to the driver. In the specification phase for TCMS, project partners have gathered requirements from different existing sources. These requirements had to be consolidated, harmonized and refined among the various project partners. As can be expected, the natural language expression of the requirements tended to exhibit very different styles and quality, often related to a non uniform use of natural language sentences. Consequently, a set of writing rules were enforced in the project and they were (almost always) followed when providing new requirements by the various partners. However, an analysis of the natural language requirements by automatic tools has been considered as an added value for guaranteeing the successful outcome of the project, due to the capability to point out potential sources of ambiguity and other weaknesses.
TCMS requirements have been stored in a single repository, associating to each requirement some metadata attributes to provide several notions of traceability (to the author, to the package, and so on). In order to be able to use QuARS on the TCMS requirements it was necessary interface it with the repository. A new version of the QuARS tool (QuARS Express) has therefore been developed for the MODCONTROL project, to address these needs. In particular the tool is now able to handle a more complex and structured data format containing metadata. QuARS Express produces an analysis report rich of categorized information. The information grows in function of the number of metadata items available (e.g. in function of the number of authors, the number of packages and so on) and the size of the report grows consequently and can be made 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 extended to cope with the needs of the MODCONTROL project. Is section 5 we present the results of the analysis of MODCONTROL requirements, while in section 6 we discuss the experience made and we envisage some future improvement directions.

2. MODCONTROL TCMS CASE STUDY

The MODCONTROL project [11] addresses the standardization of an innovative Train Control and Monitoring System (TCMS) for the future interoperable European trains. The TCMS controls and monitors the various subsystems of a train, providing the necessary information to the driver. It also performs other integrational tasks like allowing trainwide diagnosis and maintenance. TCMS can be considered as the "neural backbone" of the train and has well-identified own characteristics, satisfying a specific set of requirements, and clear interfaces to other on-board subsystems and ground facilities. A key objective of MODCONTROL is the standardization of TCMS functional modules and their interfaces with other subsystems on-board and external to the train.

MODCONTROL approach is to elaborate a generic 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 Control, Air Conditioning, Doors, Brakes or Auxiliary Power Distribution. During MODCONTROL's specification phase, project partners gather requirements from different sources such as specifications of existing trains, standards or drafted specifications from other EU projects. These requirements are then consolidated, harmonized and refined among the project partners in several review sessions.

For the production of harmonized and consistent FRS and SRS, the collection of requirements into a common, project-wide structure is essential.

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 subdivided in the following way:

- Functional Requirements (**FREQ**): Requirements for a TCMS function.
- System Requirements (**SREQ**): Requirements for devices carrying some functions (or sub-functions).
- Glossary Items (**TERM**): Identifies all glossary items within the project.
- Use Cases (**UC**): Description of use cases in the project.

3. NATURAL LANGUAGE REQUIRMENTS ANALYSIS

A NL requirements document, merged from 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 at removing as many bad readability and ambiguity problems 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 them, QuARS [4,3,23], (see the next subsection) and ARM [8,25] perform a lexical analysis of documents detecting and possibly correcting ambiguous terms or wordings, while tools such as LOLITA [9] and Circe-Cico [1] exploit syntactic analyzers to detect ambiguous sentences having different interpretations.

3.1. QuARS

In the context of MODCONTROL the tool **QuARS** (Quality Analyzer for Requirements Specifications) [4,3,23] has been initially chosen for the evaluation of the TCMS requirements document. QuARS has been used in several projects as detailed in Figure 1. QuARS performs an initial parsing of the requirements for automatic detection of potential linguistic defects that can determine ambiguity problems impacting the following development stages.

Applications	**Case Studies**	**Req.**	**References**
`Business`	Functional Requirements of a Transaction and Customer Service (TACS) Check Cashing module	265	4
`Space Software`	Functional Requirements of a sub-system of a space vehicle	444	4
`Telecommunication`	Requirements Specification of a project aiming for a new generation STM switches and SDH telecommunication equipments	2067	4,2
`Security`	Functional Security Requirements for an Application Level Firewall Protection Profile	119	4,5

Table 1. NL Requirements Document Analysis Experiences

The functionalities provided by QuARS are:

- Defect Identification: QuARS performs a linguistic analysis of a requirements document in plain text format and points out the sentences that are defective according to the expressiveness quality model described in [4,3].

The defect identification process is split in two parts: (i) the "lexical analysis" capturing *optionality, subjectivity, vagueness and weakness* defects, by identifying candidate defective words; and (ii) the "syntactical analysis" capturing *implicitly, multiplicity and under-specification* defects. 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 and a possible source of ambiguity inserted on purpose to give more freedom to implementors.

- Requirements clustering: The capability to handle collections of requirements, i.e. the capability to highlight clusters of requirements holding specific properties, can facilitate the work of the requirements engineers.
- Metrics derivation: Metrics have been defined in QuARS for evaluating the quality of NL requirements document with respect to measures on the readability of the document plus measures on the percentage of defects throughout the whole document. Among the metrics calculated by QuARS, we cite the readability index and the defect rate. The readability index is given by the Coleman-Liau Formula [16]. The reference value of this formula for an easy-to-read technical document is 10, if it is greater than 15 the document is considered difficult to read. The defect rate is the percentage ratio calculation of defects distribution in the document with respect to the performed kind of analysis.

4. ANALYSIS NEEDS OF THE PROJECT

As said previously, MODCONTROL aims to produce the Requirements Specifications (FRS and SRS) for a new generation of train control systems. It is therefore evident that the produced specifications should not be possibly misinterpreted due to weaknesses and ambiguities in the NL requirements for TCMS. An added difficulty from this point of view was the fact that requirements have been produced by several partners, and then merged in a single repository. A set of writing rules were enforced in the project [10], and they were (almost always) followed when inserting new requirements by the various partners. However, an analysis of the natural language text by automatic tools has been considered an added value to guarantee the quality of the requirements document, due to the capability to point out potential sources of ambiguity and other weaknesses.

Each requirement stored in the repository is constituted by several attributes, one of which is the text: once a weakness is pointed in the text, the other attributes are useful to trace the origin or responsibility of such potential weakness: the *Source* attribute tells from which previous product requirements document, if any, the requirement derives; the *Responsibility* attribute refers to the person who has actually inserted the requirement in the repository, the *Package* attribute tells which part of the system the requirement refers to, finally *Type* attribute describes the belonging of the requirement to a certain category (i.e., Functional, Architectural, Performance, Realtime, etc.). Using the Responsibility attribute, it is possible therefore to ask an individual for the resolution of the potential weakness, either by correction of the requirements, or by recognition of a so called *false defect*. The *Source* attribute may instead reveal that a defective requirement has actually been borrowed from some standard and hence it cannot be resolved if not issuing a standard change request.

The QuARS tool was developed to deal with pure text, in various formats. In order to associate each weakness found in the NL text to the relevant attributes, the tool needed to be interfaced with the repository. A new version of QuARS (QuARS Express) has therefore been developed for the MODCONTROL project, to address these needs.

5. QuARS Express

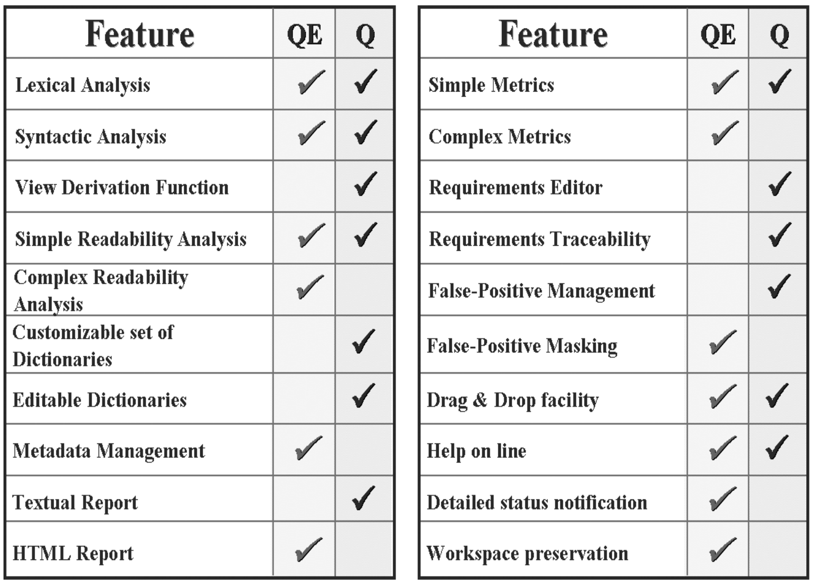
The new version of QuARS exploits the same core engine of QuARS, but the huge number of requirements to be analyzed claimed the availability of a simpler tool producing rich reports and able to manage a minimum metadata set.

To address these needs, a new GUI has been developed allowing the user to perform the time-consuming analysis in a click.
Exploiting the RequisitePro plugin called SoDA [22], a text format has been established to handle five metadata fields: a requirement unique ID, the *Responsibility*, the *Type*, the *Source*, and the *Package*.
Any requirement is traceable by means of at least one of its five metadata fields and the produced report is tailored to be used both for analysis and correction purposes, or for productiveness investigations.
The readability analysis has been introduced allowing the requirement authors to improve their writing style.
Finally, the set of metrics has been enriched adding statistics both on the whole document and on requirements subsets singled out by means of metadata fields.

Figure 2 shows a feature based comparison between QuARS and QuARS Express. Although most of the features are shared, it is clear that the two tools are complementary rather than one the extension of the other.

In the following, the QuARS Express features are described more in detail.

- *Defect Identification.* As we already said, QuARS Express shares with QuARS the core analysis engine and produces the same analysis results. These are based on the same quality model, and divided in lexical analysis, capturing *optionality, subjectivity, vagueness* and *weakness* defects, and syntactic analysis, capturing *implicitly, multiplicity* and *under-specification* defects, as well.
- *Readability Analysis.* In QuARS Express, seven readability indexes has



Feature	QE	Q	Feature	QE	Q
Lexical Analysis	✓	✓	Simple Metrics	✓	✓
Syntactic Analysis	✓	✓	Complex Metrics	✓	
View Derivation Function		✓	Requirements Editor		✓
Simple Readability Analysis	✓	✓	Requirements Traceability		✓
Complex Readability Analysis	✓		False-Positive Management		✓
Customizable set of Dictionaries		✓	False-Positive Masking	✓	
Editable Dictionaries		✓	Drag & Drop facility	✓	✓
Metadata Management	✓		Help on line	✓	✓
Textual Report		✓	Detailed status notification	✓	
HTML Report	✓		Workspace preservation	✓	

Fig. 1. QuARS vs QuARS Express

been introduced. This new feature exploits the GNU program called "Diction/Style" [13]. The Style program analyzes the surface characteristics of the writing style of a document and calculates the values of seven readability indexes well known in the readability research field: *Kincaid*[18], *ARI* [14], *Coleman-Liau*[16], *Flesh*[17], *FOG*[19], *LIX*[20], *SMOG*[15].
These readability 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 isn't 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 synyactic analysis. Moreover the readability scores calculated for all the sentences, even the not defective ones, and for the whole document are reported as well but in separate files.

- *Metrics and Statistics derivation.* The set of metrics has been enriched with the *analysis defect rate* and *error defect rate*, explained in detail in the following.
  - *Defect Rate.* It is the percentage ratio between the number of sentences with at least a defect and the total number of analyzed sentences.

Moreover, the same ratio is calculated with respect to requirements subsets cataloged by metadata fields.

– *Analysis Defect Rate.* Related to the specific analysis performed (e.g. *Optionality*, *Subjectivity*, *Vagueness*, *Weakness*, *Implicity*, *Multiplicity*, *Underspecification*), it is the percentage ratio between the number of requirements with at least a defect of the related type divided by the number of defective requirements found in the document.
  The same ratio is calculated with respect to requirements subsets belonging to metadata fields as well.
– *Error Defect Rate.* Related to the errors found during the specific kind of analysis performed (e.g.*Optionality*, *Subjectivity*, *Vagueness*, *Weakness*, *Implicity*, *Multiplicity*, *Underspecification*), it is the percentage ratio between the number of defects of the related type and the total number of defects found.
  The same ratio is calculated with respect to requirements subsets belonging to metadata fields as well.

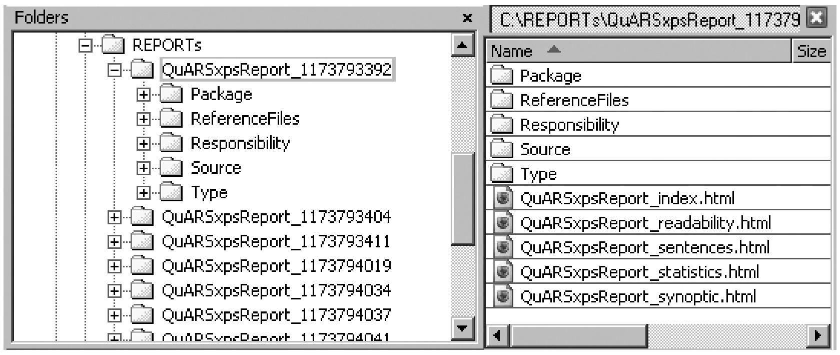
Note that all the defect rates are calculated with respect to both general analysis results, and to any single kind of analysis (e.g. *Optionality*, *Subjectivity*, *Vagueness*, *Weakness*, *Implicity*, *Multiplicity*, *Underspecification*).
Moreover tool separately produces metrics reports based on requirements subsets related to metadata fields (e.g. *Responsibility*, *Type*, *Source* and *Package*).

• *False Defects masking/hiding.* During the evaluation false defects may be detected. QuARS Express provides a simple mechanism to mask to the analysis engine false defective wording. Due to the 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 mainteined in QuARS Express the more refined false positive management implemented in QuARS.

QuARS Express produces an analysis report rich of categorized information. The information grows in function of the number of metadata items available (e.g. in function of the number of authors, the number of packages and so on) and the size of the report grows consequently and can be made of several pages.
As an evolution of the simple text-based report made by QuARS, the new report exploits the html technology to produce structured hypertextual pages. The structure of the analysis report is quite complex and will be explained in detail in the following section.

5.1. QuARS *Express Report Structure*

The analysis report that QuARS Express produces is a collection of HTML pages organized in a main directory and five subdirectories (Figure 3).


```
Folders
  REPORTs
    QuARSxpsReport_1173793392
      Package
      ReferenceFiles
      Responsibility
      Source
      Type
    QuARSxpsReport_1173793404
    QuARSxpsReport_1173793411
    QuARSxpsReport_1173794019
    QuARSxpsReport_1173794034
    QuARSxpsReport_1173794037
```

```
C:\REPORTs\QuARSxpsReport_117379
Name                                  Size
Package
ReferenceFiles
Responsibility
Source
Type
QuARSxpsReport_index.html
QuARSxpsReport_readability.html
QuARSxpsReport_sentences.html
QuARSxpsReport_statistics.html
QuARSxpsReport_synoptic.html
```

Fig. 2. The Report structure

The name of the main directory is formed by the fixed string "*QuARSxpsReport*" followed by the "*ReportID*", a unique report identifier based on the date of generation, that allows users to store several reports without overwriting risks. The main directory contains general report files, the reference files directory and four subdirectories. The general report files show the analysis performed on the whole document and give a general idea of the defects distribution showing concise overview tables and global statistics. The reference files directory contains explanations about how the tool works and about the meaning of the various analysis performed, the statistics calculated and the readability indexes formulas utilized.

The other four subdirectories, namely *Responsibility, Type*, *Source* and *Package*, are the metadata related ones, containing report files about the analysis filtered through the metadata field values. Each of them can contain several HTML files, depending on how many values the specific metadata field contains. Each of these files gives a projection of the performed analysis over the subset of the requirements catalogued by means of the metadata field, hence providing help for traceability with respect to authors, source document, requirement type or originating package.

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 to allow for easily jumping from a detailed point of view to a more general one and vice versa.

6. Quality Analysis Process

The overall Quality Analysis Process adopted in the project is depicted in Figure 3 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, that is, Functional Requirements (FREQ) and System Requirements (SREQ).

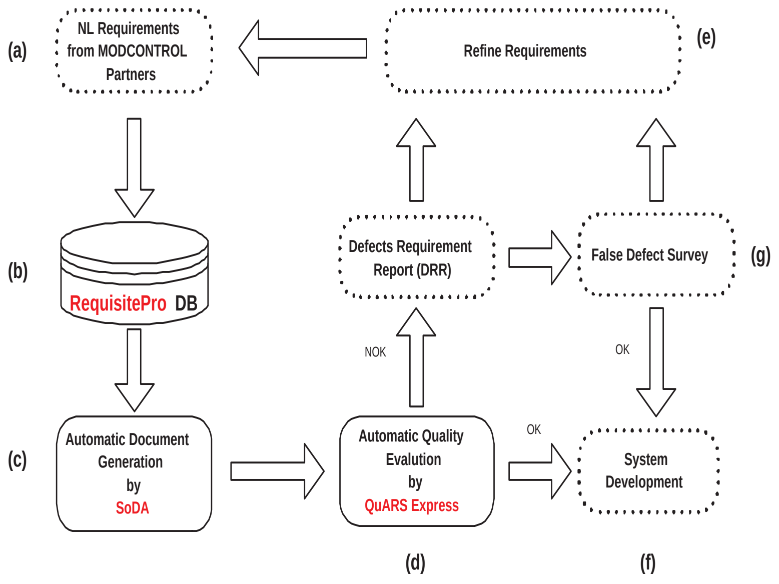

(a)
NL Requirements from MODCONTROL Partners
Refine Requirements
(e)
(b)
RequisitePro DB
Defects Requirement Report (DRR)
False Defect Survey
(g)
NOK
OK
(c)
Automatic Document Generation by SoDA
Automatic Quality Evalution by QuARS Express
OK
System Development
(d)
(f)

Fig. 3. Evaluation Process

(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 with **txt** 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 **txt** 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*", in order to establish whether a refinement is really necessary or not.
(f) Otherwise, the approved requirements document is released.

7. THE RESULTS OF THE ANALYSIS OF MODCONTROL REQUIREMENTS

In MODCONTROL project, we have analyzed by means of QuARS Express the whole set of produced requirements, that is 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 4 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). The project partner **B** has had apparently more responsibility on system requirements, while **C** has had more responsibility on functional requirements.

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

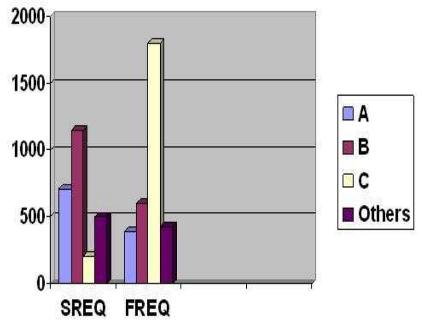


Fig. 4. Requirements for each partner.

In Table 2 and 3 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, 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. In Table 5 and 4 we can notice the *multiplicity* and *vagueness* are more frequent. Table 6 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, depends on the text length for 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. Figure 7 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. The experience leads to advise a readability improvement (i.e. writing simplification) for values of the Kincaid, ARI, Coleman-Liau, FOG, SMOG indexes higher than 15, for values of the LIX index higher than 58, and for values of the

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 2. 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 3. SREQ: Defect Rate and Errors

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 4. FREQ: Defects for Type

Flesh index lower than 60. Though this is not a dramatic defect, it is advisable to improve the readability of functional requirements, for example shortening phrases and splitting paragraphs.

8. CONCLUSIONS AND FUTURE WORK

The concurrent and distributed nature of the MODCONTROL project, which include both physical meetings and meeting by means of groupware system like email, has demanded the necessity of a quality evaluation process of the requirements doc-

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 5. SREQ: Defects for Type

Documents	**Requirements**	**Time (min)**	**Speed (Req/min)**
FREQ	3209	210	15.28
SREQ	2568	52.8	48.63

Table 6. Execution Time of Analysis

Readability Index	**FREQ Readability Scores**	**SREQ Readability 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 7. Readability Analysis Results

ument. In fact in MODCONTROL, the objective has been the quality evaluation of the merged requirements document from different partners in order to have the TCMS requirements document that is unambiguous and readable. We discuss in the following the key points that have emerged after the application of the evaluation process; these key points have differentiated this experience from the previous experiences, referred in Figure 1, in which NL quality analysis was applied to monolithic requirement documents.

- **Process automation and learning phase**: the evaluation process introduced in Figure 3 is very simple to use and has a high degree of automation.

The only things that are demanded to the customer are the learning of the tools, the insertion of the requirements in the database and the definition of a SoDA template in order to generate in automatic way the requirements document that is going to be analyzed by QuARS Express.

- **Scalability:** Table 1 shows that previous experience with QuARS were mostly limited to documents with a few hundreds of requirements. QuARS Express has been shown to easily scaled up of an order of magnitude. Moreover, the documents analyzed for MODCONTROL were not only text lines, but included metadata which have been used for a better and more accurate presentation of results. Another direction of the flexibility of QuARS Express is witnessed by the connection to RequisitePro by means of the SoDA documentation tool; any other tool for requirements specification (e.g. DOORS [24]) is believed to be adoptable as well.
- **Review Process**: in MODCONTROL, after the first evaluation process execution, the partners have been invited not only to correct defects, but also to return knowledge about false defects. We have hence indeed identified some typical sources of false defects, such as:
  - Words usually indicating vagueness are used to allow for implementation freedom by the manufacturers, that is not to impose implementation choices.
  - Sometimes the use of passive voice 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 a 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.

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 tools in order to diminish the false defects percentage. Actually, in MODCONTROL this collection has been performed pointwise, and no systematic means to collect feedbacks, and hence to measure the false defect rate, was established. This is a point to improve in future applications of the approach.

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.

9. Acknowledgments

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

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. 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).
3. 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.
4. F. Fabbrini, M. Fusani, S. Gnesi and G. Lami. *The Linguistic Approach to the Natural Language Requirements Quality: Benefits of the use of an Automatic Tool*, Proc. of 26th NASA Software Engineering Workshop, IEEE November 2001.

5. F. Fabbrini, M. Fusani, S. Gnesi and G. Lami. *An Automatic Quality Evaluation for Natural Language Requirements*, Proc. of 7th Int. Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'01), Interlaken, Switzerland, June 4-5, 2001.
6. R. H. Thayer and M. Dorfman. *IEEE Software Requirements Engineering*, Second Edition, IEEE Computer Society, New York, NY. 1997.
7. 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).
8. 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.
9. L. Mich and R. Garigliano. *Ambiguity Measures in Requirement Engineering*, International Conference on Software Theory and Practice. ICS 2000, Beijing, China.
10. A. Winzen. *Process for MODCONTROL*, Technical Report of the MODTRAIN-MODCONTROL project, 2004.
11. MODTRAIN: Innovative Modular Vehicle Concepts for an Integrated European Railway System. See: http://www.modtrain.com/.
12. D. Berry *Natural language and requirements engineering - nu?*, http://www.ifi.unizh.ch/groups/req/IWRE/papers/Berry.pdf
13. Diction/Style reference site. See: http://www.gnu.org/software/diction
14. E. A. Smith, R. J. Senter. *Automated Readability Index (ARI)* Wright-Patterson AFB, OH: Aerospace Medical Division. AMRL-TR, 66-22, 1967.
15. H. McLaughlin. *SMOG grading - a new readability formula*, Journal of Reading, 22, 639-646, 1969.
16. M. Coleman and T.L. Liau. *A Computer readability formula designed for machine scoring*, Journal of Application Psychology, 60, 283-284, 1975.
17. R. Flesch. *How to Write Plain English*, HarperCollins - 1st edition (August 1979)
18. J. P. Kincaid, R. P. Fishburne, R. L. Rogers and B. S. Chissom. *Derivation of new readability formulas for navy enlisted personnel.* (1975).
19. R. Gunning. *The Technique of Clear Writing.* McGraw-Hill New York, 1952
20. 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).
21. IBM Rational RequisitePro.
    See: http://www-306.ibm.com/software/awdtools/reqpro/.
22. IBM Rational SoDA. See: http://www-306.ibm.com/software/awdtools/soda/.
23. QuARS (Quality Analyzer for Requirements Specification). See: http://www.quars.isti.cnr.it.
24. Telelogic DOORS/ERS. See: http://www.telelogic.com/products/.
25. Automated Requirement Measurement (ARM) Tool. See: http://satc.gsfc.nasa.gov/tools/arm/.