A Quality Evaluation Process for Collaborative Requirements: An Industrial Case Study

Antonio Bucchiarone*and Stefania Gnesi
Istituto di Scienza e Tecnologie dell'Informazione
"A. Faedo" (ISTI - CNR)
Area della Ricerca CNR di Pisa, 56100 Pisa, Italy
{antonio.bucchiarone, stefania.gnesi}@isti.cnr.it

Alessandro Fantechi
Dipartimento di Sistemi e Informatica
Universita' degli studi di Firenze
I-50139 Firenze, Italy
fantechi@dsi.unifi.it

Andreas Winzen
Siemens AG, Transportation Systems
D-91052 Erlangen, Germany
andreas.winzen@siemens.com

Abstract

When requirements come from different companies, as it often happens in big industrial projects, it is crucial for the success of the project to offer means and automatic tools for the early validation of the requirements. This paper introduces an approach for quality evaluation of natural language (NL) requirements documents performed with the help of an automatic support tool. The key factors of the approach are the concepts of readability, defects and errors detection in NL requirements documents (RDs) with support of the tools QuARS that exploits for the analysis NL processing techniques. The approach has been successfully applied to the collection of requirements for a Train Control and Monitoring System (TCMS), produced by writing several requirements coming from different people. The application was carried out inside the EU/IP MODTRAIN project, subproject MODCONTROL on the MODTRAIN requirements database containing more than 5500 requirements.

1. Introduction

A requirements document (RD) is used to communicate systems requirements to customers, system users, managers and system developers. Therefore the RD should follow a company-standard structure. Adherence to this structure should be checked as part of RD quality assurance. This quality assurance is particularly important when the requirements for a complex system are written from different sources and collected into a single RD. For example, when competitors agree to harmonize their products according to some interoperability guidelines or standards. In this case, the RD is the reference for interoperability that leaves implementation choices to each implementor.

Quality assurance of a RD is needed to avoid later problems in the system development. In particular a RD written from different sources may suffer differences in style and accuracy, producing an unbalanced and ambiguous final RD. Several approaches can be followed to ensure a good quality RD. One approach is a linguistic analysis of a NL RD aimed at removing as many how bad readability and ambiguity problems as possible.

Several natural language processing (NLP) tools have been applied to software RDs for consistency and completeness verification. Typically a tool reveals sentences having lexical or syntactical defects. Among them lexical tools such as QuARS [14, 5, 4] and ARM [10, 9] detect and possibly correct ambiguous terms or wordings, while syntactic tools such as LOLITA [6] and Circe-Cico [1] exploit syntactics analyzers to detect ambiguous sentences having different interpretations, as in the case of the tools .

This paper focuses on linguistic sources of problems. We applied the tool QuARS in order to evaluate the System Requirements Document for a Train Control and Monitoring System (TCMS) produced by collecting several requirements coming from different people. The experience was carried out in the EU/IP MODTRAIN project [13], subproject MODCONTROL, on the MODTRAIN requirements database containing more than 5500 requirements.

Section 2 introduces the activity of collaborative requirements writing, Section 3 describes the case study, Section 4

*IMT (Institutions Markets Technologies) Graduate School, Via San Micheletto, 3 - 55100 Lucca

describes the overall quality evaluation process, while Section 5 describes the results of the approach application.

2 Collaborative Requirements Management

Requirements management is often difficult and tedious, especially when several stakeholders are involved in the definition of RD for large projects. Indeed, the key for a successful project is having an accurate and thorough collaborative requirements management process. There are many different influences on the requirements for a system. These include end-users who are involved in the process which the system is designed to support, managers and others in the organization whose work is indirectly affected by the system, and customers of the organization buying the system. All of these and others are potential sources of system requirements. Collecting requirements from different sources and merging them into a single requirements document is a useful way of collaborative requirement management. The principal concerns of a collaborative requirements management are:

1. Each requirement should be assigned a *unique identifier* or reference number which may be used to refer to that requirement in other parts of the RD on in other system documentation;
2. It must define the *traceability policies*. Traceability information allows dependencies between requirements, and between the requirements and the system design, components and documentation to be found;
3. *Use a Database* to manage requirements: rather than maintaining requirements in text documents, establish a requirements database and store the individual requirements as entries in this database.
4. Define *change management policies*: each partner can modify a requirement written previously.
5. Define a *common style* of requirements writing in order to have a well-structured document.

One important issue in collaborative requirements management is *semantic consistency* among requirements coming from different sources. This issue has been for example addressed in [7]. In the context of the MODCONTROL project semantic consistency has been addressed by separation of concerns, giving responsibility of each different function or subsystems to one partner.

3. MODCONTROL TCMS case study

The MODCONTROL project [13] 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 does 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 project is the standardization of TCMS functional modules and their interfaces with other subsystems on-board and external to the train. This objective will be achieved by the production of specifications for standardized interfaces and functionality first and then, later, by the development of a "Virtual Train Control System" and its demonstrators.

The MODCONTROL approach is to elaborate a generic Functional Requirements Specification (FRS) and a System Requirements Specification (SRS) for a 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. The intended result of this requirements engineering activity is a set of harmonized specifications for the train's functionality and the interoperability of subsystems and devices. Also precision, consistency and feasibility of requirements from already existing specifications are checked in this phase of the project. The requirements are collected on a common central server, with a central tool installation [11] and in a common project-wide structure. This common server is an important means for the collection of requirements and the project wide communication of requirements.

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

Project partners enter their requirements on functions or devices into the appropriate positions of these structures.

The result so far is a document composed of 5777 elements subdivided in requirements for a TMCS function (**FREQ**), requirements for devices carrying some functions or sub-functions (**SREQ**), all glossary items within the project (**TERM**) and description of use cases (**UC**)

4 Quality Evaluation

Several studies dealing with the evaluation and the achievement of quality in NL RD can be found in the lit-

erature. In order to automatize the evaluation process some tools have been proposed [10, 9, 4]. We describe here the tool **QuARS** (Quality Analyzer for Requirements Specifications). QuARS has been developed by "ISTI - CNR" [14, 5, 4] and it has been recently used for the evaluation of NL RD in real world projects [2]. It 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 lingustic analysis of a RD in plain text format and points out the sentences that are defective according to the expressiveness quality model described in [5, 4]. 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.
- 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 over the whole document. Among the metrics calculated by QuARS, we cite the readability index and the defect rate.

Readability: This metric is given by the Coleman-Liau Formula readability index: *5.89*(Nl/Nw)-0.3*(Ns/(Nw/100))-15.8* [3] where *Nl* is the number of letters in the RD, *Nw* is the number of words in the RD and *Ns* is the number of requirements sentences. The reference value of this formula for an easy-to-read technical document is 10, if it is greater than 15 the document is difficult to be read.

Defect Rate: It is equal to *Nds/Ns* where *Nds* is the number of requirements that after an evaluation with the QuARS tool presents defects in the RD and *Ns* is the total number of requirements.

The availability of tools such as QuARS allow a Quality Analysis process to be defined, which is here conjugated in the case of several partners providing the requirements to be merged (see Figure 1).

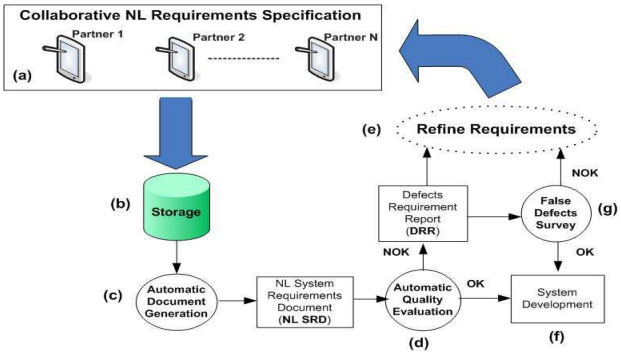

Figure 1. Quality Analysis Process

(a) The partners of the project create a new file project in RequisitePro [11] and insert the requirements with common attributes (name, text, author, Package, etc.). ***(b)*** The different requirements are stored in a unique Requirements File (***NL SRD***). ***(c)*** At this point, in automatic way, the tool SoDA [12] generates a text document and saves it with **txt** format (alternative formats are *doc*, *html* and *xml*). ***(d)*** The txt file is input to QuARS that analyzes the sentences (requirements) and gives in output the Defects Requirement Reports (DRR), for both FREQ and SREQ documents, together with the calculation of relevant metrics. ***(e)*** In the case QuARS points to some defects lowering the quality of the RD, a refinement activity is needed, followed by another quality analysis step. ***(f)*** Otherwise, the development process can start on the basis of the approved requirements document. Actually, it may well be the case that the detected defects are *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. Hence, the DRR should be filtered by experts, in a "*false defect survey*" ***(g)***, in order to establish whether a refinement is really necessary or not. The key aspect of this process is that the structure of a DRR document is very clear and easy-to-use for the partners. After the quality analysis each partner is able to refine its requirements on the basis of DRR documents, to be resubmitted to the central storage.

The DRR is a collection of HTML pages organized in a main directory and four subdirectories (shown in Fig. 2).

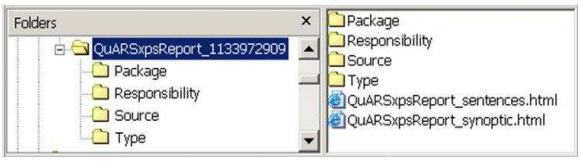

Figure 2. The DRR structure

The name of the main directory follows the form: "*QuARSxpsReport_ReportID*" where "*QuARSxpsReport*" stands for "*QuARS Express Edition Report*" and the "*ReportID*" is a unique progressive number assuring that a new file will never overwrite an old one. In the main directory there are two files and four sub-directories. The two files are the *"synoptic"* report page and a report page listing only *"sentences"*. The synoptic page is a summarized report of the analyzed document, showing all the defective Requirements, some statistics and a synoptic view of the performed analysis by means of the metadata attributes (e.g. *Package, Responsibility, Source* and *Type*). The *sentences* page focuses on the set of requirements regardless their attributes: it is a summary of all the defective requirements. The synoptic page can be really huge and a browser can take several seconds to load the whole page. The four sub-directories "*Package*", "*Responsibility*", "*Source*" and "*Type*" can contain several HTML files. These files are the reports about the same set of Requirements analysis cataloged by the metadata point of view allowing an aided traceability in respect of authors, source document, requirement type or belonging package.

As mentioned, the DRR is composed by HTML pages organized in four folders. These pages share some common items:

- the "*header*" of the file that specifies the path of the analyzed requirements file, the reference metadata item, the session pointed out by the unique 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*" listed by their IDs, 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 defective wording and the kind of analysis performed, where every *Sentence ID* is a link pointing to the complete description (full view) of the requirement analysis;
- any defective sentence analyzed one by one more in detail with all the metadata shown and the separated lexycal and syntactic analysis results. Any *Sentence ID* in the page points to this section of the page, actually the *"full view"* analysis section;
- some statistics: the "*global synoptic version*" is related to the whole document, to all of the sentences found. This version is used in the *synoptic* page and in the *sentences* page.

QuARS points out the "*Defect Rate*", the "*Analysis Defect Rate*" and the "*Error Rate*". In the following there is an explanation of the meaning of these statistics:

- **Defect Rate** : the number of requirements found in the document with at least an error (defective requirements) divided by the number of the analyzed requirements (e.g. all the requirements found in the document).
- **Analysis Defect Rate** : the number of requirements with at least an error of the kind of the related analysis item (e.g. "*Optionality, Subjectivity, Vagueness, Weakness, Implicity, Multiplicity, Underspecification*") divided by the number of defective requirements found in the document.
- **Error Rate** : the percentage of errors of the Optionality, Subjectivity, Vagueness, Weakness, Implicity, Multiplicity, and Underspecification w.r.t. the total number of errors found in the document.

Moreover, in the synoptic page there are synthetic views of the distribution of the defective sentences according to the chosen metadata with summaries by Responsibility, Package, Source and Type in both a simple view and a more detailed view containing links to specific requirements.

In this case the "Defect Rate" is computed among the sentences belonging to the metadata item value (in this case the number of sentences without a specified Responsibility).

5 Results

Here we present the outcomes of the MODCONTROL experience, performed using the following tools: RequisitePro [11] for Requirements Management and for some metrics, SoDA [12] for the automatic generation of a collaborative requirements document with the use of some templates and the current version of QuARS [14] for the NL quality evaluation. We have analyzed the System Requirements (SREQ) and the Functional Requirements (FREQ) documents. These documents have different size in terms of number of requirements: 2568 and 3209 respectively. The results of these experiments have shown that the underlying methodology not only can be able to point out linguistic defects, but can provide also some indications on the writing style of different NL requirements editors (from different partners) giving them the opportunity to become aware of defects in their requirements and potential improvements. In particular, it has been noted that a requirement editor is inclined to repeat the same type of mistakes, unless becoming aware of them. In figure 3 we can see the number of requirements (SREQ or FREQ) assigned to 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 the project partner **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

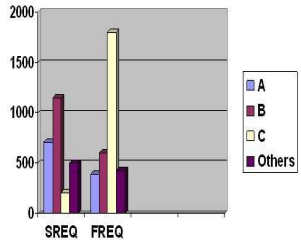

Figure 3. Requirements for each partner

We have the following results for the various metrics presented in section 4. In figure 4 we can see the results of the QuARS application on SREQ and FREQ documents; giving their **"Readability Index"**. We can see that the values of this metric is *12.178* for the FREQ document and *12.7724* for the SREQ document, for this reason the first document is more readable and understandable of the second one, but anyway both are in the range of readable documents.

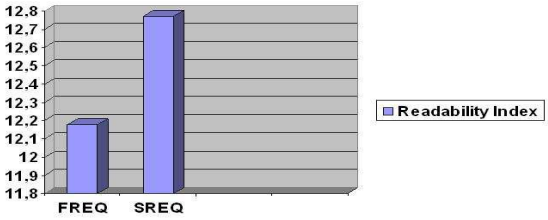

Figure 4. Readability Index

In figure 5 and 6 we can see the number of defective requirements, the errors present inside the RDs (FREQ and SREQ) and the *"Defect Rate"* associated to each partner of the project. These numbers, once false defect have been filtered out, can give an indication on which partner can be considered less accurate in the process of writing requirements. Another important result is what kind of defects is mainly revealed from the analysis. In Figure 7 and 8 we can notice that multiplicity and vagueness are more frequent. Figure 9, in the end, gives an idea about the execution time needed to perform the described analysis over such large documents. The difference between the execution speed depends on the text length of each requirement. SREQ requirements tend to be more concise than FREQ ones: apparently, describing functions require more verbosity.

Partners	Analyzed Requirements	Defective Requirements	Errors	Defetc 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%**

Figure 5. FREQ: Defect Rate and Errors

Partners	Analyzed Requirements	Defective Requirements	Errors	Defetc Rate
A	710	353	900	50%
B	1153	524	998	45%
C	208	46	88	22%
others	497	356	836	72%
Total	**2568**	**1282**	**2822**	**50%**

Figure 6. SREQ: Defect Rate and Errors

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

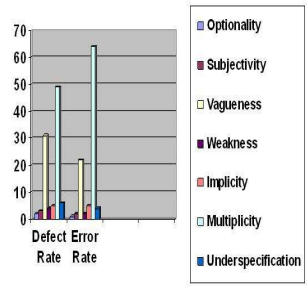

Figure 7. SREQ: Defects for Type

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

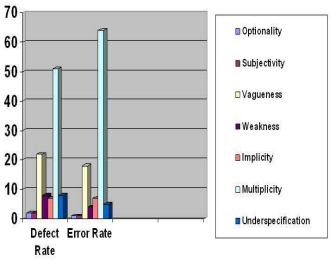

Figure 8. FREQ: Defects for Type

6 Conclusions

In the MODCONTROL project, we are still at the first refinement cycle: we have produced the DRR document for the whole set of requirements established so far. For the moment,we have not received the required feedback from all partners. Partners have been invited not only to correct

Documents	Number of Requirements	Execution Time (minutes)	Execution Speed (Req/minutes)
FREQ	3209	210	15.28
SREQ	2568	52.8	48.63

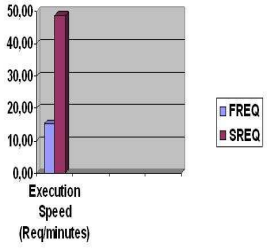

Figure 9. Execution Time of Analysis

defects, but also to give us knowledge about false defects. We have already identified some false defects, such as:

- Words usually indicating vagueness (like the word "deep") can be part of a well defined term in the problem domain (like "battery deep discharge" in electrical engineering).
- Sometimes the use of passive voice can be deliberately choosen by authors not to address a specific subject for the requirement. But in such cases, a discussion of the requirement among experts, initiated by reporting the "false defect", is useful to clarify and intended meaning of the requirement.

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. The results of this experience are promising. It has been very well appreciated inside the project, that the authors have been given the possibility to receive a feedback on the quality of their work, thus enabling them to improve style and precision in the next step of refinement of the collection of requirements. The availability of analysis tools based on natural language processing techniques has been crucial for the achievement of this goal.

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

[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] 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.

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

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

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

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

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

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

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