QuARS Express: A Tool for Evaluating Natural Language Requirements

A. Bucchiarone, S. Gnesi, G. Lami and G. Trentanni
Istituto di Scienze e Tecnologie dell'Informazione, CNR, Pisa, Italy
{antonio.bucchiarone,stefania.gnesi,giuseppe.lami,gianluca.trentanni}@isti.cnr.it

A. Fantechi
Dipartimento di Sistemi e Informatica
Universita' degli studi di Firenze, Italy
fantechi@di.unifi.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 in the automatic analysis of a large collection of natural language requirements, produced inside the MODCONTROL project using this tool.*

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 to the subsequent phases of system and software development. The availability of techniques and tools for the analysis of requirements may improve the effectiveness of the requirement process and the quality of the final product. Particularly, the availability of automatic tools for the quality analysis of Natural Language (NL) requirements [13] is recognized as a key factor. QUARS (Quality Analyzer for Requirements Specifications) [24] was introduced as an automatic analyzer of such requirement documents. It performs a lexical analysis of requirements detecting ambiguous terms or wordings. In this paper we introduce QUARS EXPRESS, a modified version of QUARS, specialized to be applied to the analysis of a large collection of NL requirements produced inside the EU/IP MODTRAIN project, subproject MODCONTROL [12]. MODCONTROL addresses the standardization of an innovative Train Control and Monitoring System (TCMS) for the future interoperable European trains. 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. 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 project, due to the capability to point out potential sources of ambiguity and other weaknesses. TCMS requirements have then 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 be able to use QUARS on the TCMS requirements it was necessary to interface it with the repository. A modified version of the QUARS tool (QUARS EXPRESS) has therefore been developed for the MODCONTROL project, to address these needs. 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

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. 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 train-wide diagnosis and maintenance.

MODCONTROL approach is to elaborate a 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 categorized 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.

As previously said, 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 with consequent problems due to heterogeneous writing syles. A set of writing rules were enforced in the project [11] and they were (almost always) followed when inserting new requirements by the various partners. Using RequisitePro [22] , each requirement has been stored in a repository : a requirement is constituted by several attributes, that are:

- *Text:* provides the NL text of a single requirement;
- *Source:* indicates from which previous product requirements document, if any, the requirement derives. It may reveal that a defective requirement has actually been borrowed from some standard and hence it cannot be resolved unless issuing a standard change request;
- *Responsibility:* refers to the person who has actually inserted the requirement in the repository. Using it, is possible therefore to ask an individual for the resolution of potential weakness, either by correction of the requirements, or by recognition of a so called *false defect*;
- *Package:* indicates which part of the system the requirement refers to;
- *Type:* describes the category the requirement belongs to (i.e., Functional, Architectural, Performance, Real-time, etc.).

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, QUARS[4, 3, 24], (see the next subsection) and ARM [9, 26] perform a lexical analysis of documents detecting and possibly correcting ambiguous terms or wordings, while tools such as LOLITA [10] 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) was initially chosen for the evaluation of the TCMS requirements document, since it was used also in previous several projects [2, 5, 6]. 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.

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 that are identified into a corresponding set of dictionaries; and (ii) the "syntactical analysis" capturing *implicitly, multiplicity and under-specification* defects. In the same way, detected defects may however be *false defects*. This may occur mainly for three reasons: (i) a correct usage of a candidate defective word, (ii) a usage of a candidate defective wording which is not usually considered a defect in the specific system or domain, and (iii) a possible source of ambiguity inserted on purpose to give more freedom to implementors. For this reason, a false positive masking feature is provided.

- *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 [17]. 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.

- *View derivation:* A View is a subset of the input requirements document, consisting of those sentences that deal with particular quality attributes or other non-functional aspects of the system. The view deriver identifies and collects together those sentences belonging to a given View. The availability of Views makes the detection of inconsistencies and incompleteness easier because the reviewer only has to consider smaller sets of sentences where possible defects can be found with much less effort.

4 QuARS Express

QUARS EXPRESS exploits the same core engine of QUARS, but the huge number of requirements to be analyzed claimed the availability of a simpler to use tool producing richer reports and able to manage a minimum metadata set. To address these needs, four improvements have been implemented:

- a new graphical user interface has been developed allowing the user to perform the time-consuming analysis in a click (Figure 1);

- since the requirements and the metadata are stored in a repository based on RequisitePro, the tool has been interfaced to it by means of the SoDA plug-in [23]. This has required the definition of a text format that has been established to handle the 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 text format is illustrated in Figure 2 with an example;

- several readability analysis has been introduced allowing the requirement authors to improve their writing style;

- 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 3 shows a feature based comparison between QUARS and QUARS EXPRESS. Although most of the fea-

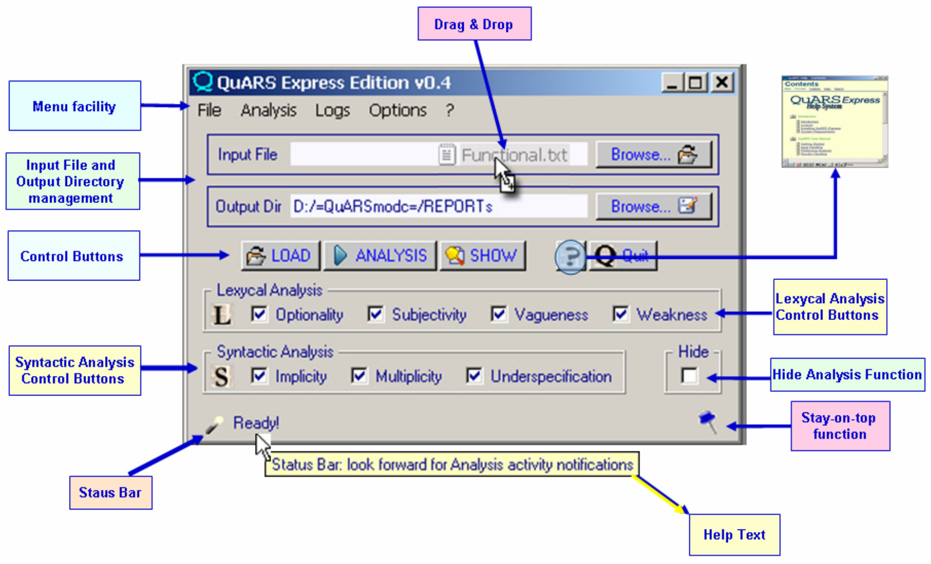
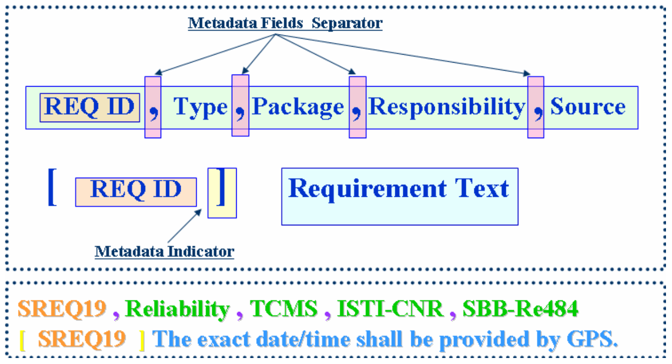


Figure 1. QUARS EXPRESS **Graphical User Interface**



SREQ19 , Reliability , TCMS , ISTI-CNR , SBB-Re484
[SREQ19] The exact date/time shall be provided by GPS.

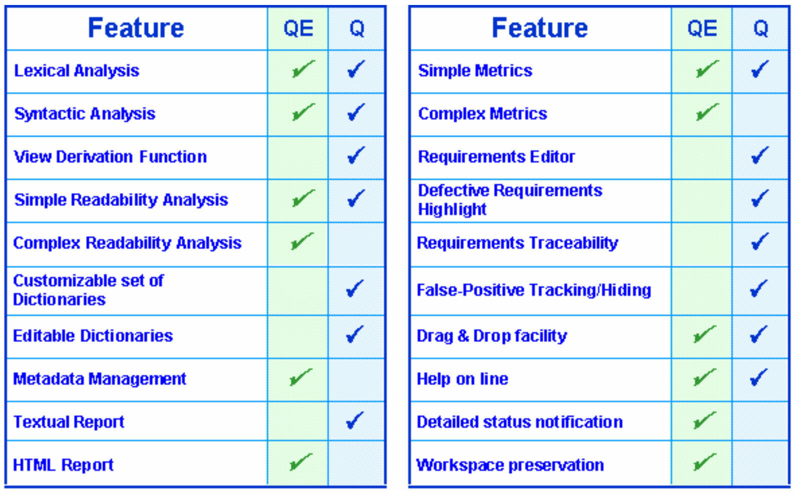
Figure 2. The Interchange Data Format with an example

tures 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 have been introduced. This new feature exploits the GNU program called ”Diction/Style” [14]. 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*[19], *ARI* [15], *Coleman-Liau*[17], *Flesh*[18], *FOG*[20], *LIX*[21], *SMOG*[16].
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 syntactic 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 sep-



Feature	QE	Q
Lexical Analysis	✓	✓
Syntactic Analysis	✓	✓
View Derivation Function		✓
Simple Readability Analysis	✓	✓
Complex Readability Analysis	✓	
Customizable set of Dictionaries		✓
Editable Dictionaries		✓
Metadata Management	✓	
Textual Report		✓
HTML Report	✓	

Feature	QE	Q
Simple Metrics	✓	✓
Complex Metrics	✓	
Requirements Editor		✓
Defective Requirements Highlight		✓
Requirements Traceability		✓
False-Positive Tracking/Hiding		✓
Drag & Drop facility	✓	✓
Help on line	✓	✓
Detailed status notification	✓	
Workspace preservation	✓	

Figure 3. QuARS **vs** QuARS Express

arate 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 requirements with at least a defect and the total number of analyzed requirements. Moreover, the same ratio is calculated with respect to requirements subsets catalogued by metadata fields.

- *Analysis Defect Rate*. It is the percentage ratio between the number of requirements with at least a defect of a chosen type (*Optionality*, *Subjectivity*, *Vagueness*, *Weakness*, *Implicity*, *Multiplicity*, *Underspecification*), 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.

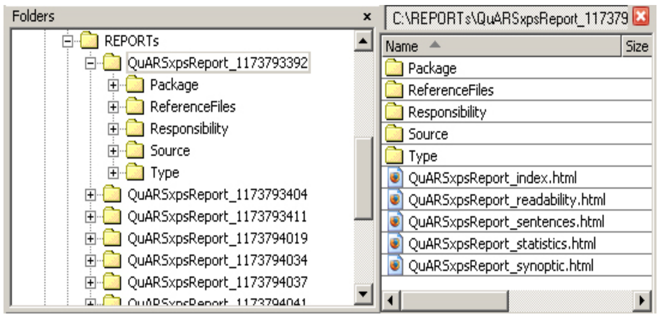
- *Error Defect Rate*. Since more defects can be found in a single requirement, this finer metric gives the percentage ratio of defects of the chosen type and the total number of defects found.

Note that all the defect rates are calculated with respect to both general analysis results, and to any single chosen kind of analysis. Moreover QuARS Express separately produces metrics reports based on requirements subsets related to the 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 maintained in QuARS Express the more refined false positive management implemented in QuARS.

4.1 QuARS Express Report Structure

QuARS Express 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 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, organized in a main directory and five subdirectories (Figure 4).



```
Folders
  REPORTs
    QuARSxpsReport_1173793392
      Package
      ReferenceFiles
      Responsibility
      Source
      Type
    QuARSxpsReport_1173793404
    QuARSxpsReport_1173793411
    QuARSxpsReport_1173794019
    QuARSxpsReport_1173794034
    QuARSxpsReport_1173794037
    QuARSxpsReport_1173794041
```

```
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
```

Figure 4. 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 time of generation, that allows users to store several reports without overwriting risks. The main directory contains general report files, the *ReferenceFiles* directory and four additional 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 *ReferenceFiles* 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.

The HTML pages share one each other common items:

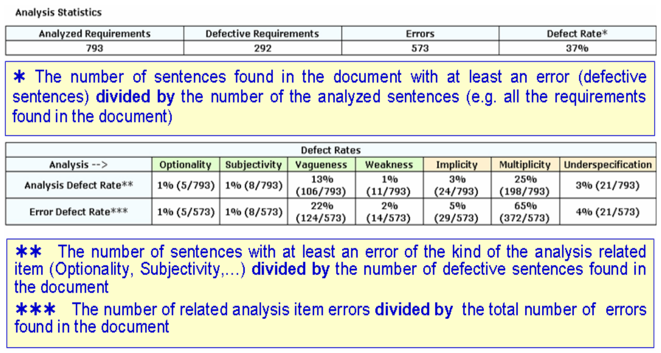
- the *heading* of the file that specifies the path of the analyzed requirements file, the belonging to the metadata item and its name, the session pointed out by the univoque *Report ID* and the *date* of the performed analysis;

- a *Table of the page Contents*: a list of links pointing the related sections of the page;

- the *index* of defective sentences by means of their *ID* where every *ID* is a link pointing to the complete defect description;

- a *synoptic view* of all defective sentences shown as a table, with associated the defect wording and the kind of analysis performed, where exist a link pointing to the complete description (full view) of the requirement analysis.

- some statistics (i.e., Analysis Defect Rate, Error Defect Rate, etc.) related to the whole document. In Figure 5 we show an example of this kind of output.


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 5. Analysis Statistics

5 Quality Analysis Process

The overall Quality Analysis Process adopted in the project is depicted in Figure 6 and is summarized in the following: ***(a)*** The partners of the project create a new file project in RequisitePro [22] 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 [23] generates a text document containing the requirements and the relevant attributes, and saves it in **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*” ***(f)***, in order to establish whether a refinement is really necessary or not.
(g) Otherwise, the approved requirements document is released.

6 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 the SREQ and FREQ documents. The results of the analysis have shown that the underlying process not

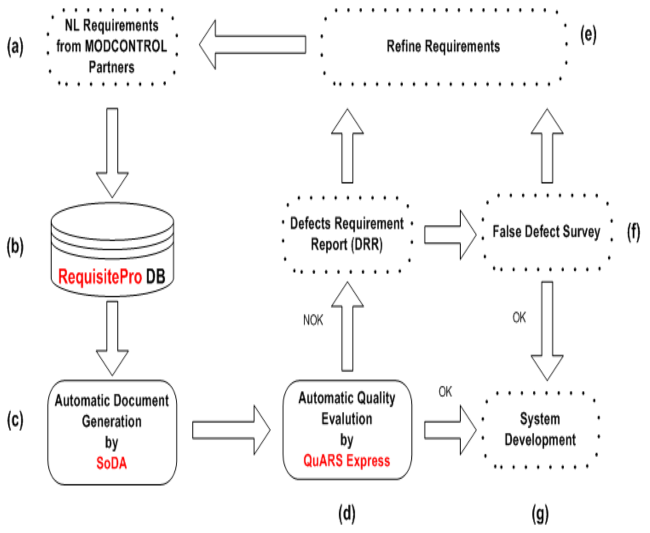
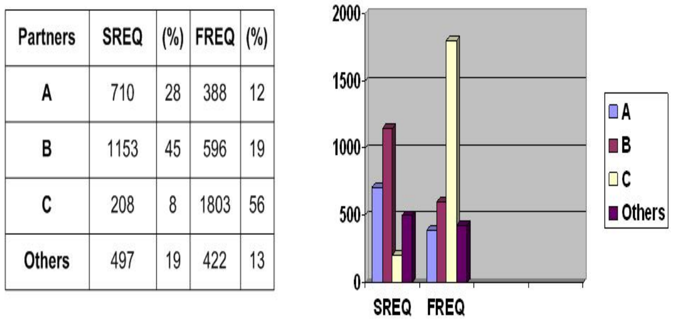

(a)
NL Requirements from MODCONTROL Partners
Refine Requirements
(e)
(b)
RequisitePro DB
Defects Requirement Report (DRR)
False Defect Survey
(f)
NOK
OK
(c)
Automatic Document Generation by SoDA
Automatic Quality Evalution by QuARS Express
OK
System Development
(d)
(g)

Figure 6. Evaluation Process

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 7 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

Figure 7. Requirements for each partner

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, 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 the *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, 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. 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.

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

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

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 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, 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.

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

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.

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 point-wise, 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. Anyway, the application of QuARS Express to check the quality of the requirements has been appreciated at the project level, as an added means to consolidate the results of the project.

7 Conclusions and Future Works

In this paper we have presented the tool QuARS Express aimed at the analysis of Natural language requirements, and we have reported on its application to the analysis of a large set of requirements coming form the MODCONTROL project. We discuss in the following the two key points that have emerged after this experience and the main issue that we would like to consider in the future.

- **Process automation and learning phase**: the evaluation process introduced in Figure 6 is very simple to use and has a high degree of automation. The user is demanded to learn the use of the tools, to insert the requirements in the database and to define the SoDA template in order to generate, in an automatic way, the requirements document that is going to be analyzed by QuARS Express.
- **Scalability:** QuARS Express has been shown to easily scale up by 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 or any other tool able to export in a customized txt format accepted by QuARS Express (e.g. DOORS [25]).

Another important issue in requirements management is *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.

8 Acknowledgments

This work has been partially supported by the European project MODTRAIN (FP6-PLT-506652/TIP3-CT-2003-506652) subproject MODCONTROL. Moreover authors thank Andreas Winzen of Siemens AG, Erlangen, Germany, for his valuable insights about this work.

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] 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.

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

[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] 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.

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

[11] A. Winzen. *Process for MODCONTROL*, Technical Report of the MODTRAIN-MODCONTROL project, 2004.

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

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

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

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

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

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

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

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

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

[21] 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).

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

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

[24] QuARS (Quality Analyzer for Requirements Specification).
http://www.quars.isti.cnr.it.

[25] Telelogic DOORS/ERS.
http://www.telelogic.com/products/.

[26] Automated Requirement Measurement (ARM) Tool.
http://satc.gsfc.nasa.gov/tools/arm/.