Software Requirements Quality Evaluation: State of the art and research challenges

Roxana Saavedra[1], Luciana Ballejos[2], Mariel Ale[3]

1CIDISI – UTN – Facultad Regional Santa Fe – rsaavedra@frsf.utn.edu.ar
2CIDISI- UTN - Facultad Regional Santa Fe – lballejos@santafe-conicet.gov.ar
3CIDISI- UTN - Facultad Regional Santa Fe – male@frsf.utn.edu.ar

Abstract. Quality models are important tools for quality management. In software development projects, they are useful as predictive tools for assessing the state of the product being developed and the process used. In order to achieve software quality, a high-quality Software Requirements Specification (SRS) is required. This document is generated at the beginning of a software development project, and is used in all stages. Thus, it is essential to evaluate the quality of the SRS in order to be able to take early corrective and enhancement actions. However, assessing the quality of a SRS is not a simple process, mainly by the multitude of proposals, often contradictory, of the attributes to be evaluated and the methodologies used for that purpose. Thus, it is mandatory to consider proven quality models for guiding this evaluation process. Related to this, this work performs an exploratory analysis of various quality models proposed in this area which can be used as a basis for SRS quality evaluation. Moreover, the work is intended to be a compendium of the most important tendencies and strategies in the field that serves as a starting point for developing comprehensive models and tools for quality attributes evaluation in a SRS.

Keywords. quality model; Software Requirements Specification; quality evaluation.

1 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. Some of them also propose

quality models to be considered when evaluating a Software Requirements Specification (SRS), the main deliverable produced in RE, which is used throughout the project [3, 4]. The main goal of this paper is to analyze the state-of-the art in this area, in order to give a more consistent support for software engineers when generating requirements specifications. Moreover, the results might be considered for the future development of tools for supporting automate or semi-automate evaluation of SRS quality.
The paper is organized as follows: Section 2 describes the research methodology used for the study. Section 3 describes existing quality models found in the literature, analyzing their similarities and differences. Meanwhile, Section 4 discusses and analyzes diverse proposals or approaches for evaluating the attributes described in the previous section. Related to this, Section 5 describes influences between these properties. Finally, Section 6 is devoted to discuss conclusions and future trends in this area.

2 Research Methodology

The main goal of a research analysis is to provide an overview of a research area. Thus, a research question or main goal must be defined [5]. In this sense, the purpose of this study is to "understand the trend of quality-related attributes assessment for SRS" by examining published articles and offering, at the same time, insights and future directions in this area.
To this end, the following electronic journal databases were searched to provide an exhaustive bibliographic revision of research papers in the area:

- EBSCO Discovery Service,
- EconPapers,
- Compendex. Engineering Village,
- Engineering Village 2: Referex Engineering ,
- National Digital Library of Theses and Dissertations (NDLTD),
- NTRS: NASA Technical Reports,
- Scopus. Elsevier,
- Social Science Research Network (SSRN),
- ACM Portal,
- IEEE Library.

The search process was performed based on three descriptors: "Software Requirements Specification", "Quality Models" and "Quality Attributes Evaluation". The selection of the descriptors was performed according to their different levels of generality, which allows refining the search results. The search process started from more general (e.g. "quality models") to more specific (e.g. "Software Requirements Specification") phrases, using the facilities of "search within the results" offered by most search engines. Despite the above, and due to the large quantity of papers returned by the databases, the following exclusion criteria were used:

- Given the limited number of indexed publications addressing this particular subject, the consideration of publications only in English. This was done in order to ensure that these articles have been accessed and validated by the widest possible audience.

For the same reasons, non-refereed conference papers, unpublished master and doctoral dissertations were excluded.
- Because the idea of evaluating quality attributes for the SRS is relatively new, the search was restricted from 1990 onwards.

3 Requirements Quality Models

Commonly, a quality model is composed of quality properties to be evaluated by means of quality indicators [4]. Diverse proposals exist specifically for the RE area. Some of them suggest a list of desirable requirements quality properties [3, 4, 6, 7, 8, 9, 10, 11, 12], while others provide a taxonomy of possible defects which can be found when evaluating requirements [13, 14]. Moreover, According to Gnesi [15], a quality model for a SRS also includes syntactic and semantic rules, document structure and sentence structure characteristics.
Considering those proposals describing concrete desirable properties, Table 1 shows those more referenced in the analyzed literature, which are described afterwards.

3.1 Davis et al. Quality Model.

Davis et al. [3] proposed a comprehensive set of quality attributes that must exhibit a SRS. They promote that a SRS, in order to have quality, must be free of any errors that violate these attributes.
The authors present a list of attributes that are a compilation of lists made by other authors. The attributes are:

1) *Unambiguous*: Every requirement stated in the SRS has a unique interpretation.
2) *Complete*: The SRS contains everything that is supposed to make the software; software responses to all possible data inputs in all possible situations; all pages numbered, all figures and tables numbered, named and referenced , all terms defined, all units of measure provided, and all referenced material present; sections completed, i.e., there is not "To Be Determined" (TBD).
3) *Correct*: Every requirement in the SRS represents something required of the system to be built, i.e., every requirement contributes to the satisfaction of some need.
4) *Understandable*: All SRS readers can easily understand the meaning of all requirements with a minimum of explanation.
5) *Verifiable*: There are finite, cost effective techniques that can be used to verify that every requirement in the SRS is satisfied by the system as built.
6) *Internally Consistent*: There is no subset of requirements in the SRS that includes conflicts.
7) *Externally Consistent*: The requirements in the SRS exclude conflicts with any project documentation.
8) *Achievable*: There is at least one system design and implementation that correctly implements all requirements of the SRS.
9) *Concise*: SRS is as short as possible without affecting its quality.
10) *Traceable*: The SRS is written in a way that facilitates the referencing of each requirement.

Table 1. Requirements quality models proposed by different authors.

Quality Properties \ Models	Davis et al. [3]	Fabbrini et al. [4]	Wiegers [6]	IEEE 830-R2009 [7]	Loucoupulus and Karakosta [8]	Wilson et al. [9]	Pohl[10]	Swathi et al. [11]	Génova et al. [12]
Achievable	X		X						
Annotated by Relative Importance	X		X	X		X	X	X	
Annotated by Relative Stability	X			X		X	X	X	
Annotated by Version	X								
At Right Level of Detail	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-Referenced	X								
Design Independent	X				X				X
Electronically Stored	X								
Externally Consistent	X	X	X		X			X	
Internally Consistent	X	X	X	X	X	X	X	X	X
Modifiable	X		X	X		X	X	X	X
Not Redundant	X				X				
Organized	X					X			
Precise	X								X
Prototypable	X								
Reusable	X								
Traceable	X		X	X		X	X	X	X
Traced	X		X	X			X	X	
Unambiguous	X		X	X	X	X	X	X	X
Understandable	X	X				X	X		X
Up to Date							X		
Verifiable	X	X	X	X		X	X	X	X

11) *Modifiable*: If the structure and style of a SRS is such that any changes can be made easily, completely and consistently.
12) *Electronically Stored*: The entire SRS has been produced with a word processor, has been generated from a requirements database, or has otherwise been synthesized from some other form.
13) *Executable/Interpretable/Prototypable*: A software tool must be capable of taking the SRS as input and providing a dynamic behavioral model.
14) *Annotated by Relative Importance*: A reader can easily determine which requirements are most important to customers, which are the next most important, etc.
15) *Annotated by Relative Stability*: A reader can easily determine which requirements are most likely changing, which are the next most likely, etc.

16) *Annotated by Version*: A reader can easily determine which requirements will be satisfied in which program versions.
17) *Not Redundant*: The same requirement is not declared more than once in the SRS.
18) *At Right Level of Detail*: The SRS should be specific enough so that any system built that satisfies the requirements in the SRS, satisfies all user needs, and abstract enough so that all systems that satisfy all users needs also satisfy all requirements.
19) *Precise*: In the SRS, numeric quantities are used whenever possible and appropriate levels of precision are used in all numeric quantities.
20) *Reusable*: SRS sentences, paragraphs and sections can be easily adopted or adapted for their use in future SRS.
21) *Traced*: The origin of each SRS requirement is clear.
22) *Organized*: SRS content is ordered, so that readers can easily locate information and logical relations between adjacent sections.
23) *Cross-Referenced*: Cross-references are used in SRS sections SRS in order to relate requirements to other sections containing requirements: redundant requirements, more abstract or more detailed descriptions of the same requirements, requirements that depend on them, or on which they depend.

Davis et al. [3] point out that although SRS quality is achievable, perfection is not. Any of the quality attributes above mentioned can be achieved, but often at the expense of other attributes. Therefore, on a given project, requirements analysts need to agree on what quality attributes are most important.
As seen in Table 1, this proposal is the most complete, since it considers most quality aspects which must be considered when writing or verifying the SRS. It has also the particularity of separating all quality properties, while some authors group several of them in a single one. For example, Davis et al. [3] distinguish *Traceable* and *Traced* properties, while the standard IEEE 830:2009 [7] joint these two quality attributes in a single property called *Traceable*.
On the other hand, another related to this proposal is that quality attributes are defined in relation to the overall SRS document. Moreover, although the proposal is considered the most comprehensive, it does not provide explanation for quality attributes mention by other authors like *Atomic* [10, 12] and *Up to Date* [10].

3.2 Loucopoulos and Karakostas Quality Model

Loucopoulos and Karakostas [8] proposed six desirable properties for a SRS that should be verified. They are listed below:
1) *Internal Consistency:* No contradictory conclusions can be derived from the SRS.
2) *Non-Ambiguity:* Each requirement in the SRS cannot be interpreted in more than one way.
3) *External Consistency:* Agreement between what is stated in the SRS and what is true in the problem domain must exist.
4) *Minimality:* This is equivalent to Design Independent and, in a way is the opposite to over specification which is the tendency to include in the SRS more than it is necessary. Usually over specification is an attempt to recommend a design solution while the requirement is specified.

5) *Completeness: The* SRS must not omit essential information about the problem domain, which could result in a system that does not meeting users needs.
6) *Redundancy:* A requirement is redundant if it can also be obtained from some other parts of the SRS.

As shown in Table 1, this proposal defines just a few quality properties for the entire requirements document. In particular, *Minimality* is equivalent to *Independent Design* defined by other authors [3, 12], since Loucopoulos and Karakostas [8] define that the *over specification* is the opposite of *Minimality*, which is simply the tendency to include in the SRS more than necessary.

3.3 Wilson et al. Quality Model

Wilson et al. [9] define a set of desirable requirements specification characteristics. They also propose two sets of categories for quality indicators, one concerning SRS and other related to individual requirements, which allow the evaluation of other properties. The quality properties defined by this proposal are:

1) *Complete:* A SRS must precisely define all the real world situations that will be encountered and the responses to them.
2) *Consistent:* In the SRS there is no conflict between requirements that define the behavior of essential capabilities. Moreover, specified behavioral properties and constraints do not have an adverse impact on that behavior.
3) *Correct:* The SRS must accurately and precisely identify the conditions and limitations of all situations that the desired capability will encounter and it must also define proper responses to those situations.
4) *Modifiable:* In a modifiable SRS related concerns must be grouped together, while unrelated ones must be separated.
5) *Ranked:* A ranking according to the stability and/or importance is established in the organization and structure of the SRS.
6) *Testable:* The SRS must be expressed in a way so that pass/fail or quantitative evaluation criteria can be derived from the SRS and/or reference information.
7) *Traceable:* Each requirement in the SRS must be uniquely identified.
8) *Unambiguous:* A requirement in the SRS is unambiguous if it can only be interpreted one way.
9) *Valid:* All project participants, managers, engineers and customer representatives, must be able to understand, analyze, and accept or approve the SRS.
10) *Verifiable:* In order to be verifiable, requirements at one level of abstraction must be consistent with those at another level of abstraction.

In addition, some quality properties not described by the proposal, but which can be evaluated using the categories of quality indicators defined by this author are:

1) Understandable
2) Concise
3) Organized
4) At Right Level of Detail

These quality properties are related to the entire requirements document. In addition, although Wilson et al. [9] do not define the quality properties *Understandable*, *Concise*,

Organized and *At Right Level of Detail*, diverse categories of indicators to evaluate them are presented.
Some features of this quality model are equivalent to others shown in Table 1: *Consistent* refers to *Internally Consistent*, *Ranked* refers to *Annotated by Relative Importance* and *Annotated by Relative Stability*; *Testable* makes reference to *Verifiable*; *Traceable* refers exclusively to *Traceable* (excluding *Traced*). Finally, *Valid* and *Correct* refer to *Correct*.

3.4 Fabbrini et al. Quality Model

Fabbrini et al. [4] propose a quality model for software requirements in natural language compound of high level quality properties that can be evaluated in the requirements document through quality indicators. The high level quality properties are:
1) *Testability:* each requirement in the SRS should be able to be evaluated in a pass/fail or quantitative manner.
2) *Completeness:* the requirements in the SRS should be able to refer to precisely identified entities.
3) *Understandability:* each requirement in the SRS must be fully understood. Also, SRS must be fully understood when read by the user.
4) *Consistency:* each requirement specification in the SRS should be able to avoid potential or actual discrepancies.

The quality properties above do not cover all aspects of software requirements quality, but are specific enough to verify the quality of requirements documents with the support of an automated tool. In addition, this property set includes many of the issues related to the syntax of a requirements document as well as semantic issues.
As shown in Table 1, should be noted that *Testability* refers to *Verifiable* and *Consistency* refers to *Internally Consistent* and *Externally Consistent*.

3.5 Wiegers Quality Model

Wiegers [6] proposes two sets of desirable characteristics, one applicable to individual requirements and other applicable to the entire requirements document. The author considers that it is not enough to have excellent individual requirements and requires, therefore, that the requirements set collected in a SRS must be also of good quality.
According to Wiegers [6], each user, business and functional individual requirement exhibit the following qualities:
1) *Complete*: Each requirement in the SRS must fully describe the functionality to be provided.
2) *Correct*: Each requirement in the SRS must accurately describe the functionality to be built.
3) *Feasible*: Each requirement in the SRS must be possible to implement within the known capabilities and limitations of the system and its operating environment.
4) *Necessary*: Each requirement in the SRS should document a capability that customers really need or that is required for compliance with an external system requirement or a standard.

5) *Prioritized*: An implementation priority must be assigned to each functional requirement, feature, or use case in the SRS, in order to indicate how essential the element is to a particular product release.
6) *Unambiguous*: All readers of a requirement statement should arrive at a single, consistent interpretation.
7) *Verifiable*: Some tests must be defined or other verification methods must be used to determine whether the product properly implements each requirement.

Furthermore, the set of requirements which are collected in a SRS should exhibit the following characteristics:
1) *Complete*: In the SRS should not be absent requirements or necessary information.
2) *Consistent*: The SRS requirements do not conflict with other requirements of the same type or with higher-level business, system or user requirements.
3) *Modifiable*: Each requirement in the SRS must be uniquely labeled and expressed separately from other requirements, and it must appears only once in the SRS. For facilitating SRS modification, a table of contents and an index should be included.
4) *Traceable*: Each requirement in the SRS can be linked backward to its origin and forward to the design elements and source code that implement it, and test cases that verify the correct implementation.

As it can be seen, there is a set of quality attributes relative to individual requirements and another set that refers to entire requirements document. The particularities found in this model compared to the others shown in Table 1 are: *Feasible* refers to *Achievable*; *Necessary* is part of the definition of *Correct*; *Prioritized* refers to *Annotated by Relative Importance*; *Consistent* refers to *Internally Consistent* and *Externally Consistent*; *Traceable* refers to *Traceable* and *Traced*.
It should be noted that Wiegers [6] do not define *Atomic* and *Not Redundant* quality attributes, but he states them as required features to achieve *Modifiable* quality attribute.

3.6 IEEE 830:2009 Quality Model

The IEEE 830:2009 standard [7] defines a set of features for a good SRS. The properties that a SRS should meet are:
1) *Correct*: Every requirement in the SRS is one that the software will satisfy.
2) *Unambiguous*: Every requirement in the SRS has a unique interpretation.
3) *Complete*: The SRS includes the following elements: all significant requirements, definition of the responses of the software to all realizable classes of input data in all realizable classes of situations, and full labels and references to all figures, tables and diagrams and definition of all terms and units of measure.
4) *Consistent*: A SRS is internally consistent if no subset of requirements is in conflict. If a SRS does not agree with some higher-level document, then it is not correct.
5) *Ranked for Importance and/or Stability*: Each requirement in the SRS has an identifier which indicates its importance or stability.
6) *Verifiable*: Every requirement in the SRS is verifiable, i.e., there is some finite, cost effective process with which to check that the software product meets each one.
7) *Modifiable*: The structure and style of the SRS is such that any changes in requirements can be made easily, completely, and consistently while maintaining SRS structure and style.

8) *Traceable*: The origin of each requirement in the SRS is clear and facilitates the referencing of each requirement in future development or enhancement documentation. The standard recommends two types of traceability: Backward traceability and Forward traceability.

The IEEE 830:2009 [7] proposes a set of desirable characteristics relative to the SRS. The attributes listed in this proposal coincide with some of the attributes listed by Davis et al. [3], since the model is based on the 1984 version of the IEEE 830 standard.
As shown in Table 1, should be noted that *Consistent* refers to *Internally Consistent*, if a SRS disagree with some top-level document -such as the system requirements specification- so it is *Not Correct* instead of *Externally Consistent*. Meanwhile, for *Traceable* two types of traceability are defined: *Backward Traceability* (*Traced*) and *Forward Traceability* (*Traceable*). *Ranked for Importance and/or Stability* refers to *Annotated by Relative Importance* and *Annotated by Relative Stability*.

3.7 Pohl Quality Model

Pohl [10] states that quality criteria define the expected quality of requirements documents. The quality of requirements set depends on the quality of individual requirements, as well as the satisfaction of the quality criteria defined for the (set of) requirements. Thus, the quality criteria can be defined for each individual requirement as well as for an entire requirements document, or for a specific section in a document or set of requirements defined in this document.
The quality criteria for individual requirements defined by Pohl [10] are:

1) *Complete*: A requirement is complete if it adheres to the rules and guidelines defined for this type of requirements artifact, and does not omit any information that is relevant to some stakeholder.
2) *Traceable*: A requirement is traceable if the source, evolution, impact and use in subsequent development phases is traceable.
3) *Correct*: A requirement is correct if the relevant stakeholders confirm its correctness and demand that the system must realize the documented requirement completely.
4) *Unambiguous*: A requirement is unambiguous if its documentation allows only one valid interpretation.
5) *Comprehensible*: Requirements content must be easy to comprehend.
6) *Consistent*: A requirement is consistent if the statements inside the artifact do not contradict each other.
7) *Verifiable*: A requirement is verifiable if stakeholders can check whether the implemented system satisfies the documented requirement.
8) *Rated*: A requirement is rated if its relevance and/or stability have been identified and documented.
9) *Up to Date*: A requirement is up to date if it reflects the current status of the system and its context, such as current stakeholder desires or current legal regulations.
10) *Atomic*: A requirement is atomic if describe a single, coherent fact.

The proposal also defines three essential quality criteria to be applied to the whole document:

1) *Completeness*: A SRS is complete if all relevant requirements are specified, and if each documented requirement is specified completely.

2) *Consistency*: A SRS is consistent if each requirement is consistently defined and there are no inconsistencies between the requirements defined in the SRS.
3) *Modifiability*: A SRS is modifiable if its structure and style supports a simple, consistent and complete modification of the requirements, still retaining the structure and style.
4) *Readability*: A SRS is readable if the reader can easily extract and comprehend its content.

As it can be realized, Pohl [10] defines two sets of quality attributes, one relative to individual requirements, and the other refers to the entire requirements document.
As shown in Table 1, some features found in this quality model are: the property that many authors define as *Understandable* is defined here as two different attributes, on one hand, *Comprehensible* (on individual requirements) and, on the other, *Readability* (relative to the entire requirements document). Moreover, *Consistent* and *Consistency* refer to *Internally Consistent* for individual requirements and entire requirement document respectively, while *Traceable* refers to *Traceable* and *Traced. Rated* refers to *Annotated by Relative Importance* and *Annotated by Relative Stability*.
It should be noted that *Up to Date* quality attribute was only found as such in this quality model. *Not Redundant* is not defined as a quality attribute, but described as a characteristic required to achieve other quality attributes such as *Modifiable* and *Readability*.

3.8 Swathi et al. Quality Model

Swathi et al. [11] propose some characteristics that a good SRS should exhibit. Listed below are the desirable characteristics:
1) *Correct*: The SRS correctly reflects the actual needs.
2) *Unambiguous*: Every requirement in the SRS has a unique interpretation.
3) *Complete*: In the SRS are declared all conditions under which the requirement applies are declared in the SRS and every requirement is expressed a whole idea or statement.
4) *Consistent*: This refers to SRS internal consistency, and must ensure that it does not conflict with other documents.
5) *Verifiable*: Every requirement in the SRS is verifiable if there is a finite cost-effective process with which it can be checked that the software product meets the requirements.
6) *Traceable*: The origin of software requirements is clear and facilitates the referencing of each requirement in future development or enhancement document.
7) *Modifiable*: The structure and style of the SRS are such that any changes to the requirements can be made easily, completely, and consistently while retaining the structure and style.
8) *Ranked for Importance and/or Stability*: Each requirement in the SRS must be ranked for importance and/or stability.

Swathi et al. [11] define a set of desirable characteristics relative the entire software requirements document. If this quality model is compared with the proposed by the IEEE 830:2009 [7], it can be found that the quality properties are the same in both models, but there is one difference in the definition of consistency. Swathi et al. [11] defines *Consistent* as *Internally Consistent* and *Externally Consistent* and IEEE 830:2009 [7]

proposal defines *Consistent* as Internally *Consistent* and, as described above, if a SRS does not agree with some top-level document, then this is *Not Correct* instead of *Externally Consistent*.
As shown in Table 1, note that *Traceable* refers to *Traceable* and *Traced*, and *Ranked for Importance and/or Stability* refers to *Annotated by Relative Importance* and *Annotated by Relative Stability*. Furthermore, these authors do not define quality attributes *Not Redundant*, *Atomic,* and *At Right Level of Detail*, but they are taken into account when guidelines to be considered in the requirements documentation are described.

3.9 Génova et al. Quality Model

Génova et al. [12] propose qualitative desirable requirements properties, which depend on a subjective judgment, and quantitative measurable indicators, based on the objectives characteristics of requirements.
Listed below are the desirable properties of a SRS proposed by Génova et al. [12]:

1) *Validability*: The client must be able to confirm that the requirements in the SRS express the system that answers his/her needs.
2) *Verifiability*: The engineer must be able to check that the developed system meets the one specified in the SRS.
3) *Modifiability*: The requirements in the SRS must be modifiable to facilitate the system modification for maintenance.
4) *Completeness*: The SRS requirements set covers all needs.
5) *Consistency*: There are no contradictions among requirements in the SRS.
6) *Understandability*: The requirements in the SRS are properly understood without difficulty.
7) *Unambiguity*: Every requirement in the SRS has a unique interpretation.
8) *Traceability*: Every requirement in the SRS has an explicit relationship with design, implementation and testing artifacts.
9) *Abstraction*: Software requirements tell what the application must do without telling how it must do it, i.e., avoid excessive technical details about the implementation.
10) *Precision*: All used terms in the SRS are concrete and well-defined.
11) *Atomicity*: Every requirement in the SRS is clearly determined and identified, without mixing it with other requirements.

Génova et al. [12] propose a list of quality attributes referring to entire requirements document. As shown in Table 1, some features found in this quality model are: *Validability* refers to *Correct*, *Consistency* refers to *Internally Consistent*, *Traceability* refers to *Traceable*, and *Abstraction* refers to *Independent Design*.

4 Quality Properties Evaluation

The diverse proposals for evaluating SRS quality attributes previously analyzed can also be grouped in relation to the approach used for giving some concrete idea on how properties can be evaluated and verified. Table 2 shows the approaches identified as result of this research. For more detail, Table 3 shows the authors with concrete proposal on attributes evaluation techniques.

Table 2. Approaches identified for concrete proposals on attribute evaluation techniques.

APPROACH / PROPERTY	Vocabulary or Language			Relations between Requirements and Artifacts		SRS Structure			Software Requirements Themselves		Jointly Evaluation of Requirements Groups
	Use of Domain Vocabulary	Use of Domain Knowledge	Use of Natural Language Patterns	Use of Overlap between Requirements	Use of Requirements Dependencies	Use of Text Structure in the SRS	Specific Characteristics of the SRS	Achievable Features from SRS	Use of Specific Requirement Characteristics	Use of Deductible Features from a Requirement	Use of Metrics that Calculate the Percentage of Requirements that Meet the Attribute in Question
Achievable								X			
Annotated by Relative Importance, Relative Stability or Version			X								X
Atomic	X	X	X	X	X				X		
Complete		X	X				X				
Concise							X				
Correct		X							X		X
Cross-Referenced											
Design Independent			X					X			
Electronically Stored							X				
Externally Consistent											X
Internally Consistent		X		X					X		
Modifiable							X				
Not Redundant				X							X
Organized						X	X				
Precise	X		X								
Prototypable								X			
Reusable							X	X			
Right Level of Abstraction/Detail						X					
Traceable				X	X				X		
Traced									X		
Unambiguous	X	X	X	X							
Understandable	X	X	X	X	X	X					
Verifiable (Testable)			X						X	X	

Some approaches considering the use of **vocabulary or language** are:

- *Use of domain vocabulary*, related to the use of user vocabulary (glossary) in requirements descriptions. This approach is proposed for the evaluation of Unambiguous and Understandable [16, 17], Atomic and Precise [12] quality attributes.
- *Use of domain knowledge*, which considers interpretation and domain semantic knowledge (some authors use ontologies as knowledge resource). This approach is included in assessment techniques proposed for Unambiguous [3, 18, 19], Complete [3, 18, 19, 20, 21, 22, 23], Internally Consistent [3, 23], Correct [18, 19, 20, 21], Understandable [3] and Atomic [21] quality attributes.

Table 3. Authors with Concrete Proposal on Attributes Evaluation Techniques.

Property \ Authors	Davis et al. (1993) [3]	Sinha and Popken (1996) [23]	Wilson et al. (1997) [9]	Hammer et al. (1998) [25]	Durán et al. (2001) [16]	Fabbini et al. (2001) [4]	Durán et al. (2002) [17]	Kaiya and Saeki (2005) [18]	Kaiya and Saeki (2006) [19]	Tjong et al. (2007) [24]	Verma and Kass (2008) [22]	Dzung and Ohnishi (2009) [20]	Hu et al. (2010) [21]	Swathi et al. (2011) [11]	Génova et al. (2013) [12]
Achievable	X														
Annotated by Relative Importance, Relative Stability or Version	X						X								
Atomic													X	X	X
Complete	X	X		X	X	X	X	X	X		X	X	X		
Concise	X		X	X											
Correct	X							X	X			X	X		X
Cross-Referenced															
Design Independent	X														X
Electronically Stored	X														
Externally Consistent	X														
Internally Consistent	X	X				X		X	X		X	X	X		
Modifiable	X														
Not Redundant	X											X			
Organized			X		X		X								
Precise															X
Prototypable	X														
Reusable	X														
Right Level of Abstraction/Detail			X												
Traceable	X				X		X								X
Traced					X		X								
Unambiguous	X		X	X	X		X	X	X	X					X
Understandable	X		X	X	X	X	X								X
Verifiable (Testable)	X			X		X									X

- *Natural language patterns detection* using keywords, key phrases and/or symbols as evidence of the occurrence for certain attributes. This approach is proposed for evaluation techniques related to Unambiguous [9, 12, 24, 17, 25], Complete [4, 16, 17, 25], Verifiable [4, 25], Annotated by Relative Importance and Relative Stability [17], Understandable [4, 9, 12, 25], Atomic [11, 12], Design Independent and Precise [12] quality attributes.

Other approaches analyze **relations between requirements and artifacts** in SRS, in order to evaluate diverse attributes. Among them:

- Those proposing *overlapping between requirements* consider requirements referring to the same subject, where contradictions between requirements, redundancy when there is a unnecessary repetition, or simple coupling (which implies some kind of dependency relationship) can be distinguished. This approach is proposed for the

evaluation of Internally Consistent [18, 19, 20, 21, 22], Not Redundant [20], Unambiguous, Traceable, Understandable and Atomic [12] quality attributes.
- Other approach considers the evaluation of *requirements dependencies* with other requirements or other artifacts of the development process. In this approach, evaluation techniques are proposed for quality attributes such as Traceable, Understandable and Atomic [12].

Moreover, diverse approaches are based in the analysis of **SRS structure** for evaluating some attributes. Between them:
- Approach analyzing *text structure in the SRS* considers requirements found in each SRS hierarchical level. This approach is considered when proposing evaluation techniques for quality attributes such as Understandable (considering the degree of nesting between requirements) [12], Organized (number of requirements at each hierarchical level), and Right level of Abstraction/Detail (considers the specification depth) [9];
- Proposing the consideration of *SRS specific characteristics* includes the analysis of the presence of *sections*, *table of contents* and *index*, SRS *size*, etc. This approach, is applied in assessment techniques related to quality attributes like Complete (considering that certain sections are present in the SRS) and Organized (considering whether the required sections are present in the required order and with the content required) [16, 17], Concise (considering the size of the SRS) [3, 9, 25], Modifiable (considering the presence of table of contents and index, and the degree of cohesion and coupling of SRS sections), Electronically Stored (considers SRS volume that has been electronically stored) and Reusable (considers paragraphs in the SRS that exhibit reuse properties) [3].
- Approach describing *achievable features from SRS document* includes the evaluation of actual solution system designs, if it is a 'single' system, etc. In this approach, a set of evaluation techniques for diverse quality attributes is proposed. The attributes are: Design Independent (number of actual solution system designs that satisfy all requirements of SRS), Achievable (considering the existence of a single system), Prototypable (considering whether SRS can be partially written in a executable, interpretable or language), and Reusable (considering whether the content of the SRS has been used in subsequent SRS) [3].

The analysis of the software **requirements themselves** is performed by diverse proposals. In this sense:
- The *use of specific requirements characteristics* approach considers the use of explicit references, unique identifiers, cross-references, versions and requirements size. Also, this approach is used for evaluating quality attributes like Internally Consistent (considering the use of explicit references in a requirement) [4], Traceable (considering using of requirement unique identifier) [3, 16, 17], Traced (considering the use of requirement source unique identifier [16, 17]), Correct and Verifiable (considering the number of versions of a requirement) [12], Atomic (considering the size of a requirement) [12].
- The analysis of *deductible features from a requirement* includes the evaluation of required cost and time for verifying a requirement. Moreover, in this approach, Davis et al. [3] propose assessment techniques for the quality attribute Verifiable, considering the cost and time required to verify the requirement.

Finally, some approaches which consider the **jointly evaluation of requirements groups**, propose *metrics that calculate the percentage of requirements that meet the analyzed attribute*. In this approach, Davis et al. [3] propose diverse metrics for evaluating quality attributes such as Externally Consistent, Correct, Annotated by Relative Importance, Relative Stability and Version, and Not Redundant. The metrics proposed, in general, consider the ratio between the requirements satisfying some specific attribute, over the total number of requirements in the SRS. However, no concrete proposals are described by the authors, in order to give guidance over how considering or identifying the requirements which satisfy each attribute.
Davis et al. [3] indicate that Organized, Cross-Referenced, Traced and Right Level of Abstraction/Detail attributes cannot be measured for different reasons. The authors affirm that "organization" is purely subjective and, thus, it cannot be measured. On the other hand, Cross-Referenced cannot be measured because there is no way to determine how many cross-references are appropriate in an SRS. Also, measuring the level of traceability is not possible, making it impossible to measure Traced attribute. In relation to Right Level of Abstraction/Detail, measuring the appropriateness of the SRS level of abstraction is highly scenario-dependent. Moreover, for Up to Date attribute was not found in the literature any evaluation proposal.

5 Influences between Quality Properties

The possibility of creating a SRS of reasonable quality exists. Nevertheless, most quality properties mentioned in this paper have positive or negative incidence on other properties (see Figure 1). Because of this, it is necessary to determine which quality attributes are most important to the project, in order to achieve them. Thus, influences which can be detected between quality attributes described above are now described:

- Generally, requirements considered *unverifiable* are *ambiguous* [6, 7, 12], *incomplete* or *inconsistent* [6, 12], or *unfeasible* requirements [6].
- The elimination of *ambiguity* in the SRS requires adding formality, which is not understood by people who are not computer experts, e.g., users or customers [3, 6, 12].
- The less *ambiguous* is the SRS, the easier its *modification* [12].
- If a SRS is not *complete* in terms of objectives, rules, facts, and constraints of the problem domain, then it is considered *incorrect* for that domain [8, 12].
- Exceeding the *completeness* of the SRS causes losing *conciseness* [3].
- It is not always possible to significantly reduce the SRS *size* without negatively affecting other quality attributes [3].
- If a SRS is *inconsistent*, then the client cannot confirm that requirements effectively express the system that responds to their needs (*correctness)* [12].
- A SRS is more *modifiable* if it is *traceable*, *organized*, *cross-referenced* and *electronically stored* [3].
- Traceability facilitates *correctness* [7] and *verifiability* [12].
- Requirements not *understood* are not *verifiable* and its *correctness* may not be validated [12].
- A non-*atomic* requirement has the risk of excessive detail, and thus, may affect *design independence* [12].

- The *redundancy* is often used to increase *readability* of the SRS. Nevertheless, redundancy can negatively affect the *modifiability*, since not changing all occurrences of a redundant requirement generates an *inconsistent* SRS [3].
- *Atomicity* influences requirements *precision* and *traceability* [12].
- *Design independence* can help *verifiability*. The presence of requirements technical details is more difficult to *verify* and *modify* [12].
- A more *precise* language helps to write more *complete*, *consistent*, *understandable* and *unambiguous* requirements [12].

Figure 1 resumes the influences among quality attributes previously described. In addition, it can be noted that no dependencies can be found for some attributes, due to it is possible to reach them without affecting other attributes. They are: Reusable, Up to Date, Annotated, Prototypable and At Right Level of Detail.

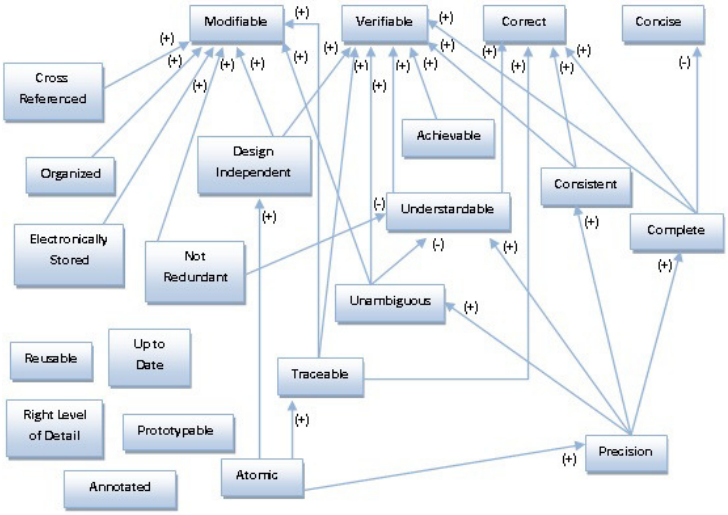

Fig. 1. Influence between quality properties.

6 Conclusions and Future Work

In this work, an exploratory analysis of the various software requirements quality models that can be used to perform a evaluation of SRS quality is presented.
Diverse quality models exist in bibliography proposing attributes and properties to evaluate SRS. As was detailed, several quality properties are present in those models. Many of them coincide in their names and descriptions, but others not. This is due to the effort made by some authors in order to consider diverse issues related to SRS structure or content, and thus, generating new attributes. Moreover, analyzed quality models differ in scope, since some authors propose quality properties to SRS, while others contemplate further quality properties for individual requirements in the SRS.
It can be also observed that some quality models are more likely to be automatically verified through tangible indicators, because they propose more specific quality proper-

ties and measurement methods, for example, Fabbrini et al. [4] quality model. Furthermore, in Table 1 it is shown that there are more comprehensive quality models that cover most of the quality aspects of software requirements specification [3], while others are more limited, suggesting only some desirable characteristics [4, 8].
Other particularity detected is that some authors detail each separately quality property, giving more specificity to the explanation, while others group several of them on a single property. For example, Davis et al. [3] distinguish the *Traceable* and *Trace* properties, while the IEEE 830: 2009 [7] brings together these two attributes in *Traceable* property. Moreover, not all the quality models refer in the same way the same property. For example, Génova et al. [12] describes *Abstraction* to refer to *Independent Design*.
Analyzing the properties proposed in each quality model, it can be observed that most of them provide the attributes: *Complete*, *Internally Consistent*, *Unambiguous*, *Verifiable*, *Correct*, *Traceable* and *Modifiable*. On the other hand, some quality attributes are considered by a unique proposal, including: *Cross-Referenced*, *Prototypable* and *Up to Date*. Moreover, some authors define *Not Redundant*, *Atomic* and *At Right Level of Detail* as properties, while others only suggest some guidelines to consider them in the requirements documentation.
Finally, it can be concluded that the definition of a quality model including desirable properties applicable to the entire requirements document and to the individual requirements contained in it is required. This will allow not only the consistent and concrete definition of each property, but also the definition of quantifiable indicators. This constitutes the main goal for future research, in order to analyze and propose possible metrics and methods to evaluate properties compliance level and to develop supporting tools for implementing metrics, so automatic assessing can be performed for obtaining quality SRS.

7 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. Davis, A.; Overmyer, S.; Jordan, K.; Caruso, J.; Dandashi, F.; Dinh, A.; Kincaid, G.; Ledeboer, G.; Reynolds, P.; Sitaram, P.; Ta, A.; Theofanos, M.: Identifying and measuring quality in a software requirements specification. In: Proc. 1st International Software Metrics Symposium, pp. 141-152. (1993).
4. Fabbrini, F.; Fusani, M.; Gnesi, S.; Lami, G.: An Automatic Quality Evaluation for Natural Language Requirements. In: Proc. 7th International Workshop on Requirements Engineering: Foundation for Software Quality, Interlaken, Switzerland. (2001).
5. Petersen, K.; Feldt, R.; Mujtaba, S.; Mattsson, M.: Systematic Mapping Studies in Software Engineering. In: Proc. 12th International Conference on Evaluation and Assessment in Software Engineering (EASE), Giuseppe Visaggio, Maria Teresa Baldassarre, Steve Linkman, and Mark Turner (Eds.). British Computer Society, Swinton, UK, UK, pp. 68-77. (2008).
6. Wiegers, K.: Software Requirements, Second Edition. Microsoft Press. (2003).
7. IEEE Recommended Practice for Software Requirements Specifications. IEEE Standard 830-1998 (R2009), Institute of Electrical and Electronics Engineers. (2009).

8. Loucopoulos, P.; Karakostas, V.: System Requirements Engineering. McGraw-Hill, Inc. New York, NY, USA. (1995).
9. Wilson, W.M.; Rosenberg, L.H.; Hyatt, L.E.: Automated analysis of requirement specifications. Proceedings of the 19th International Conference on Software Engineering. (1997).
10. Pohl, K.: Requirements Engineering: Fundamentals, Principles, and Techniques. Springer-Verlag Berlin Heidelberg. (2010).
11. Swathi, G.; Jagan, A.; Prasad, Ch.: Writing Software Requirements Specification Quality Requirements: An Approach to Manage Requirements Volatility. Int. J. Comp. Tech. Appl., **2**(3), 631-638. (2011).
12. Génova, G.; Fuentes, J.M.; Llorens, J.; Hurtado, O.; Moreno, V.: A Framework to Measure and Improve the Quality of Textual Requirements. Requirements Engineering, **18**(1), pp 25-41. (2013).
13. Lanubile, F.; Shull, F.; Basili, V. R.: Experimenting with Error Abstraction in Requirements Documents. In: Proc. 5th International Symposium on Software Metrics, pp. 114-121. (1998).
14. Schneider, G.M.; Martin, J.; Tsai, W. T.: An Experimental Study of Fault Detection In User Requirements Documents. ACM Transactions on Software Engineering and Methodology, **1**(2), pp. 188–204. (1992).
15. Gnesi, S.; Lami, G.; Trentanni, G.; Fabbrini, F.; Fusani, M.: An Automatic Tool for the Analysis of Natural Language Requirements. International Journal of Computer Systems Science & Engineering, **20**(1). (2005).
16. Durán, A.; Bernárdez, B.; Ruiz, A.; Toro, M.: An XML–based Approach for the Automatic Verification of Software Requirements Specifications. In: Proc. 4th Workshop on Requirements Engineering, pp. 181-194. (2001).
17. Durán, A.; Ruiz-Cortés, A.; Corchuelo, R.; Toro, M.: Supporting requirements verification using XSLT. In: Proc. IEEE Joint International Conference on Requirements Engineering, pp. 165–172. (2002).
18. Kaiya, H.; Saeki, M.: Ontology Based Requirements Analysis: Lightweight Semantic Processing Approach. In: Proc. 5th International Conference on Quality Software, pp. 223-230. (2005).
19. Kaiya, H.; Saeki, M.: Using Domain Ontology as Domain Knowledge for Requirements Elicitation. In: Proc. 14th IEEE International Requirements Engineering Conference, pp. 189-198. (2006).
20. Dzung, D.; Ohnishi, A.: Ontology-based Reasoning in Requirements Elicitation. In: Proc. 7th IEEE Int. Conf. on Software Engineering and Formal Methods, pp. 263-272. (2009).
21. Hu, H.; Zhang, L.: Ye, C.: Semantic-based Requirements Analysis and Verification. In: Proc. International Conference on Electronics and Information Engineering, pp. 241-246. (2010).
22. Verma, K.; Kass, A.: Requirements Analysis Tool: A Tool for Automatically Analyzing Software Requirements Documents. In: Proc. 7th International Conference on The Semantic Web, pp. 751–763. (2008).
23. Sinha, A.P.; Popken, D.: Completeness and Consistency Checking of System Requirements: An Expert Agent Approach. Expert Systems With Applications, **11**(3), pp. 263-276 (1996).
24. Tjong, F.; Hartley, M.; Berry, D.M: Extended disambiguation rules for requirements specifications. In: Proc. 10th Workshop in Requirements Engineering, pp. 97-106. (2007).
25. Hammer, T.; Huffman, L.; Rosenberg, L.: Doing Requirements Right the First Time. CROSSTALK, The Journal of Defense Software Engineering, pp. 20-25. (1998).