A semi-automatic verification tool for software requirements specification documents

Fabio Konig, Luciana Ballejos, and Mariel Ale

***Abstract*—Most software problems arise from deficiencies in the manner in which software requirements are elicited and expressed. Ensuring that the Software Requirements Specification document (SRS) has the necessary quality is crucial to the success of any software development project, since its information is used across all project stages. In this paper, we present a semi-automatic verification tool for SRS documents based on a comprehensive quality model.**

***Index Terms*— Semi-automatic Verification, Software Requirements Specification, Software Quality Models**

I. Introduction

THE primary measure for a software-intensive information system to be successful is the degree in which it meets the intended purpose. Requirements Engineering (RE) is a subtask of Software Engineering, which deals with the discovering of that purpose, by identifying stakeholders and their needs, and documenting them for their future analysis, communication, and subsequent implementation [1].

In RE processes there is a continual need for efficiently managing a great volume of information and knowledge generated and used during all activities involved in software development process. Thus, diverse are the challenges that must be considered when managing requirements-related information in software development projects. In this sense, ambiguous requirements must be minimized, since they produce waste of time and repeated work. The same occurs with software requirements volatility, where unstable requirements have significant impact on project performance, regarding time and effort [2].

Related to this, there exist in the literature diverse proposals in order to give guidance in the assessment of different attributes or properties for requirements, which helps in controlling if their specification is made in a correct way. Most of these proposals lack of a concrete implementation, or they are proprietary products of high cost and difficult customization and configuration.

In this paper, an open-source semi-automatic verification tool for SRS documents based on a comprehensive quality model is presented.

The paper is organized as follows: Section 2 describes the most representative software requirements quality models in the literature. Section 3 describes the tools for evaluating the quality of the requirements that can be found in the market. Section 4 presents RQV Tool together with an application case and finally, Section 5, is devoted to discuss conclusions and future trends in this area.

II. Software Quality Models

The quality evaluation of software requirements, like any evaluation process, has to be carried out following a model that provides a reference frame, in order to formalize the definition of quality to be associated to a type of software product or artifact. This allows objectivity in the evaluation of the product under verification [3][4].

A quality model, in general, is composed by quality properties to be evaluated through quality indicators [3]. Then, a requirements quality model is defined as the set of rules against which a requirements document (syntactic and semantic rules, structural characteristics for the document and its sentences) should be evaluated [4].

There are several proposals for requirements quality models. Some of them, propose a list of desirable quality properties of requirements [3][5][6][7][8][9][10][11][12] while others, provide a defect taxonomy, where a defect in requirements reflects the absence of any of the quality characteristics [13][14].

It should be considered that, although the quality of the SRS is attainable, perfection is not. Any of the quality properties can be achieved, but often at the expense of other properties. Requirements analysts for each project must agree on which quality properties are priorities [5].

Table 1 summarizes the proposals that describe quality models and the specific desirable properties that make them the most referenced in the literature in the area.

As it can be seen, several quality properties are repeated in many of these models. Many of them coincide in their names and descriptions, but others do not. On the other hand, the analyzed quality models differ in their scope, since some authors propose quality properties for SRS, while others

This work was supported by Universidad Tecnológica Nacional – Facultad Regional Santa Fe

• F. Konig is with the Universidad Tecnológica Nacional – Facultad Regional Santa Fe - Argentina. E-mail: konigfabio@gmail.com.

• L. Ballejos is with the Centro de Investigación y Desarrollo de Ing. en Sistemas de Información – UTN-FRSF - Argentina. E-mail: lucianaballejos@gmail.com.

• A. Mariel is with the is with the Centro de Investigación y Desarrollo de Ing. en Sistemas de Información – UTN-FRSF - Argentina E-mail: male@frsf.utn.edu.ar.

contemplate quality properties for individual requirements in the SRS.

It can also be observed that some quality models are more likely to be automatically verified through specific indicators, since they propose more specific quality properties and measurement methods, for example Fabbrini et al. quality model [3]. On the other hand, other models presented in Table 1 are more general models [5], while others are more limited, suggesting only a few desirable properties [3][8].

TABLE 1
REQUIREMENTS QUALITY MODELS PROPOSED BY DIFFERENT AUTHORS

Quality Models Quality Properties	Davis y otros (1993)	Fabbrini y otros (2001)	Wiegers y Beatty (2013)	IEEE 830:1998 (R2009) (2009)	Loucopoulos y Karakostas (1995)	Wilson (1997)	Pohl (2010)	Swathi y otros (2011)	Gènova y otros (2011)
Reachable	X		X						
Annotated by	X		X	X		X	X	X	
Annotated by	X			X		X	X	X	
Annotated by	X								
Correct Level of	X					X			
Atomic							X		X
Complete	X	X	X	X	X	X	X	X	X
Concise	X					X			
Correct	X		X	X		X	X	X	X
Cross-references	X								
Design	X				X				X
Electronically	X								
Consistently	X	X	X		X			X	
Internally	X	X	X	X	X	X	X	X	X
Modifiable	X		X	X		X	X	X	X
No Redundant	X				X				
Organized	X					X			
Precise	X								X
Interpretable	X								
Reusable	X								
Traceable	X		X	X		X	X	X	X
Tracing	X		X	X			X	X	
Unambiguous	X		X	X	X	X	X	X	X
Understandable	X	X				X	X		X
Updated							X		
Verifiable	X	X	X	X		X	X	X	X

Another particularity detected was that some authors detail each quality property separately, giving a deeper explanation, while others group several characteristics into a single property. For example, Davis et al. [5] distinguish traceable and traced properties, whereas the IEEE 830: 1998 (R2009) [7] group these two properties into traceable property.

In addition, not all quality models refer in the same way to the same property. For example, Genova et al. [6] define the abstraction property to refer to the design independent property.

It can be concluded then, that it is necessary to select a subset of quality properties, that is, desirable properties applicable to the complete requirements document and to individual requirements contained therein, in order to obtain a unifying and integral proposal that serves as a quality model for SRS evaluation. This will allow not only the coherent and concrete definition of each proper-ty, but also the definition of quantifiable indicators

A. Comprehensive Quality Model

The evaluation of the software requirements quality has to be carried out following a model that provides a guidance, in order to formalize the definition of quality to be associated with a work artifact, and that provides objectivity in the evaluation of quality [3][4]. Saavedra [15] proposes a requirements quality model, selecting a sub-set of desirable quality properties to be achieved in a SRS. The selection criterion for quality properties considers, on the one hand, the need for the presence of certain properties to guarantee the quality of the SRS and its requirements and, on the other hand, the feasibility of verification or compliance of such properties.

The properties of the quality requirements model are:

- Unambiguous: A SRS is unambiguous if each requirement stated in it has only one possible interpretation, that is, if all stakeholders with approximately the same knowledge about the sys-tem and its context, interpret each requirement in the same way.
- Understandable: A SRS is understandable if its readers can easily understand the meaning of all requirements with minimal explanation.
- Complete: A SRS is complete if it contains the following elements: 1) All significant requirements; 2) Definition of software responses to all feasible classes of input data, in all kinds of realizable situations; 3) Complete labels and references to all figures, tables and diagrams in the SRS and definition of all terms and units of measure.
- Correct: An SRS is considered correct if each requirement contributes to the satisfaction of some need.
- Internally Consistent: An SRS is considered internally consistent if no subset of requirements de-fined in it has conflicts.
- Accurate: A SRS is accurate when all the terms used in it are concrete and well defined.
- Atomic: A requirement is atomic if it describes a single, coherent event.
- Modifiable: A SRS is considered modifiable if its structure and style are such that any change can be introduced in an easy, complete, and consistent manner, without affecting those characteristics.
- Organized: A SRS is considered organized if its content allows readers to easily locate information and logical relationships between adjacent sections are evident. To achieve a useful organization, a standard should be followed.
- Annotated by Relative Importance: A SRS is considered annotated/classified by importance if each requirement in it has an identifier to indicate its importance.
- Annotated by Relative Stability: A SRS is considered

annotated/classified by stability if each requirement in it has an identifier to indicate the stability of that particular requirement.

- *Traceability*: A SRS is considered traceable (for-ward traceability) if each of its requirements is easily referenced in the later development phase or improvement documentation, i.e.: in all documents generated from the SRS.
- *Traceability*: A SRS is considered traced (back-wards traceability) if each one of its requirements has a clear origin, that is, it is linked to the previous stages of development.
- *No Redundant*: A SRS is redundant if the same requirement was specified more than once. Un-like other quality properties, redundancy is not necessarily bad. It is often used to increase the readability of the document. However, it causes problems when the SRS is reviewed. If all occurrences of a redundant requirement are not modified, then the SRS becomes inconsistent.
- *Concise*: A SRS is considered concise if it is as short as possible, without adversely affecting any other quality property of the document.
- *Correct Detail/Abstraction Level*: A SRS can provide different levels of detail in its content. It is considered good practice to write requirements in a consistent level of detail.
- *Design Independent*: A SRS is design independent if there is more than one system design that implements all the requirements defined in it. The SRS requirements should tell what the system should do without saying how it should do it.
- *Electronically Stored*: A SRS is considered electronically stored if the entire document was made with a word processor, was generated from a requirements database, or was synthesized in some other way.
- *Verifiable*: A SRS is considered verifiable if every requirement stated on it can be verified. A requirement is verifiable if there is a finite and cost effective process with which a person or machine can verify that the software product meets the requirement.

The strategy proposed by Saavedra [15] consists of a plan to implement a set of quality indicators and quality indexes, at requirements and SRS levels, that allow to measure and evaluate the quality properties of the requirements model.

The indicators are instruments that allow to quantitatively express the qualitative SRS and its requirements quality properties, serving as a guide to evaluate the quality of these artifacts. The indicators give an indication of how to interpret the measurement performed.

The existence of a set of indicators allows to objectively know the quality of requirements and the SRS, also facilitating the comparison of results. This information, which will be processed automatically by the tool presented in this paper, is the basis for detecting the aspects to be improved in the SRS and its requirements.

On the other hand, quality indices are presented as a complex grouping of different indicators. Quality level requirement indexes that allow to measure quantitatively the quality properties of each requirement (by means of an aggregation of the results of each indicator that affects the property concerned) at the level requirement were established in Saavedra [15] model. These indexes allow the measure of each requirement quality and determines which of them should be improved and gives priority for improvement.

Finally, the model also define quality level SRS indexes that allow to measure quantitatively the quality proper-ties of the SRS, by means of an aggregation of the results of each indicator level SRS affecting tested quality property. These indexes allow the measure of the quality of the SRS and determine improvements required to achieve a good quality SRS.

The quality indicators proposed in the model are classified according to the assessment approach that follows its procedure of measurement.

The evaluation approaches are listed below and the quality indicators defined for each of them are presented in Table 2:

- Use of natural language patterns: is based on the detection of keywords, key phrases or symbols, defined in corpus, as evidence of the failure of certain quality properties. The indicators that follow this approach son I-1, I-2, I-3, I-4, I-5, I-6, I-7, I-8, I-9, I-10, I-11, I-12, I-13, I-14, I-15 and I-16.
- Use of domain vocabulary: related to the use of user vocabulary (glossary) in requirements descriptions. Lexicon Extended Language (LEL) is used as a glossary, one of the RI information sources that was selected because of its expressive power. The indicators that follow this approach are I-29 and I-30.
- Use of domain knowledge: it considers domain interpretation and semantic knowledge. In this model, LEL is also used as a knowledge re-source. The indicators that follow this approach are I-21 and I-32.
- Use of grammatical rules: is based on the identification of grammatical rules. The indicator that follows this approach is I-17.
- Use of specific characteristics of a requirement: it considers the detection of specific characteristics of a requirement, such as: readability, unique identifier, traceability to the origin, etc. The indicators that follow this approach are I-18, I-22, I-25 and I-26.
- Use of the SRS document-specific features: includes the analysis of specific SRS characteristics, such as: presence of sections, readability, etc. The indicators that follow this approach are I-19, I-20, I-23, I-27, I-28, I-31, I-33 and I-34.
- Use of overlap between requirements: considers requirements referencing the same subject, where can be distinguished: contradictions between requirements, redundancy when there is an un-necessary repetition and simple coupling when there is none of the above, but implies some kind of dependency relation. The indicator that follows this approach is I-24.

The indicators proposed have a measurement scope, that is, some indicators are applied at the requirement level, where can be found those that apply only to functional requirements (FR) and those that apply to functional and/or non-functional requirements (FR/NFR).

On the other hand, indicators that applied at the SRS level - the complete requirements document-, were also defined. It is important to mention that the quality indicators defined in this paper, when they do not meet the established goal for each of them, negatively affect, directly or indirectly, certain quality properties of the requirements model, either as a potential problem or suggestion/warning to improve.

The tool to support this model contemplates the possibility of enabling / disabling quality indicators according to the importance of this indicator for the organization or project.

In table 2 are shown, highlighted with an "X" in the corresponding row and column, the quality properties that are adversely affected when the evaluation of the quality indicators it is not satisfactory. In addition, it is highlighted with an "I" in the corresponding row and column, the quality properties that are negatively affected indirectly.

III. TOOLS FOR QUALITY ASSESSMENT OF SOFTWARE REQUIREMENTS

Several tools deal with some form of evaluation of the SRS requirements quality. Following are the most prominent ones found in the literature and studied by the RAMP (Requirements Analysis and Modeling Process) project, which is described below.

IBM Rational DOORS[1] is a commercial-grade, industrial-use requirements management application developed by IBM to optimize communication, collaboration, and requirements verification across an organization. This solution targets business objectives by managing project scope and cost. Rational DOORS lets capture, plot, analyze, and manage changes to information while maintaining compliance with regulations and standards.

RQA[2] is a commercial tool for industrial use, developed by The Reuse Company in close collaboration with the Knowledge Reuse Group of the University Carlos III of Madrid that allows defining, measure, improving and managing the quality of the specifications of requirements. It was not designed to be a general requirements management tool, but to operate in collaboration with other tools. Therefore, it receives the requirements as input data and calculates quality metrics and recommendations as output.

LEXIOR[3] was developed in France by the company Cortim. It is of commercial type. It is for industrial use and offers requirements documents revision services. Documents can be reviewed during their initial drafting phase, allowing the identification of recurring document errors and can also be reviewed during the formal review phase, allowing detailed error reports to be generated. The LEXIOR document review service is a turnkey solution.

Requirements Assistant[4] is a commercial tool for industrial use, developed in the Netherlands by the company Sunny Hills, designed to meet only the criteria: complete, consistent, feasible and unambiguous, in the requirements phase of a project. It detects bad requirements, that is, those that contain words considered diffuse and detect the lack of some type of non-functional requirement such as reliability, security, and so on.

DESIRe[5] is a commercial tool for industrial use, developed in Germany by the company HOOD that gives support to requirements engineers to guarantee the quality of the requirements in natural language. DESIRe is a method for automatically identifying words in the requirement text and indicating predefined questions, observations and information according to those previously identified words. These questions, comments and information indicate possible weaknesses in the requirement. The engineer will then try to answer the questions raised and if necessary, he/she can rewrite the content of the requirement. This ensures that the rules of complete-ness, non-ambiguity and comprehensibility are respected.

TABLE 2
RELATED QUALITY INDICATORS AND PROPERTIES

Quality Properties Indicators	Unambiguous	Understandable	Complete	Correct	Internally Consistent	Accurate	Atomic	Modifiable	Organized	by Relative Importance	by Relative Stability	Traceability	Traceability	No Redundant	Concise	Correct Detail/Abstraction -	Design Independent	Electronically Stored	Verifiable
I-1 level of occurrence of copulative conjunctions terms	X					I	X					I	I				I		I
I-2 level of occurrence of connector terms	X					X	X					I	I				I		I
I-3 level of occurrence of ambiguous terms	X					X													I
I-4 level of occurrence of rational terms															X	X			
I-5 level of occurrence of speculative terms	X					X													I
I-6 level of occurrence of subjective terms						X													
I-7 level of occurrence of negative terms		X		I															I
I-8 occurrence level of parentheses																			
I-9 level of occurrence of imperatives and optional terms	X					X													I
I-10 level of occurrence of continuations terms															X	X			
I-11 level of occurrence of flow control terms						I	X	I				I	I				X		I
I-12 level of occurrence in of design terms								I									X		I
I-13 level of occurrence of incomplete terms			X																I
I-14 level of occurrence of conditional modes						X													
I-15 level of occurrence of annotation by relative importance terms										X									

[1] http://www-03.ibm.com/software/products/en/ratidoor

[2] http://www.reusecompany.com/requirements-quality-analyzer

[3] http://www.cortim.com/pageLibre0001010b.html

[4] http://www.requirementsassistant.nl/

[5] http://www.hood-group.com/en/requirements/beratung/vorgehensentwicklung/desirer/

I-16 level of occurrence of annotation for relative stability terms											X								
I-17 level of occurrence of passive voice						X													
I-18 index of readability at requirement level		X		I															I
I-19 index of readability at SRS level		X		I															I
I-20 level of occurrence of univocally identified requirements												X							
I-21 level of occurrence of wrong functional requirement				X															
I-22 level of occurrence of origin of identified requirements													X						
I-23 level of occurrence of identified origins of requirements													X						
I-24 Level of occurrence of redundant requirements								I						X					
I-25 level of occurrence of subject	X		X			X													I
I-26 level of occurrence of symbol subject				X															
I-27 level of organization of the SRS									X										
I-28 level modifiability of the SRS								X											
I-29 level of occurrence of domain terms	X	X		I		X													I
I-30 level of occurrence of acronyms and abbreviations	X	X		I															I
I-31 level of occurrence of incomplete sections of the SRS			X																I
I-32 low specification level	X					X										X			I
I-33 level of low reference to the domain vocabulary				I	X														I
I-34 level of electronically stored SRS																		X	

QuARS[6] is a commercial tool for academic use, developed in Italy at the University of Pisa, designed to perform a syntactic analysis of the SRS requirements sentences, specifically a document in text format written in English language, and indicate a series of potential error sources in a SRS.

TigerPro[7] is an open-source, academic-use tool, developed in Australia by the University of South Australia, to import and elicitate requirements that help write good requirements, allow quick review of documents, ensure the completeness of requirements contained in multiple paragraphs and conveys the lessons learned. In addition, this tool can help to correct some defects in requirements, clarify requirements from the testing perspective, and point out those requirements that may be difficult to verify, or written in a way that complicates the testing. It does not find all the defects in the requirements, but it goes a great way for the improvement of the written requirements.

From the tools analysis and the results obtained by the RAMP project, the following issues can be observed:

- Many of the evaluated tools are commercial, which implies their acquisition implies certain cost. Related to this, they have specific software requirements (using proprietary software) and hardware that limits their installation.
- In some tools, for example RQA which is the most promising according to RAMP, the input requirements are limited to specific formats without considering more usual formats such as txt, doc, pdf, xml.
- Some tools, such as LEXIOR, which are known to be semiautomatic, require too much human intervention by reviewers in the process of reviewing requirements documents, which have to be native English speakers and have industrial experience in complex systems.
- In general, there is a lack of consideration of the specific information sources of Requirements Engineering that optimize the metrics of requirements evaluation , since they provide specific information of the domain that should be included or considered in the SRS.

As a consequence of the aforementioned issues, arises the need of the construction of a tool that implements the defined metrics and indicators, in order to support an automatic evaluation, promoting the generation of a quality SRS.

IV. RQVTOOL

In order to give support to the Analyst or Requirements Engineer in the semiautomatic verification of SRS quality, a tool was developed that implements Saavedra [15] model presented in 2.1.

The objective of this tool is to automate, within the technical possibilities, the evaluation of the quality properties of the requirements quality model proposed by Saavedra [15], using as input the SRS document and the Extended Language Lexicon (LEL) as Domain knowledge resource (for cases where it is available) and incorporating the best practices of what has been developed so far.

Then, to perform its quality assessment, the tool uses some control mechanisms:

- Words corpus, phrases and/or key symbols used for the evaluation of some quality indicators that detect the presence of natural language patterns.
- ISO / IEC / IEEE 29148: 2011 (ISO / IEC / IEEE, 2011), which includes IEEE 830: 1998 (R2009) (IEEE, 2009) used for the evaluation of some quality indicators that are based on this standard for their calculation procedure.
- Grammar Rules, used in indicators where the presence of certain grammatical rules is verified.

As a result, RQV Tool calculates:

- Quality indicators (at the request level and at the SRS level). If these indicators do not meet their target, there is a potential problem or suggestion / warning to be improved, with a negative impact (directly or indirectly) on certain quality properties.
- Quality indices (at the request level and at the SRS level). They are based on the aforementioned quality indicators,

[6] http://quars.isti.cnr.it/

[7] www.therightrequirement.com/TigerPro/TigerPro.html

grouping those indicators that affect a certain quality property, at a certain level (requirement or SRS). They provide statistical information as an indication of the quality of each requirement or the overall quality of the SRS, for certain quality properties.

Figure 1 shows the above described in schematic form.

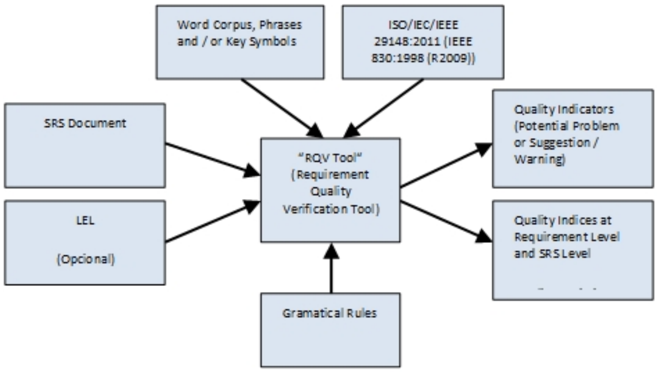

Word Corpus, Phrases and / or Key Symbols
ISO/IEC/IEEE 29148:2011 (IEEE 830:1998 (R2009))
SRS Document
LEL
(Opcional)
"RQV Tool" (Requirement Quality Verification Tool)
Quality Indicators (Potential Problem or Suggestion / Warning)
Quality Indices at Requirement Level and SRS Level
Gramatical Rules

Fig. 1. RQVTool Support Tool.

This tool was implemented following the premise of an OpenSource application. To do this, the Java Platform was used to create applications in Java programming language, which offers powerful user interfaces, performance, versatility, portability and application security.

Another of the premises of this tool is its adaptability, which is why a 3-layer architecture was used, which is a client-server architecture with the primary objective of separating business logic from design logic. The main advantage of this style is that the development can be carried out in several levels and, in case of any change, only the required level is modified without having to check between mixed code.

The layers involved in the RQV Tool architecture (Figure 2) are:

- *Presentation Layer:* corresponds to the graphic interface that the user sees, which communicates and captures user information with a minimum of process. In this tool the premise is that the interfaces are user-friendly, that is, understandable and easy to use. This layer only communicates with the business layer.

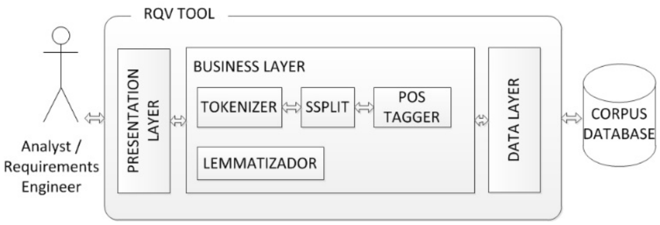


Fig. 2. RQVTool architecture.

- *Business Layer:* it corresponds to the logic of the business since it establishes all the rules that must be fulfilled. In this layer reside the programs that are executed, the user's requests are received and the answers are sent after the process. It communicates with the presentation layer, to receive the requests and provide the results and with the data layer, to request the database manager to store or retrieve data from it. The application programs are also considered here. Because processing of the SRS and LEL and evaluating the quality indicators proposed in RQV Tool requires natural language processing, this layer works with an integrated set of widely used natural language processing tools called the Stanford CoreNLP, whose software distributions are open source, licensed under the GNU (General Public License).
- *Data layer:* this is where the data resides and is responsible for accessing the data. It consists of a database manager that performs data storage, receives requests for storage or retrieval of information from the business layer. This tool used the object-relational database management system called PostgreSQL, distributed under BSD license and with its freely available source code, which, due to its technical characteristics, make it one of the most powerful and robust databases in the database market. RQV Tool uses this database manager for storing the corpus of words, phrases and key symbols.

The components / annotators included in Stanford CoreNLP that were used in RQV Tool are:

- *Tokenizer[8]:* This annotator allows you to divide the text into a sequence of tokens, which correspond approximately to words. It is used in RQV Tool to divide the SRS text into words.
- *Ssplit[9]:* this annotator takes as input the symbol sequence generated with "Tokenizer", and divides it into sentences. RQV Tool uses it to separate in sentences the text of the SRS, previously divided into words.
- *Part-Of-Speech Tagger (POS Tagger)[10]:* this tagger identifies within each sentence, expressed in a particular language, the parts of the sentence for each word, such as: noun, verb, adjective, etc. Inside RQV Tool is used to identify, for example, nouns and verbs in a requirement.

In addition, the Lemmatizator[11] component is used and corresponds to the use of a vocabulary and morphological analysis of words, with the objective of eliminating the inflectional terminations and returning the base form of a word, known as lemma. It provides, for each word, a tree-like structure that represents the family of that word. This family is composed of the lemma, which is the root of the tree and the different conjugations of such lemma or word. That is, words that are different but belong to the same family of words, are part of the same tree and share the same lemma.

In RQV Tool the Lemmatizator component is useful for cases where it is necessary to compare sentences that differ in the conjugations of some words. It is a way of bringing a word

[8] http://nlp.stanford.edu/software/tokenizer.shtml
[9] http://stanfordnlp.github.io/CoreNLP/ssplit.html
[10] http://nlp.stanford.edu/software/tagger.shtml
[11] http://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

or sentence into a canonical form, then comparing it and determining whether it is the same sentence or not.

Based on the architectural design presented, the main features of this tool are:

- OpenSource and, therefore, not restricted to a particular platform, since it was developed using free, non-proprietary technologies.
- SRS and LEL are accepted as a plain text entry (txt), in English.
- The SRS should follow the recommendations of ISO / IEC / IEEE 29148: 2011 (ISO / IEC / IEEE, 2011), which includes IEEE 830: 1998 (R2009) (IEEE, 2009), with the requirements univocally numbered.
- Quality indicators are available, at requirement and SRS level, which, in the event of failure to meet their target, have a direct or indirect negative effect on certain quality properties.
- Allows the configuration of Quality Indicators:
  - The corpus of key words, phrases or symbols were defined separately to facilitate their modification or maintenance, providing functionality for the user to adapt the terms to their needs.
  - It is possible to enable or disable quality indicators according to the needs of the Analyst or Requirements Engineer and its organization, thus selecting which indicators to evaluate.
- Quality indices are available, at requirement and SRS level, for the different quality properties, which provide statistical information about the quality of each individual requirement and the overall quality of the SRS.
- The tool automatically determines the weights assigned to each indicator, taking into account whether they directly or indirectly affect the quality property in question, following the criterion that the weights of the indicators with a direct impact on quality property must represent 70% and those of indirect incidence should represent 30%.

It is important to highlight that RQV Tool was designed with the aim of mainly covering the following qualities:

- *Easy to use:* the tool has to be used with little effort in terms of user training and time consuming.
- *Flexible / Customizable:* the tool has to be adaptable in order to be effective for particular application domains and allow different quality criteria of organizations.

A. Study Case

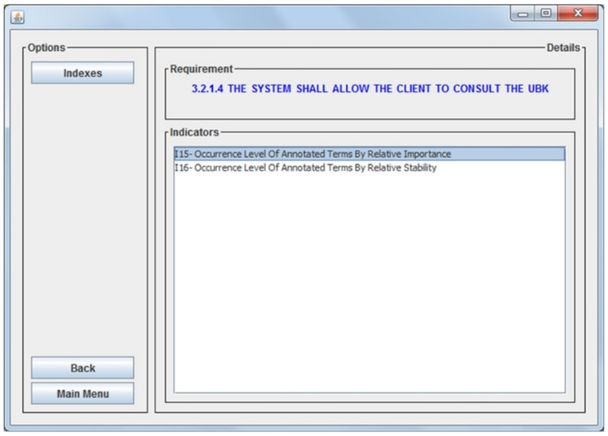
The study case used refers to the specification of a Home Banking System, which corresponds to a technological evolution of some functionalities of the Automated Teller Machine (ATM) available in banks.

Home Banking is the service by which electronic banking transactions can be executed, specifically via private or public networks such as the Internet.

To make use of this service, the financial institution provides various types of access to their computer systems, which allow them to validate the identity of a client and thus allow the use of their services, which are generally limited for security reasons.

Usually in a Home Banking service a client can make balance inquiries, request extracts or account summaries, transfers between accounts of the same financial institution or to third parties, and transaction tracking.

In order to show how RQV Tool analyzes the individual requirements and how it evaluates the indicators and indices at requirements level, the results obtained with the tool are presented for a subset of requirements, selected for being considered interesting due to the indicators that affect.



3.2.1.4 THE SYSTEM SHALL ALLOW THE CLIENT TO CONSULT THE UBK

I15- Occurrence Level Of Annotated Terms By Relative Importance
I16- Occurrence Level Of Annotated Terms By Relative Stability

Fig. 3. Requirement Level Indicators for Requirement 3.2.1.4 in RQVTool.

Finally, the results of indicators and quality indexes at SRS level obtained with RQV Tool are shown.

The requirement :

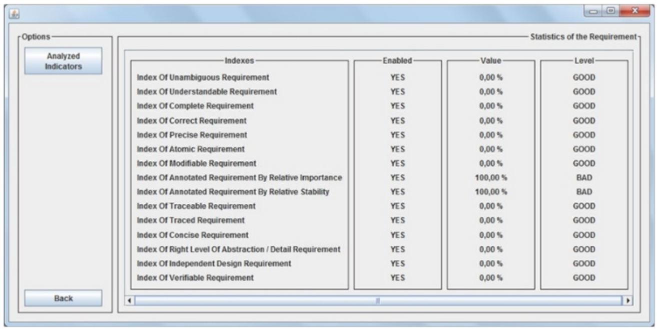
3.2.1.4 The system shall allow the client to consult the UBK."

The indicators detected by RQV Tool to breach its goal are presented in Figure 3. The indexes at the required level detected by the tool were (Figure 4):

- Index of annotated requirement by relative importance = 100% (Bad)
- Index of annotated requirement by relative stability = 100% (Poor)

The rest of the indexes give a value of 0%, that is, its result is Good.

It can be concluded that requirement 3.2.1.4 is not annotated for importance and relative stability.



Indexes	Enabled	Value	Level
Index Of Unambiguous Requirement	YES	0,00 %	GOOD
Index Of Understandable Requirement	YES	0,00 %	GOOD
Index Of Complete Requirement	YES	0,00 %	GOOD
Index Of Correct Requirement	YES	0,00 %	GOOD
Index Of Precise Requirement	YES	0,00 %	GOOD
Index Of Atomic Requirement	YES	0,00 %	GOOD
Index Of Modifiable Requirement	YES	0,00 %	GOOD
Index Of Annotated Requirement By Relative Importance	YES	100,00 %	BAD
Index Of Annotated Requirement By Relative Stability	YES	100,00 %	BAD
Index Of Traceable Requirement	YES	0,00 %	GOOD
Index Of Traced Requirement	YES	0,00 %	GOOD
Index Of Concise Requirement	YES	0,00 %	GOOD
Index Of Right Level Of Abstraction / Detail Requirement	YES	0,00 %	GOOD
Index Of Independent Design Requirement	YES	0,00 %	GOOD
Index Of Verifiable Requirement	YES	0,00 %	GOOD

Fig. 4. Requirement Level Indices for 3.2.1.4 Requirement in RQV Tool.

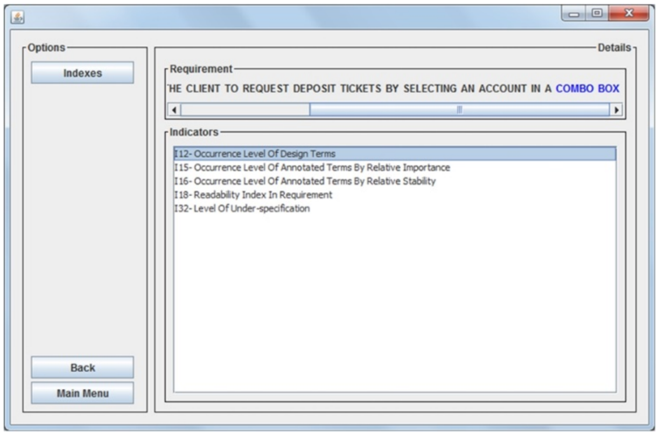
For requirement 3.2.1.18:

3.2.1.18 The system shall allow the client to request deposit tickets by selecting an account in a combo box."

The indicators detected by the tool due to its non-fulfillment of the goal were (Figure 5):

- I-12 level of occurrence in of design terms (Potential Problem): The requirement has the design term "combo box".
- I-15 I-15 level of occurrence of annotation by relative importance terms (Suggestion/Warning): the requirement is not annotated by relative importance.
- I-16 level of occurrence of annotation for relative stability terms (Suggestion/Warning): The requirement is not annotated for relative stability.
- I-18 index of readability at requirement level (Potential Problem): The Flesch readability index of the requirement gives 56.9, so being <60 is considered less legible/understandable.
- I-32 low specification level (Potential Problem): The requirement has a general term ("account") instead of a specific one ("current account").

The rest of the indicators meet their target for this requirement, so they are not detected.



Requirement

HE CLIENT TO REQUEST DEPOSIT TICKETS BY SELECTING AN ACCOUNT IN A COMBO BOX

Indicators

I12- Occurrence Level Of Design Terms
I15- Occurrence Level Of Annotated Terms By Relative Importance
I16- Occurrence Level Of Annotated Terms By Relative Stability
I18- Readability Index In Requirement
I32- Level Of Under-specification

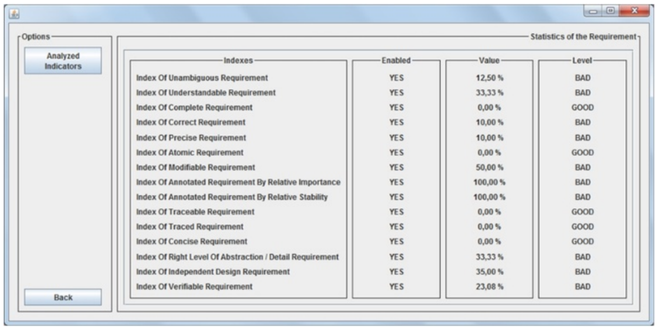
Fig. 5. Requirement Level Indicators of 3.2.1.18 Requirement in RQV Tool.

The indexes at the requirement level identified by the tool were (Figure 6):

- Non-Ambiguous Requirement Index = 12.50% (Bad)
- Understanding Requirement Index = 33.33% (Bad)
- Correct Requirement Index = 10% (Bad)
- Precise Requirement Index = 10% (Poor)
- Modifiable Requirement Index = 50% (Bad)
- Requirement Index Scored by Relative Importance = 100% (Bad)
- Annotated Index of Relative Stability = 100% (Poor)
- Correct Requirement Index Abstraction Level / Detail = 33.33% (Bad)
- Independent Design Requirement Index = 35% (Bad)
- Verifiable Requirement Index = 23.08% (Bad)

The rest of the indexes give a value of 0%, that is, its result is Good.

It can be concluded that requirement 3.2.1.18 is 12.50% ambiguous, 33.33% unintelligible, 10% incorrect, 10% inaccurate, 50% unmodifiable, not annotated by importance and relative stability, 33.33% incorrect Level of abstraction/detail, 35% depending on the design and 23.08% unverifiable.

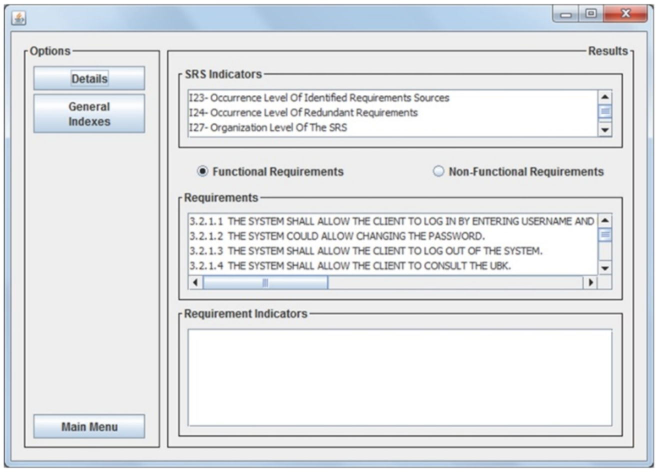


Indexes	Enabled	Value	Level
Index Of Unambiguous Requirement	YES	12,50 %	BAD
Index Of Understandable Requirement	YES	33,33 %	BAD
Index Of Complete Requirement	YES	0,00 %	GOOD
Index Of Correct Requirement	YES	10,00 %	BAD
Index Of Precise Requirement	YES	10,00 %	BAD
Index Of Atomic Requirement	YES	0,00 %	GOOD
Index Of Modifiable Requirement	YES	50,00 %	BAD
Index Of Annotated Requirement By Relative Importance	YES	100,00 %	BAD
Index Of Annotated Requirement By Relative Stability	YES	100,00 %	BAD
Index Of Traceable Requirement	YES	0,00 %	GOOD
Index Of Traced Requirement	YES	0,00 %	GOOD
Index Of Concise Requirement	YES	0,00 %	GOOD
Index Of Right Level Of Abstraction / Detail Requirement	YES	33,33 %	BAD
Index Of Independent Design Requirement	YES	35,00 %	BAD
Index Of Verifiable Requirement	YES	23,08 %	BAD

Fig. 6. Requirement Level Indices of 3.2.1.18 Requirement in RQVTool.

Finally, the quality indicators at the SRS level that the RQV Tool shows for the complete SRS are (Figure 7):

- I-23 level of occurrence of identified origins of requirements (Potential Problem): At least one SRS requirement has no origin, i.e. it does not map to any impact of any LEL symbol.



SRS Indicators

I23- Occurrence Level Of Identified Requirements Sources
I24- Occurrence Level Of Redundant Requirements
I27- Organization Level Of The SRS

● Functional Requirements ○ Non-Functional Requirements

Requirements

3.2.1.1 THE SYSTEM SHALL ALLOW THE CLIENT TO LOG IN BY ENTERING USERNAME AND
3.2.1.2 THE SYSTEM COULD ALLOW CHANGING THE PASSWORD.
3.2.1.3 THE SYSTEM SHALL ALLOW THE CLIENT TO LOG OUT OF THE SYSTEM.
3.2.1.4 THE SYSTEM SHALL ALLOW THE CLIENT TO CONSULT THE UBK.

Requirement Indicators

Fig. 7. SRS Level Indicators in RQVTool.

- I-24 Level of occurrence of redundant requirements (Suggestion/Warning): SRS has redundant requirements.
- I I-27 level of organization of the SRS (Potential Problem): SRS lacks the "Index" section and the "Definitions" and "Product overview" sections are in different order to that set by the ISO / IEC / IEEE standard 29148: 2011.
- I-28 level modifiability of the SRS (Potential Problem): SRS is not modifiable because it is not organized according to the ISO / IEC / IEEE 29148: 2011 (I-27) standard, it has redundant requirements (I-24), it is non-atomic since at least one requirement has connector terms (I-2) and at least one requirement has control flow terms (I-11).
- level of occurrence of acronyms and abbreviations (Potential Problem): The SRS contains the acronym "UBK" which was not defined as an LEL symbol.
- level of occurrence of incomplete sections of the SRS (Potential Problem): there are sections in the SRS that are incomplete as they have the "TBD" mark.
- I-33 level of low reference to the domain vocabulary (Potential Problem): in the SRS is made a reference to

a requirement ID not contained in it and has many entities (nouns in requirements) not defined as LEL symbols.

The rest of the indicators meet their target for SRS, so they are not detected by the tool.

Based on the tool assessment presented in Section 3, it can be observed that the RQV Tool improves many of the issues detected during the analysis:

- RQV Tool is an OpenSource tool, with little or no hardware and software limitation for its installation.
- It accepts the SRS in plain text format (txt), with the advantage that most formats (doc, pdf, xml, etc.) can be easily converted to txt for processing.
- It does not require the participation of the Analyst or Requirements Engineer during the verification process. The Analyst or Requirements Engineer intervenes at the end of the execution, once the statistical information is provided as a result of the verification, to determine and prioritize the improvements to be incorporated into the SRS.
- It considers the LEL (source of specific information in the Engineering of Requirements) as resource of knowledge of the domain.

It solves or improves several of the weaknesses of the evaluation approaches described in section 2, for example: the (metric) calculation procedures of the quality indicators were expressed in such a way that they can be implemented automatically. On the other hand, for the SRS to be accepted as input of the tool is not required to be generated by any specific tool for that purpose. It is only required to comply with the ISO / IEC / IEEE 29148: 2011 (ISO / IEC / IEEE, 2011) standard, which includes IEEE 830: 1998 (R2009) [7].

V. CONCLUSIONS AND FUTURE TRENDS

Within the framework of this paper, "RQVTool" (Requirement Quality Verification Tool), which implements Saavedra's (2016) software requirements quality model, was presented.

The objective of this tool is to automate, within the technical possibilities, the evaluation of the quality properties of the proposed requirements quality model, using LEL as a knowledge resource of the domain for the cases where it is available, that can help in the verification of the quality of the SRS and incorporate the best practices of the already developed until the moment.

The proposal follows the trend and integrates many ideas from other previously developed tools, particularly ARM, QuARS and RQA.

The RQVTool is useful in the SRS verification process, which can occur when creating the SRS or verifying its quality, but before being validated by the client.

RQVTool receives the SRS as an input and, optionally, the LEL, in plain text and English language format, calculates quality indicators and indices, at requirement and SRS level, and outputs quality statistical information.

To perform the quality assessment, the tool uses some control mechanisms: Corpus of words, phrases and/or key symbols, ISO / IEC / IEEE Standard 29148: 2011 (ISO / IEC / IEEE, 2011), which includes IEEE 830 : 1998 (R2009) (IEEE, 2009) and Grammar Rules.

It is implemented with a 3 layer architecture, using in its business layer the Stanford CoreNLP component for the NLP of the SRS.

It improves many of the issues identified in the tool assessment presented in section 3.

It is considered an improvement to incorporate as future work the possibility that the input documents (ERS and LEL) can be in other more usual formats, in addition to plain text (txt), such as: Word (doc), pdf and xml for-mat and the option to change language.

REFERENCES

[1] Nuseibeh, B.; Easterbrook, S.: Requirements engineering: a roadmap. In: Proc. Conference on the Future of Software Engineering, pp. 35-46. (2000).

[2] Pfahl, D.; Lebsanft, K.: Using simulation to analyse the impact of software requirement volatility on project performance. Information and Software Technology, 42(14), pp. 1001-1008. Elsevier Science B.V. (2000).

[3] Fabbrini, F.; Fusani, M.; Gnesi, S.; Lami, G. (2001). An Automatic Quality Evaluation for Natural Language Requirements. In: Proc. 7th International Workshop on Requirements Engineering: Foundation for Software Quality, Interlaken, Switzerland.

[4] Gnesi, S.; Lami, G.; Trentanni, G.; Fabbrini, F.; Fusani, M. (2005). An Automatic Tool for the Analysis of Natural Language Requirements. International Journal of Computer Systems Science & Engineering, 20(1).

[5] Davis, A.; Overmyer, S.; Jordan, K.; Caruso, J.; Dandashi, F.; Dinh, A.; Kincaid, G.; Ledeboer, G.; Reynolds, P.; Sitaram, P.; Ta, A.; Theofanos, M. (1993). Identifying and measuring quality in a software requirements specification. In: Proc. 1st International Software Metrics Symposium, pp. 141-152.

[6] Génova, G.; Fuentes, J.M.; Llorens, J.; Hurtado, O.; Moreno, V. (2011). A Framework to Measure and Improve the Quality of Textual Requirements. Requirements Engineering, 18(1), pp 25-41.

[7] IEEE (2009). Recommended Practice for Software Requirements Specifications. IEEE Standard 830-1998 (R2009), Institute of Electrical and Electronics Engineers.

[8] Loucopoulos, P.; Karakostas, V. (1995). System Requirements Engineering. McGraw-Hill, Inc. New York, NY, USA.

[9] Pohl, K. (2010). Requirements Engineering: Fundamentals, Principles, and Techniques. Springer-Verlag Berlin Heidelberg.

[10] Swathi, G.; Jagan, A.; Prasad, Ch. (2011). Writing Software Requirements Specification Quality Requirements: An Approach to Manage Requirements Volatility. Int. J. Comp. Tech. Appl., 2(3), 631-638.

[11] Wiegers, K.; Beatty, J. (2013). Software Requirements, Third Edition. Redmond, WA: Microsoft Press.

[12] Wilson, W. M. (1997). Writing Effective Requirements Specifications. Software Technology Conference Proceedings.

[13] Lanubile, F.; Shull, F.; Basili, V. R. (1998). Experimenting with Error Abstraction in Requirements Documents. In: Proc. 5th International Symposium on Software Metrics, pp. 114-121.

[14] Schneider, G. M.; Martin, J.; Tsai, W. T. (1992). An Experimental Study of Fault Detection In User Requirements Documents. ACM Transactions on Software Engineering and Methodology, 1(2), pp. 188–204.

[15] Saavedra, R. (2016). Framework Para La Verificación Semiautomática De Especificaciones De Requerimientos De Software, Magister Thesis.