QuARS
A NLP Tool for Requirements Analysis

Stefania Gnesi and Gianluca Trentanni

CNR-ISTI, Pisa, Italy
stefania.gnesi@isti.cnr.it, gianluca.trentanni@isti.cnr.it

Abstract

QuARS (Quality Analyzer for Requirements Specifications) is a tool able to perform an analysis of Natural Language (NL) requirements in a systematic and an automatic way by means of natural language processing techniques with a focus on ambiguity detection. QuARS allows the requirements engineers to perform an early analysis of the requirements for automatically detecting potential linguistic defects.

1 Quality Analysis of NL Requirements: QuARS

NL requirements are widely used in software industry, at least as the first level of description of a system. Unfortunately they are often prone to errors and this is partially caused by interpretation problems due to the use of NL itself. An evaluation of NL requirements to address part of the interpretation problems due to linguistic problems was considered an interesting research problem. However, as any other evaluation process, the quality evaluation of NL software requirements needs the definition of a quality model. We defined a quality model composed of high level quality properties for NL requirements to be evaluated by means of indicators directly detectable and measurable on NL requirement documents distinguishing four quality types, namely *syntactic, structural, semantic,* and *pragmatic* [2, 3, 4, 6]. The quality model was the basis for implementing a tool, called QuARS [1] – Quality Analyzer for Requirement Specifications– for analyzing NL requirements in a systematic and automatic way [5].

The approach provided by QuARS is mainly focused on lexical and syntactic quality aspects, while the pragmatic aspect, which depends on the reader of the requirements, is not taken into account. In particular QuARS performs expressiveness analysis by means of a lexical and syntactic analysis of the input file in order to identify those sentences containing defects according to the quality model looking at:

1. Unambiguity: the capability of each Requirement to have a unique interpretation.
2. Clarity: the capability of each Requirement to uniquely identify its object or subject.
3. Understandability: the capability of each Requirement to be fully understood when used for developing software and the capability of the Requirement Specification Document to be fully understood when read by the user.

Indicators, in this case, are syntactic or structural aspects of the requirements specification documents that provide information on the defects related to a particular property of the requirements themselves.

[1] `http://quars.isti.cnr.it/`

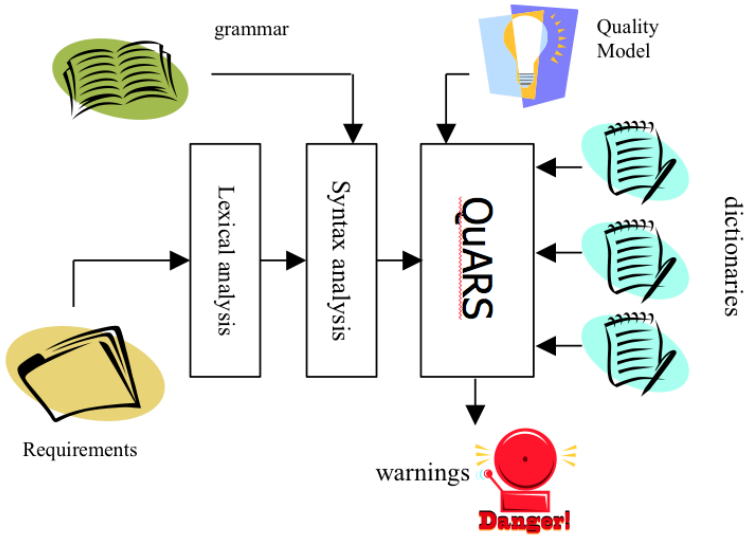

Figure 1: QuARS Process

1.1 Using QuARS

QuARS performs a linguistic analysis of a requirement document in plain text format and points out the sentences that are defective according to the expressiveness quality model according to the process depicted in Figure 1.

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 *implicity, multiplicity and under-specification* defects.

- Optionality means that the requirement contains an optional part (i.e. a part that can or cannot be considered) and example of Optionality-revealing words are: possibly, eventually, in case, if possible, if appropriate, if needed, ...
- Subjectivity means that the requirement expresses personal opinions or feelings, i.e. similar, similarly, having in mind, take into account, as [adjective] as possible,....
- Vagueness means that the requirement contains words having a no uniquely quantifiable meaning and example of Vagueness-revealing words are: adequate, bad, clear, close, easy, far, fast, good, in front, near, recent, significant, slow, strong, suitable, useful, ...
- Weakness means that the sentence contains a "weak" verb. A verb that makes the sentence not imperative is considered weak (i.e. can, could, may, ...).
- Implicitly means that the requirement does not specify the subject or object by means of its specific name but uses a pronoun or other indirect reference. Demonstrative adjectives (this, these, that, those) or Pronouns (it, they...) or terms having the determiner expressed by a demonstrative adjective (this, these, that, those) or implicit adjective (i.e. previous, next, following, last...) or preposition (i.e. above, below...) are considered implicity indicators.
- Multiplicity: the occurrence of multiplicity-revealing terms: and/or, or, ... is considered a multiplicity indicator, as well as the presence of itemized lists.
- Under-specification means that the requirement contains a word identifying a class of objects without a modifier specifying an instance of this class. The occurrence of wordings needing to be instantiated (i.e. information, interface, that must be better defined, flow instead of data flow, control flow, access instead of write access, remote access, authorized access, testing instead of functional testing, structural testing, unit testing, etc.) is considered an under-specification indicator.

Table 1: Example of Requirements sentences containing defects

Indicators	**Negative Examples**
Optionality	the system shall be.., ***possibly*** without..
Subjectivity	***..in the largest extent as possible***..
Vagueness	the C code shall be ***clearly*** commented..
Weakness	the initialization checks ***may be*** reported..
Implicity	the ***above*** requirements shall be verified..
Multiplicity	the mean time..and ***restore service***..
Under-specification	..be able to run also in case of ***attack.***

In Table 1 we can see some examples of requirements that contain linguistic defects.

When the analysis is performed, the list of defective sentences is displayed by QuARS and a log file is created. The defective sentences can be tracked in the input requirements document and corrected, if necessary. Metrics measuring the defect rate and the readability of the requirements document under analysis are calculated and stored. The available metrics are the Coleman-Liau Formula readability metrics [1] an the defect rate (i.e. the number of defective sentences / the total number of sentences).

1.2 Ambiguity versus Variability

Ambiguity defects that are found in a requirements document may be due to, intentional or unintentional, references made in the requirements to issues that may be solved in different ways, possibly envisioning a set of different products rather than a slngle product. We therefore can use the analysis ability of QuARS to elicit the potential variability hidden in a requirement document. Variability may be due to vagueness and vagueness occurs whenever a requirement admits borderline cases, e.g., cases in which the truth value of the sentence cannot be decided since vague terms are used in it. QuARS allows the creation of new dictionaries useful for defining new indicators characterising potential variability in requirements. Variability may be revealed by the occurrence of variability-revealing terms such as: if, where, whether, when, choose, choice, implemented, implement, implements, provided, provide, provides, available, feature, range, select, selected, selects, configurable, configurate, ... [11].

1.3 QuARS User Interface

The QuARS GUI is composed of three main frames. The Input Frame that allows to load, display and edit input file containing the requirements to be analyzed (the supported file format is plain text). The Dictionary Frame that allows the user to select, display and edit the dictionary corresponding to the type of analysis of interest. The Output Frame where the results of the analysis are displayed. Figure 2 shows the QuARS GUI. Figure 3 reports the output of an analysis performed according to the vagueness criterion.

2 QuARS: Application Experiences

QuARS has evolved from an initial prototype to the current reliable and user-friendly version after subsequent experiments over several case studies aimed at evaluating the effectiveness of the methodology and identifying improvement opportunities in terms of both usability and provided functionalities. Some of the case studies [4] came from industrial projects and these belonged to several application domains. More recently, two different experiences have been reported to automatically identify quality defects in natural language requirements in the Railway Domain by using QuARS and the SREE tool, that is an extension of QuARS defined by means of the GATE tool in [8] and [12].

In [**?**, 10] QuARS has been instead used to study a classification of the forms of ambiguity that indicate variation points starting from the analysis of documents describing real systems, since ambiguity or underspecification at requirements level can in some cases give an indication of possible variability, either in design choices, in implementation choices or configurability.

All these experiments provided us with feedbacks to evolve the tool itself, and allowed us to gather a record of information on the:

- Effectiveness of the tool in finding defects: the number of defects found in the NL requirement document depends more on the experience and skill of the requirements engineer than on the company maturity.

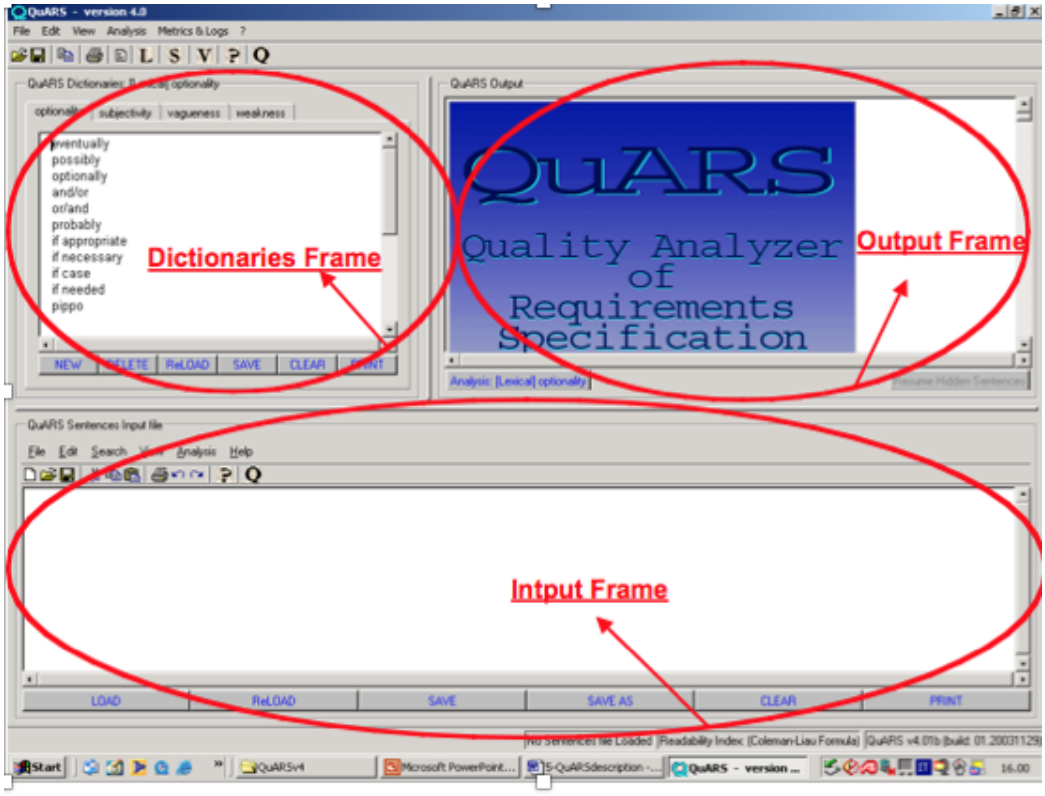

Figure 2: QuARS User Interface

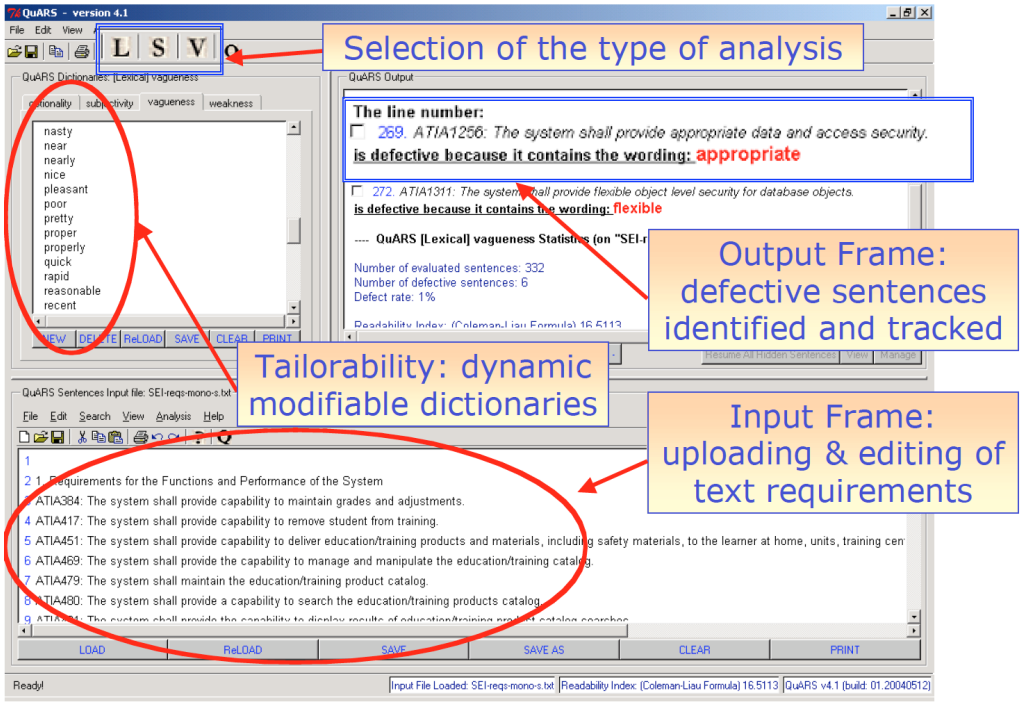

Figure 3: QuARS analysis

- Frequency and typology of false positives: the presence of false positives has been observed in every case study. It can be considered as a physiological side effect of the application of the tool. The rate of false positives respect to actual defects has been rarely over 10%.
- Effort required to apply the tool and to tailor the dictionaries for specific application: the effort required to perform the analysis of a requirements document is relatively low. The main effort is due to the preparation of the input document since this has to be in plain text format.

Acknowledgements

This work was partially supported by the H2020 Shift2Rail project AstRAIL.

References

[1] Meri Coleman, T. L. Liau, A computer readability formula designed for machine?scoring. Journal of Applied Psychology, 60, 283-284. 1975.

[2] Fabrizio Fabbrini, Mario Fusani, Vincenzo Gervasi, Stefania Gnesi, Salvatore Ruggieri: On Linguistic Quality of Natural Language Requirements. 4th REFSQ,57-62, Presses Universitaires de Namur, 1998.

[3] Fabrizio Fabbrini, Mario Fusani, Stefania Gnesi, Giuseppe Lami: The linguistic approach to the natural language requirements quality: benefit of the use of an automatic tool, 26th Annual NASA Software Engineering Workshop, 97-105, IEEE, 2001.

[4] Fabrizio Fabbrini, Mario Fusani, Stefania Gnesi, Giuseppe Lami, An automatic quality evaluation for natural language requirements, 7th REFSQ, 2001.

[5] Stefania Gnesi, Giuseppe Lami, Gianluca Trentanni: An automatic tool for the analysis of natural language requirements. Computer. Systems: Science & Engineering. 20(1), 2005.

[6] Daniel M. Berry, Erik Kamsties, Michael M. Krieger: From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity. University of Waterloo, 2017. `https://cs.uwaterloo.ca/~dberry/handbook/ambiguityHandbook.pdf`

[7] Daniel M Berry, Antonio Bucchiarone, Stefania Gnesi, Giuseppe Lami, Gianluca Trentanni, A new quality model for natural language requirements specifications, Proceedings of the international workshop on requirements engineering: foundation of software quality, 12th REFSQ, 2006.

[8] Antonio Bucchiarone, Stefania Gnesi, Gianluca Trentanni, Alessandro Fantechi: Evaluation of Natural Language Requirements in the MODCONTROL Project, ERCIM News 2008(75), 2008.

[9] Alessandro Fantechi, Stefania Gnesi, Laura Semini: Ambiguity defects as variation points in requirements. 11th VaMoS: 13-19, ACM, 2017.

[10] Alessandro Fantechi, Alessio Ferrari, Stefania Gnesi, Laura Semini: Hacking an Ambiguity Detection Tool to Extract Variation Points: an Experience Report, 12th VaMoS: 43-50, ACM, 2018.

[11] Alessandro Fantechi, Alessio Ferrari, Stefania Gnesi, Laura Semini: Requirement Engineering of Software Product Lines: Extracting Variability using NLP, 26th RE 2018: 418-423, IEEE, 2018.

[12] Benedetta Rosadini, Alessio Ferrari, Gloria Gori, Alessandro Fantechi, Stefania Gnesi, Iacopo Trotta, Stefano Bacherini: Using NLP to Detect Requirements Defects: An Industrial Experience in the Railway Domain. 23rd REFSQ, LNCS 10153, 344-360, Springer 2017.