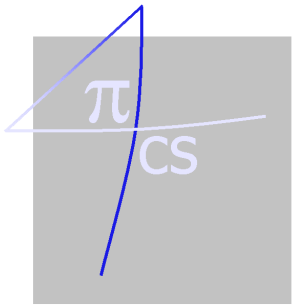
Fraunhofer Institut
Experimentelles
Software Engineering

Requirement Metrics

An initial literature survey on measurement approaches for Requirement Specifications



Authors:
Maricel Medina Mora
Christian Denger

Supported by Provincia Autonoma di Trento in the “Forpics” project.

IESE-Report No. 096.03/E
Version 1.0
October 1, 2003

A publication by Fraunhofer IESE

Fraunhofer IESE is an institute of the Fraunhofer Gesellschaft.
The institute transfers innovative software development techniques, methods and tools into industrial practice, assists companies in building software competencies customized to their needs, and helps them to establish a competitive market position.

Fraunhofer IESE is directed by
Prof. Dr. Dieter Rombach
Sauerwiesen 6
D-67661 Kaiserslautern

Abstract

High quality requirements documents are an essential success factor for software development projects. Requirements are the starting point for each development step, thus requirements of bad quality can have severe consequences for the whole development process. Therefore, it is important to evaluate the quality of the requirements as early as possible. Metrics are a means to evaluate the quality of the requirements. Requirements can have different quality aspects like readability, completeness, verifiability, etc, but how can a requirements engineer decide whether a quality attribute is fulfilled or not. This document presents the results of a literature survey on software requirements metrics. The survey was focused on metrics that are defined to measure the overall quality of a requirements document. Of special interest were metrics that can be applied in requirements documents written in natural language and written with Use Case.

Keywords: requirement engineering, requirement specification document, use cases, quality, metrics, Forpics.

Table of Contents

1	**Introduction**	**1**
2	**Software Requirement Specification and Metrics**	**3**
3	**Literature Survey Background**	**6**
4	**Summary of Most Relevant References**	**8**
4.1	Identifying and Measuring Quality in a Software Requirements Specification	8
4.1.1	Overview	8
4.1.2	Metrics	9
4.1.3	Requirements Language	16
4.1.4	Tools	17
4.1.5	Related Authors References	17
4.2	Automated Quality Analysis of Natural Language Requirement Specification	17
4.2.1	Overview	17
4.2.2	Metrics	18
4.2.3	Requirements Language	22
4.2.4	Tools	23
4.2.5	Related Authors References	23
4.3	Metrics for Requirement Engineering	23
4.3.1	Overview	23
4.3.2	Metrics	24
4.3.3	Requirements Language	27
4.3.4	Tools	28
4.3.5	Related Authors References	28
4.4	Application of Linguistic Techniques for Use Case Analysis	28
4.4.1	Overview	28
4.4.2	Metrics	29
4.4.3	Requirements Language	31
4.4.4	Tools	31
4.5	Advanced Use Case Modeling	32
4.5.1	Overview	32
4.5.2	Metrics	33
4.5.3	Tools	37
4.5.4	Requirement Language	37
4.5.5	Related Authors References	37

5	**Conclusions and Future Work**	**38**
6	**Literature Survey References**	**42**
7	**References**	**45**

1 Introduction

Each software development project starts with a set of requirements that are the basis for the whole development cycle. All later process activities are directly or indirectly based on these requirements. It is important to document these requirements so that all stakeholders in the project have a common understanding of the system to be build.

As the requirements are the basis for the complete software development process, it is important to have high quality requirements. A requirements of bad quality, e.g., the requirements is ambiguous or incompletely specified, can have disastrous consequences for the whole project if it is propagated to the later development steps. So the question that arises is: Are you building the "right" software requirement specifications "right"?. The answer to the *"right" software requirement specification* relies on the understandability desires and expectations of the stakeholders; that is, it is about the quality of the content of the requirement specification. Building the *"right" software requirement specifications* ***"right"*** refers to the way the specifications are written; that is, it is about processes, standards, methodologies, tools and techniques that are used to manage the requirements specification process.

There are hundreds of articles that propose different approaches to manage the requirement process. Many of them suggest templates for the requirements, define quality criteria of the statements and emphasize the importance of the participation of the stakeholders on the analysis of the requirement to assure and warranty the quality of them. However, how can a requirements engineer be sure that what he has specified is right and it is being specified on the right way and consequently build the right software? In [RR99] the answer is presented:

"*I often say that when you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meagre and unsatisfactory kind".*

Measurement is the key to know if we are reaching the goal of building high quality requirements. It is not just proposing processes or templates; it is about knowing quantitatively if those processes are helping to build high quality requirements documents. On the other hand, measurement is not just providing numbers; it should also provide significance to those numbers. It is about knowing the meaning of the data that is being retrieved and presented, under a certain context. Knowing if we are building high quality requirement specifi-

cation can only be guaranteed through an objective and quantitative measurement program.

The question that arises is now, are there defined metrics to measure the quality of software requirement specification? In order to answer this question, a literature survey was carried out. The survey focused on metrics that can be applied on software requirements specifications that either are written in natural languages or with Use Case. The goal of this survey is to provide an overview of existing metric approaches to evaluate the requirements quality. Moreover, the survey should reveal quality aspects of requirements and requirements documents that can be measured in a quantitative way. Related to this goal is the question, which of quality aspects of requirements (documents) like completeness, consistency, traceability, readability, correctness, as proposed by standards [IEEE830], can be measured with existing metrics and techniques.

The results of this survey can then be used as a starting point to define a quality model for requirements documents and to provide a set of metrics that allows the requirements engineers to judge whether the quality of the requirements documents is on an appropriate level. Thus, the survey results can be used to decide where additional metrics should be defined by comparing the common quality aspects of the requirements documents and the existent metrics for these attributes. However, these questions are not addressed in this report, as it is in a first step necessary to gain an overview of existing metric approaches for requirements documents.

This report contains the results of the literature research and a summary of the most relevant references that provide clear guidelines to measure the quality of the requirements. The remainder of this report is structured in five sections: Section 2, *Software Requirement Specification and Metrics*, which provides an introduction to the general topic of software requirement specification, the definition of quality and metrics; to evaluate their quality. Section 3, *Literature Survey Background*, which presents the background and the process followed to carry out the literature survey. Section 4, *Summary of Most Relevant References*, which provides a summary of the most relevant articles found through the research which focus on measuring quality attributes of the requirements; Section 5, *Conclusions and Future Work*, which summarizes the results of the survey and provide a guideline to create the "Metric Model" to be applied on software requirement specifications. The list of all references found during the literature survey is presented in Section 6, *Literature Survey References.* Finally, the references used in this technical report are presented in Section 7.

2 Software Requirement Specification and Metrics

Before describing the different approaches on how to measure the quality of software requirements specifications the basic terms used in the report should be defined.

[IEEE SW May/June 2003], based on IEEE STD 830-1984, defines a ***software requirement specification*** as

"*A document that clearly and precisely describes each essential requirement (functions, performance, design constrains, and quality attributes) of the software and the external interfaces. Each requirement is a defined in such a way that a prescribed method (such as inspection, demonstration, analysis, or test) can objectively verify its achievement*".

[AM00], based on IEEE STD 1233a-1998, defines a ***well-formed requirement*** as

"*A statement of system functionality (a capability) that can be validated, that must be met or possessed by a system to solve a customer problem or to achieve a customer objective and that is qualified by measurable condition and bounded by constraints*"

Additional to the above definition, [IEEE STD 830-1993] presents eight characteristics of a good SRS. These characteristics are:

- Correct: An SRS is correct if, and only if, every requirement stated therein is one that the software shall meet
- Unambiguous: An SRS is unambiguous if, and only if, every requirement stated therein has only one interpretation. As a minimum, this requires that each characteristic of the final product be described using a single unique term. In cases where a term used is an particular context could have multiple meanings, the term should be included in a glossary where its meaning is made more specific.
- Complete: An SRS is complete if, and only if, it includes the following elements: a) All significant requirements, whether relating to functionality, performance, design, constraints, attributes, or external interfaces. In particular any external requirements placed by a system specification should be acknowledged and treated; b) Definition of the responses of the software to all realizable classes of input data in all realizable classes of situations. Note that it is important to specify the responses to both valid and invalid input

values; c) Full labels and references to all figures, tables, and diagrams in the SRS and definition of all terms and units of measure.

- Consistent: Consistency refers to internal consistency. A SRS is internally consistent if, and only if, no subset of individual requirements described in it conflict.
- Ranked for importance and/or stability: An SRS is ranked for importance and/or stability if each requirement in it has an identifier to indicate either the importance or stability of that particular requirement.
- Verifiable: An SRS is verifiable if, and only if, every requirement stated therein is verifiable. A requirement is verifiable if, and only if, there exists some finite cost-effective process with which a person or machine can check that the software product meets the requirement.
- Modifiable: An SRS is modifiable if, and only if, its structure and style are such that any changes to the requirements can be made easily, completely, and consistently while retaining the structure and style.
- Traceable: An SRS is traceable if the origin of each of its requirements is clear and if it facilitates the referencing of each requirement in future development or enhancement documentation.

In the following we perceive these attributes as a base set of quality aspects a requirements specification should have.

[CL95] defines a ***software metric*** as:

"A measurement derived from a software product, process, or resources. Its purpose is to provide a quantitative assessment of the extent to which the product, process, or resource possesses certain attributes."

Metrics should always be ***meaningful*** and ***useful*** in order to be valuable in a certain setting. *Meaningful* includes

- a clear definition of the algorithm to calculate the metric as well as how the related data is going to be collected (e.g.: through an inspection, with a database statistics, with questions regarding developers opinions, etc),
- a interpretation of the metric; that is, what does the value of the metric tells us (e.g., what does it mean to have a 30 page requirements document)
- a definition of the usage of the metric; that, is for which purpose are we collecting the metric. (e.g.: why have we collected the number of pages of the requirements document)

A metric is perceived as *useful* if the metric can be used to measure and to control products, processes and resources. It is very important that metrics are understood as means to control software development **not** as means for personnel evaluation. Not all the metrics, even though they are well defined, structured and have a purpose, provide value to reach the goal of delivering a high quality SRS and are then not useful.

3 Literature Survey Background

The literature research was conducted mainly through digital libraries and by means of accessing the available literature at IESE Library. Some of the phrases and key words used during the survey were "requirement metrics", "requirement analysis", "use case quality", "use case quality attributes", "use cases and metrics". The survey generated more than 50 references, which on one-way or the other, were related to the main topic: metrics on software requirement specifications. After reading quickly every available reference, several categories were created and the references were grouped according the one that most fit for them. The categories created are described in Table 1.

CATEGORY	DESCRIPTION
Metrics	Any information regarding metrics (theory, methods, etc.)
Software Metrics	Any reference which focuses on metrics applied on the Software field
Requirements Metrics	Any reference which focuses on metrics (create, evaluate and use of...) applied on the Requirements Engineering field
Requirements Metrics: NL Quality Attributes	Any reference which mentions the quality attributes of a RE written using Natural language and how to measure them
Requirements Metrics: Use Cases Quality Attributes	Any reference which mentions the quality attributes of a RE written using Use Cases and how to measure them
Requirement Metrics - Others (RE Evolution, Inspection, User Documentation, etc)	Any reference which mentions subjects complementary to the RE general metrics (RE Evolution, Inspection Techniques, etc.)
Requirement Metrics - Specific Quality Attributes	Any reference which focuses on the measurement of a specific quality attribute
NOT EVALUATED YET	Any reference, which has NOT been evaluated yet but seems to be related to the main topic. The reason of not been evaluated is because there is not an available copy yet.

Table 1 Literature Survey Categories

A rating value was assigned to the references according to the following scale: *Very Relevant*, *Relevant, Might be Relevant, Not Relevant and Not Evaluated Yet.* For the categories *Requirements Metrics: NL Quality Attributes* and *Requirements Metrics: Use Cases Quality Attributes*, the references ranked as *Very Relevant* where does that presented a complete quality model for the SRS, this is, they presented several quality attributes and provided specific metrics to analyse the SRS and evaluate the degree of presence of the attributes defined by the authors.

The distribution of references found according their category and importance regarding each category is represented in Table 2.

Category	Very Relevant	Relevant	Might be Relevant	Not Relevant	Not Evaluated yet
Metrics					
Software Metrics		[FN00]		[ISMS]	
Requirements Metrics		[Uni-Magd]		[Bac99], [RR99], [Rul01]	
Requirements Metrics: NL Quality Attributes	[CL95], [DOJ93], [WRH97]	[Wie99], [RHHHW98], [FFGL01]	[KB96]		
Requirements Metrics: Use Cases Quality Attributes	[AM00], [FGLM02]	[FGL03]		[Coc00], [KG99]	
Requirement Metrics - Others (RE Evol, Inspection, User Documentation, etc)	[AF02]	[AM02], [BS02]	[AF00], [HCi02], [KAH00], [LV00], [Wie99]		
Requirement Metrics - Specific Quality Attributes	[Dra99], [KA01], [Mic01]	[Lui01]		[CC00]	
NOT EVALUATED YET					[ASJ01], [Dan01], [DRC01], [EO92], [Far90], [GW89], [Hur97], [IE830], [IE982.1], [IE982.2], [ISO], [Ken96], [Kov99], [KP00], [Kro98], [MG00], [RA98], [SS98], [Whi90], [Wie99]

Table 2 Reference by category and relevance

All information regarding the reference, such as author, date of publication, type of source (magazine, book, journal, etc.) was stored in a MS Access database. This database is available for further reference. Additionally, there are available printed and digital copies of the most relevant references.

4 Summary of Most Relevant References

This section presents the summary of the most relevant articles found under the categories: *Requirements Metrics: NL Quality Attributes and Requirements Metrics: Use Cases Quality Attributes.* The reason to present only the articles under these categories is that they provide a global view of the quality model of the SRS; that is they focus on an overall set of quality attributes and metrics for them and not just one or two aspects. Moreover, the goal of the survey is to obtain a picture of those metrics that are available in the software engineering community. Metrics like general documentation metrics are in this preliminary survey not considered in detail. However, this should be part of future work.

Each reference's summary is structured in the following way. An *Overview* section*,* which resumes the reference and presents, in a general way, the content of it; a *Metrics* section, which describes all the quality attributes, their definition and metrics to measure the fulfillment of the attributes, identified in the paper; a *Tools* section, which list all the tools mentioned by the authors that support the measuring process; and a *Related Authors References,* which provide a general comment regarding the references mentioned by the authors of the paper and also what other authors have said about the paper.

4.1 Identifying and Measuring Quality in a Software Requirements Specification

Written by Alan Davis, Scott Overmyer, Kathleen Jordan, Joseph Caruso, Fatma Dandashi, Anhtuan Dinh, Gary Kincaid, Glen Ledeboer, Patricia Reynolds, Pradip Sitaram, Anh Ta, and Mary Theofanos
Proceedings of the First International Software Metrics Symposium, IEEE Computer Society Press, Los Alamitos, California, 1993.
Reference code: [DOJ93]

4.1.1 Overview

Alan et al. defines in this article a set of attributes that contributes to evaluate the quality of a SRS. According to the authors, a ***quality*** SRS is "one that contributes to successful, cost-effective creation of software that solves real user needs".

The authors claim the real experiences demonstrate that many errors are introduced in the SRS. Therefore, if the quality of a SRS is measured, it will provide a better way to detect and prevent errors prior the implementation of the prod-

uct; that is, defects can be corrected before they are propagated to later phases. According to the authors, the errors fall into two categories:

- **Knowledge errors:** These are errors caused by not knowing what the true requirements are. These errors can be minimized using prototyping, however not all the errors can be detected through prototyping but when the system is deployed.

- **Specification errors:** These are errors caused by not knowing how to adequately specify requirements. These errors should not occur in an SRS.

The authors present a list of 24 attributes, gathered from a survey performed by the authors of the article. The attributes define the qualities a SRS should have. Even though not all attributes are measurable, they give a guideline to evaluate and measure the quality of the SRS.

The authors suggest different techniques for measuring those attributes assuming that there is a set *R* of all requirements in the SRS, composed of functional requirements and non-functional requirements. The different techniques are described with the metrics in the following section.

4.1.2 Metrics

Table 3 contains the name and definition of all quality attributes mentioned in the article and also the suggested techniques or activities that help to optimize the presence of the attribute; that is, techniques that support the fulfillment of the quality attributes. The authors also assigned a recommended weight relative to the other attributes in order to prioritize the quality attributes. Those attributes that are very relevant and should be present in a SRS are weighted by 1. A perfect quality SRS is impossible, as various quality attributes are contradictory. For example, unambiguous requirements can be gained by introducing redundancy but this would contradict the quality attribute not redundant. However, to have an overall quality of the SRS, the Equation 1 provides just one number that states the quality of an SRS.

$$Q = \frac{\sum_{i=1}^{18} W_i Q_i}{\sum_{i=1}^{18} W_i}$$

Equation 1 SRS Quality Equation

Only 18 of the 24 quality attributes presented in the article have a metric related to them Thus, the index in the equation is only from 1 to 18. (see Table 4).

QUALITY ATTRIBUTE / METRIC	DEFINITION	TECHNIQUES/ACTIVITIES	WEIGHT
Unambiguous	An SRS is unambiguous if and only if every requirement stated therein has only one possible interpretation	Replacing natural language specifications with formal notations Augment natural language with more formal models	1
Complete	An SRS is complete if: 1. Everything that the software is supposed to do is included in the SRS 2. Responses of the software to all realizable classes of input data in all realizable classes of situations is included 3. All pages numbered; all figures and tables numbered, named, and referenced; all terms defined; all units of measure provided; and all referenced material present 4. No section marked "To be determined"	Reviews of the SRS by customer or user are essential Prototypes help raise awareness of new requirements and identify poor requirements	7
Correct	An SRS is correct if and only if every requirement represents something required of the system to be built.	Involve people, who have the problem or mission, to serve as oracles for validate the correctness of the requirement	1
Understandable	An SRS is understandable if all classes of SRS readers can easily comprehend the meaning of all requirements with a minimum of explanation	Reviews of the SRS by all classes of SRS readers Augmentation with a prototype can improve effective understandability	1
Verifiable	An SRS is verifiable if there exist finite, cost effective techniques that can be used to verify that every requirement stated therein is satisfied by the system as built.	Same techniques applied for ambiguity Knowledge of undecideability and review for its presence Review of SRS by experienced testers who can determine high cost or schedule testing implications	7
Internally Consistent	An SRS is internally consistent if and only if no subset of individual requirements stated therein conflict.	Using tools like RLP, REVS, CASE tools	1
Externally Consistent	An SRS is externally consistent if and only if no requirement stated therein conflicts with any already baselined project documentation.	Create and maintain full cross-references between all requirements and relevant statements made in other documents: Requirement Specifications (RS), statements of work (SOW), software development plan (SDP), etc.) Review SRS simultaneously with other conflicting documents including SOW, development contract and SDP.	1
Achievable	An SRS is achievable if and only if there could exist at least one system design and implementation that correctly implements all the requirements stated in the SRS.	Construct a working prototype of parts of the system where achievability may be in doubt.	1

QUALITY ATTRIBUTE / METRIC	DEFINITION	TECHNIQUES/ACTIVITIES	WEIGHT
Concise	An SRS is concise if it is as short as possible without adversely affecting any other quality of the SRS.	NONE SUGGESTED	2
Design- Independent	An SRS is design independent if and only if there exist more than one system design and implementation that correctly implements all requirements stated in the SRS.	Have designers review the SRS.	5
Traceable	An SRS is traceable if and only if its is written in a manner that facilitates the referencing of each individual requirement	Number every paragraph hierarchically Number every paragraph hierarchically and include only one requirement in any paragraph. Number every requirement with a unique number in parentheses immediately after the requirement Use a convention for indication a requirement	Not mentioned (supposed 1)
Modifiable	An SRS is modifiable if its structure and style are such that any changes can be made easily, completely and consistently.	Make the SRS also traceable, machine readable form, traced, organized, and cross-referenced. Include a table of contents and index	Application Dependent
Electronically Stored	An SRS is electronically stored if and only if the entire SRS is in a word processor; it has been generated from a requirements database or has been otherwise synthesized from some other form.	NONE SUGGESTED	Application Dependent
Executable/Interpretable/Prototypable	An SRS is executable, interpretable, or prototypable if and only if there exits a software tool capable of inputting the SRS and providing a dynamic behavioral model.	Write the SRS in a language that (1) is directly understood by a computer, (2) is translatable into a language directly understood by a computer, or (3) can be interpreted by a software tool and thus simulated Use any commercial tool that provides such execution of requirements	Application Dependent
Annotated by Relative Importance	An SRS is annotated by relative importance if a reader can easily determine which requirements are of most importance to customers	Suffix every requirement with (M), (D) and (O) to denote that this requirement is mandatory, desirable, or optional	Application Dependent
Annotated by Relative Stability	An SRS is annotated by relative stability if a reader can easily determine which requirements are of most likely to change.	Suffix every requirement with (H), (M), and (L) to denote whether the probability of change is high, medium, or low	Application Dependent
Annotated by Version	An SRS is annotated by version if a reader can easily determine which requirements will be satisfied in which versions of the product	Add a column in the margin for each version of software to be produced.	Application Dependent
Not Redundant	An SRS is redundant if the same requirement is stated more than once	Redundancy is not necessary bad, if it wants to be controlled, then include an index and cross references among any redundant requirement	Usually 0

QUALITY ATTRIBUTE / METRIC	DEFINITION	TECHNIQUES/ACTIVITIES	WEIGHT
At Right Level of Abstraction/Detail	An SRS should use appropriated the level of abstraction. This means that the SRS system should be specific enough that any system built that satisfies the requirement in the SRS satisfies all user needs, and abstract enough that all systems that satisfy all user needs also satisfy all requirements.	NONE SUGGESTED	Scenario Dependent
Precise	An SRS is precise if and only if (a) numeric quantities are used whenever possible, and (b) the appropriate levels of precision are used for all numeric quantities.	NONE SUGGESTED	Not mentioned
Reusable	An SRS is reusable if and only if its sentences, paragraphs and sections can be easily adopted or adapted for use in a subsequent SRS.	Write SRS sections using "symbolic constants" Use formal models Create library of abstract requirements types.	Not mentioned
Traced	An SRS is traced if and only if the origin of each of its requirements is clear. This is, every requirement that has a basis is cross-referenced to that basis.	Include explicit cross-references in parentheses following each requirement in the SRS. Record all the requirements in a database	Not mentioned
Organized	An SRS is organized if and only if its contents are arranged so that readers can easily locate information and logical relationships among adjacent sections is apparent.	Follow any of SRS standards Group the functional requirements by (1) class of user, (2) common stimulus, (3) common response, (4) feature, or (5) object	Not mentioned
Cross-Referenced	An SRS is cross referenced if and only if cross-references are used in the SRS to relate sections containing requirements to other sections containing (1) identical requirements, (2) more abstract descriptions of same requirements, (3) requirements that depend on them or on which they depend.	Any technique applied for Traced Use explicit intext cross-references Store all requirements in a database and use specific field to store the three types of cross-references: (1) Identical requirements, (2) more abstract descriptions of same requirements, (3) requirements that depend on them or on which they depend.	Not mentioned

Table 3 Quality attributes

Based on these quality attributes the authors identified metrics that measure the degree of fulfillment of the attribute. Table 4 contains the name of the quality attributes with the respective metrics (formula) suggested by the authors. In the column "purpose" the authors provide the rationale why the metric is collected. In order to use the metric in a valuable way, the authors also present a reference value. This value gives hints about how to interpret the measured value; that is, what does a completeness value of 2.3 really mean.

QUALITY ATTRIBUTES / METRIC	FORMULA	PURPOSE	REFERENCE VALUE
Unambiguous	$Q_1 = \frac{n_{ui}}{n_r}$ Where, N_{ui} is the number of requirements for which all reviewers presented identical interpretations N_r is the total number of requirements	To obtain the percentage of requirements that have been interpreted in a unique manner by all its reviewers	The more closed to 0 the more ambiguous are the requirements (every requirement has multiple interpretation) The more close to 1 the more unambiguous are the requirements (every requirement has one interpretation)
Complete[1]	$Q_2 = \frac{n_u}{n_i x n_s}$ Where, - N_{ui} is the unique function - N_i is the *stimulus* input of the function - N_s is the *state* input of the function	To measure the total number of functions currently specified. Completeness implies that the function *f(state, stimulus)* → *(state, response)* is defined for all elements in the cross product *state x stimulus*	The more close to 1 the more complete it is.
Correct	$Q_3 = \frac{n_c}{n_c \times n_{NV}} = \frac{n_c}{n_r}$ Where, - n_C is the number of correct requirements - n_{NV} still not validated requirements - n_c total requirements	To measure percentage of requirements in the SRS that have been validated.	0: Totally incorrect 1: Totally correct
Understandable	$Q_4 = \frac{n_{ur}}{n_r}$ Where, - n_{ur} is the number of understood requirements - n_r is the total of requirements	To measure the number of requirements for which all reviewers thought they understood it.	0: No requirement understood 1: All Requirement understood

[1] The authors presents two more ways of measure completeness however, as this attribute is very difficult to measure, they are not presented.

QUALITY ATTRIBUTES / METRIC	FORMULA	PURPOSE	REFERENCE VALUE
Verifiable	$Q_5 = \frac{n_r}{n_r + \sum_i c(r_i) + \sum_i t(r_i)}$ Where, - n_r is the total of requirements - c is the cost necessary to verify presence of requirement r_i - t is the time necessary to verify presence of requirement r_i	As measurement of verifiability is very difficult, the authors present a cost-effectiveness or finiteness of the verification approach However, no suggestion is made how to define these values.	0: Very poor verifiability 1: Very good verifiability
Internally Consistent	$Q_6 = \frac{n_u - n_n}{n_u}$ Where, - n_u is the number of unique functions specified - n_n is the number of unique functions that are non-deterministic (how many of them map the same point in the domain into different points in the range)	To measure the percentage of unique functions that are deterministic assuming that SRS can be defined as a function that maps inputs and states into outputs and states (as a Finite State Machine)	0: 100% internally inconsistent 1: 100% internally consistent
Externally Consistent	$Q_7 = \frac{n_{EC}}{n_{EC} \times n_{EI}} = \frac{n_{EC}}{n_r}$ Where, - n_{EC} is the number of requirements that are consistent with all other documents - n_{EI} is the number of requirements that are not consistent - n_r is the total of requirements (N_{EC}+N_{EI})	To measure the percentage of requirements that are consistent with all other documents	Not defined
Achievable	Not defined	The measure of the existence of a single system.	1: A set of requirements are either achievable 0: Given acceptable development resources the SRS is not achievable

QUALITY ATTRIBUTES / METRIC	FORMULA	PURPOSE	REFERENCE VALUE
Concise	$Q_9 = \frac{1}{size + 1}$ Where, *size* is the number of pages	Determine if two systems SRSs describe identical systems is undecideable, therefore one way is to count pages using the hyperbole function	0: Worst case of conciseness which means a SRS of infinite size 1: Ultimate in conciseness is the null SRS
Design- Independent	$Q_{10} = \frac{D(R_E \cup R_I)}{D(R_E)}$ Where, - R_E is the total of requirements that describe pure external behavior - R_I is the total of requirements that directly address architecture or algorithms of the solution	To measure the percentage of possible solution systems that are eliminated by adding the overly constraining requirements	Values range from 0: Highly design dependent 1: Design independent
Traceable	Not defined	To measure which requirements are being supported by the components or verified by testing. If a single requirement is not traceable, then the whole SRS is untraceable.	1: If it exhibits any of the qualities described on Table 3 for the different techniques to achieve traceability 0: If it does not exhibit any.
Modifiable	Not defined	To measure if it contains a table of context and index	1: If the table of contents and index are present 0: Otherwise
Electronically Stored	Not defined	To measure the percentage of the volume of the SRS that has been electronically stored	Not mentioned.
Executable/Interpretable/Prototypable	Not defined	To measure how many requirements have the quality attribute	0: Entirely not executable 1: Entirely executable
Annotated by Relative Importance	Not defined	To calculate the percentage of requirements that are annotated	0: SRS is annotated by relative importance 1: SRS is not annotated by relative importance
Annotated by Relative Stability	Not defined	To calculate the percentage of requirements that are annotated	0: SRS is annotated by relative stability 1: SRS is not annotated by relative stability

QUALITY ATTRIBUTES / METRIC	FORMULA	PURPOSE	REFERENCE VALUE
Annotated by Version	Not defined	To calculate the percentage of requirements annotated by version	0: SRS is not annotated by Version 1: SRS is annotated by Version. When a SRS is written for just one version, this is the default value.
Not Redundant	$Q_{18} = \frac{n_f}{n_u}$ Where, - n_f is the total of functions currently specified - n_u is the current unique functions specified	To measure the percentage of unique functions that are not repeated	Values range from: 0: Completely redundant 1: No redundancy
At Right Level of Abstraction/Detail	Not defined	To measure the appropriateness of the level of abstraction; however this cannot be measured	Not defined
Precise	Not defined	Not defined	Not defined
Reusable	Not defined	To measure if the SRS has been reuse	1: If the contents of a SRS have been fully reused by later SRSs 0: If none of the contents of a SRS have been reused.
Traced	Not defined	Measuring the level of trace-ness is impossible, therefore none metric is presented	Not defined
Organized	Not defined	Not defined	Not defined
Cross-Referenced	Not defined	Not defined	Not defined

Table 4 Quality Attributes and Metrics

With this set of metrics the authors can evaluate the Equation 1 given above. This allows the requirements engineer to obtain an overall perception of the quality of the SRS.

4.1.3 Requirements Language

The article does not focus on a specific notation of the requirements.

4.1.4 Tools

The authors provide several references to different tools and specific techniques that can either support the measurement of the different quality attributes or maximize the presence of that attribute. Those tools and techniques are summarized inTable 5.

QUALITY ATRIBUTE	TOOLS / TECHNIQUES
Unambiguous	Replacing natural language with formal notations: Deterministic finite state machines (FSM), Petri Nets (PN), decision trees (DT), prepositional calculus, predicate calculus, etc.
Internal Consistent	RLP[2], REVS[3], CASE Tools
Executable/Interpretable/Prototypable	PAISLey[4], RAPID[5], STATEMATE[6], DFD-based Case tools.

Table 5 Tools and techniques referenced in the paper

4.1.5 Related Authors References

Most of the very relevant articles make references to this article and extend, improve and complete the quality attributes presented by David et al.

The references mentioned in this article are quite old, they range from 1974 to 1993, and therefore they have not been analyzed.

4.2 Automated Quality Analysis of Natural Language Requirement Specification

Written by Wilson, William M.; Rosenberg, Linda H.; Hyatt, Lawrence E.
In Proceeding International Conference On Software Engineering, May 1997
Reference Code: [WRH97]

4.2.1 Overview

The article refers to an automated tool, *Automated Requirements Measurement* (ARM), developed by the Goddard Space Flight Center's (GSFC) Software

[2] Davis, A., et al., "RLP: An Automated Tool for the Processing of Requirements." IEEE COMPSAC'79, 1979.

[3] Alford, M., "SREM at the Age of Eight: The Distributed Computing Design System." IEEE Computer, 18, 4 (April 1989), pp.36-46

[4] Zave, P., and R. Yeh, "Executable Requirements for Embedded Systems", Fifth IEEE Int'l Conf. On Software Eng., 1981, pp. 295-304.

[5] Wasserman, A., et al., "Developing Interactive Information System with the User Software Engineering Methodology." IEEE Trans. Software Eng., 12, 2 (February 1986), pp. 325-345

[6] Harel, D., et al., "STATEMATE: A working Environment for the Development of Complex Reactive System". IEEE Tenth Int'l Conf. On Software Eng., 1988.

Assurance Technology Center (SATC). The tool was created because even though several tools exist that support the creation and management of requirement specifications, very few of them really assist the evaluation of the quality of the requirement document or individual specification statements.

The ARM tool evaluates the requirements documents with respect to quality criteria defined by the SATC. These quality attributes are criteria that can be objectively and quantitatively measured and were grouped according to different categories. These categories represent primitive indicators of the specification's quality and can be detected using the ARM tool on a given specification written in ASCII-text.

The result provided by the tool can be used to improve a given requirements document. Feedback of this work is used to improve the ARM tool.

4.2.2 Metrics

Eleven quality attributes were select from the SATC based on the work done by Stokes[7] and the IEEE Std 830-1993[8], however they are not independent. For example, a specification cannot be correct if it is incomplete or inconsistent. According to the authors, most of these attributes are subjective. Therefore, they require review and analysis by technical and operational experts. However, they can be linked to primitive indicators that provide some evidence if the quality attribute is present or absent in the specification. Table 6 presents these eleven quality attributes and their definition. Note that the authors took these definitions from other sources.

QUALITY ATTRIBUTE	DEFINITION
Complete	A complete requirements specification must precisely define all the real world situations that will be encountered and the capability's responses to them
Consistent	A consistent specification is one where there is no conflict between individual requirement statements that define the behavior of essential capabilities; and specified behavioral properties and constraints do not have an adverse impact on that behavior
Correct	A requirements specification is correct if it accurately and precisely identifies the individual conditions and limitations of all situations that the desired capability will encounter and it must also define the capability's proper response to those situations.
Modifiable	In order to create modifiable requirements specifications, related concerns must be grouped together and unrelated concerns must be separated.
Ranked	Ranking specification statements according to stability and/or importance is established in the requirements document's organization and structure.
Traceable	Each statement of requirements must be uniquely identified to achieve traceability

[7] Stokes, David Alan, Requirements Analysis, *Computer Weekly Software Engineer's Reference Book,* 1991, pp. 16/3-16-21

[8] IEEE Std 830-1993 *Recommended Practice for Software Requirements Specifications,* December 2, 1993

QUALITY ATTRIBUTE	DEFINITION
Unambiguous	A statement that specifies a requirement is unambiguous if it can only be interpreted one way
Understandable	A requirements specification is understandable if the meaning of each of its statements is easily grasped by all of its readers
Verifiable	In order to be verifiable requirement specifications at one level of abstraction must be consistent with those at another level of abstraction
Validatable	In order to validate a requirements specification each of the individuals and organizations having a vested interest in the system solution must be substantiate that the requirements are true as stated. To validate a requirements specification all the project participants, managers, engineers and customer representatives, must be able to understand, analyze and accept or approve it
Testable	In order for a specification to be testable it must be stated in such as manner that pass/fail or quantitative assessment criteria can be derived from the specification itself and/or referenced information

Table 6 Quality Attributes that are considered in the approach

The categories, which represent indicators for the presence of the above mentioned quality attributes are defined in Table 9. The definition gives hints on how to measure the fulfillment of the attribute. In the column "quality indicator" the authors describe how the measures obtained by the tool can be interpreted and how the categories and the related quality indicators are related the quality attribute in Table 6.

CATEGORY	DEFINITON	QUALITY INDICATOR
Imperatives	Imperatives are those words and phrases that command that something must be provided (Shall, Must or Must not, Is required to, Are applicable, Responsible for, Will, Should).	- The more explicit the requirement was, the majority of the imperative counts were found.
Continuances	Continuances are phrases such as those listed below that follow an imperative and introduce the specification of requirements at a lower level (Below, as follows, following, listed, in particular, support).	- It indicates organization and structure of the SRS, which contribute to the tractability and maintenance of the requirement. - Extensive use of continuances also indicates the presence of very complex and detailed requirements specification statements
Directives	Directives is the category of words and phrases that point to illustrative information within the requirements document (figure, table, for example, note)	- It indicates the precision of the requirement
Options	Options is a category of words that give the developer latitude in satisfying the specification statements that contain them (can, may, optionally).	- It reduces the acquirer's control over the final product and establishes a basis for possible cost and schedule risk.
Weak Phrases	Weak Phrases is the category of clauses that are apt to cause uncertainty and leave room for multiple interpretations (adequate, as a minimum, as applicable, easy, as appropriate, be able to, be capable, but not limited to, capability of, capability to, effective, if practical, normal, provide for, timely).	- The total number of weak phrases is an indication of the extent that the specification is ambiguous and incomplete.

CATEGORY	DEFINITON	QUALITY INDICATOR
Size	Size is the category used by the ARM tool to report three indicators of the size of the requirements specification document. They are the total number of: · lines of text · imperatives · subjects of specification statements · paragraphs	- The number of subjects used in the specification document indicates the scope of the document - The ratio of imperatives to subjects provides an indication of the level of detail being specified. - The ratio of lines of text to imperatives provides an indication of how concise the document is in specifying the requirements.
Specification Depth	Specification Depth is a category used by the ARM tool to report the number of imperatives found at each of the documents levels of text structure.	- It reflects the structure of the requirements statements as opposed to that of the document's text.
Readability	Readability Statistics are a category of indicators that measure how easily an adult can read and understand the requirements document. Four readability statistics produced by Microsoft Word are calculated and compared (Flesch Reading Ease index, Flesch-Kincaid Grade Level index, Coleman-Liau Grade Level index)	- It indicates the understandability of the requirement.
Text Structure	Text Structure is a category is used by the ARM tool to report the number of statement identifiers found at each hierarchical level of the requirements document.	- It provides an indication of the document's organization, consistency, and level of detail.

Table 7 Categories that are used as indicators for a quality attribute

As mentioned before, the idea of having these categories is to find indicators that can help to identify whether or not the quality attributes are realized in the specifications. These indicators provide information regarding one or more quality attributes. In addition, the authors present the relation between the quality attributes and the risk associated with the project's product and its resources. The definition of the different kinds of risk identified by the author is presented in Table 8.

RISK	DEFINITION
Product Risks	Inadequacies in the requirements specification document introduce the possibility that the product's design will contain deficiencies that will be propagated as implementation faults. The possibility of these faults increases the probability that one or more of the products characteristics will be unsatisfactory.
Acceptance Risk	An Acceptance Risk is the probability that the product will not satisfy its acceptance criteria. A deficiency in any of the requirements specification's quality attributes may introduce a risk that the product will not be acceptable.
Availability Risk	An Availability Risk is the probability that the product will not be present or functional at a time when it is needed. A deficiency in any of the requirements specification's quality attributes may introduce a risk that the product will not be delivered on time or its functionality cannot be maintained in the operational environment.
Performance Risk	A Performance Risk is the probability that the product will not be capable of performing properly in the operational environment. A performance risk will arise if the requirements specification is difficult to understand or inadequately specifies the functional capabilities that are to be provided.

RISK	DEFINITION
Reliability Risk	A Reliability Risk is the probability that the product will fail in the operational environment. A deficiency in any of the requirements specification's quality attributes can introduce a risk thatthe product will not achieve the level of reliability need to successfully perform its mission.
Reproducibility Risk	A Reproducibility Risk is the probability that the product cannot be reproduced for replacement of the original product or for distribution to additional sites. If the requirements specification is poorly written and functions are inadequately prescribed there will be a risk that the product or components of the product cannot be replicated when replacement is necessary.
Supportability Risk	A Supportability Risk is the probability that the product cannot be adequately maintained or logistically provided in the operational environment. A deficiency in any of the requirements specification's quality attributes can introduce a risk that the resources needed to operate and maintain the product will not be provided and/or the product itself does facilitate maintenance.
Utility Risk	A Utility Risk is the probability that the product will be less useful than demanded by the operational environment. A risk that the delivered capability will not provide the user with the full range of necessary functionality will arise if the requirements specification is difficult to understand or the functions that are to be provided are inadequately specified.
Resource Risks	The estimate of resources needed to provide a capability, including the time allocated for its acquisition, is based of the specification of the requirements that the capability is to satisfy. If the requirements are improperly specified the resource estimates will be incorrect. Product risks introduced by deficient requirements can result in delays and rework that introduce cost and schedule risks.
Cost Risk	A Cost Risk is the probability that the production or acquisition of the product will exceed allocated resources available for that purpose. Any deficiency in the requirements specification, such as incompleteness or ambiguity that causes the development effort to be under estimated or requires rework will introduce a risk that costs will exceed available funding.
Schedule Risk	A Schedule Risk is the probability that the product will not be delivered as scheduled. Any deficiency in the requirements specification's quality attributes can introduce a risk that the established schedule is inadequate or that product deficiencies will require unanticipated additional time to correct.

Table 8 Risks that should be considered with respect to the quality

The described risks are linked to the quality attributes a good SRS should have. However, the relation between the risk and the attributes is quite abstract and needs further investigation.

Table 9 shows the relation between different indicators, quality attributes and different kind of risks. This relationship provides hints how the indicators and related quality attributes can be used to reduce the level of a certain risk in the development process.

CATEGORIES OF QUALITY INDICATORS		QUALITY ATTRIBUTES										
	CATEGORY	Complete	Consistent	Correct	Modifiable	Ranked	Testable	Traceable	Unambiguous	Understand-able	Validatable	Verifiable
Individual Statements	Imperatives	X			X			X	X	X	X	X
	Continuances	X			X	X	X	X	X	X	X	X
	Directives	X		X			X		X	X	X	X
	Options	X					X		X	X	X	
	Weak Phrases	X		X			X		X	X	X	X
Entire Re-quirement	Size	X					X		X	X	X	X
	Specification Depth	X	X		X			X		X		X
	Readability				X		X	X	X	X	X	X
	Text Structure	X	X		X	X		X		X		X
Product Risk	Acceptance	X	X	X	X	X	X	X	X	X	X	X
	Availability	X		X	X	X	X					
	Performance	X	X	X	X	X			X	X		
	Reliability	X	X	X	X	X	X	X	X	X	X	X
	Reproducibility	X	X	X	X	X	X	X	X	X	X	X
	Supportability	X	X	X	X	X	X	X	X	X	X	X
	Utility	X		X	X	X			X	X		X
Resource Risk	Cost	X		X	X	X	X		X	X		
	Schedule	X		X	X	X	X	X	X	X	X	X

Table 9 Relation between quality attributes, risks and indicators

Based on these results the authors provide a technique to measure the quality of the SRS with the ARM tool. By linking the quality to certain risks in the software development process, the authors provide some hints on how the measures gained with the tool can be used for project management and risk management purposes

4.2.3 Requirements Language

The authors created the tool and used requirements written in natural language (English). A pre-requisite to use the tool is that the requirements document is in plain ASCII text.

4.2.4 Tools

The ARM tool is used to obtain the quantitative values for the categories. MS Word is also used to obtain readability statistics.

4.2.5 Related Authors References

From the same authors, the following references, which are also available, provide also the information regarding to the topic they present in this article. Because the information does not differ too much from the one presented in this article, they are not summarized here:

Rosenberg, Linda H. Hammer, Theodore F. Huffman, Lenore L. *Requirements, Testing and Metrics*

Rosenberg, Linda H. Hammer, Theodore F. Shaw, Jack. *Software Metrics and Reliability*

Wilson, William M. Rosenberg, Linda H. Hyatt, Lawrence E. *Automated Analysis of Requirement Specifications*

4.3 Metrics for Requirement Engineering

Written by Rita J. Costello and Dar-Biau Liu
J. Systems Software, 1995 Elsevier Science Inc.
Reference Code: [CL95]

4.3.1 Overview

The purpose of metrics, from the point of view of the authors, is to provide objective information, which is needed by technical and managerial personnel in order to control and improve the development effort. The literature found by Costello and Liu provided little information regarding metrics to be applied under specific phases. It is already known that most of failed software development projects are due to problems in requirements. Therefore, it is the aim of the authors to define metrics that can be used to identify, isolate and resolve problems before affecting lower levels of the life-cycle phases.

The article of [Dav et al., 93] contains the broadest attempt in suggesting metrics for requirement engineering. However, according to the authors, many of the 24 quality attributes (See Table 3) defined by Davis et al., are subjective and very difficult to quantify.

In this article, the authors identify and define a set of requirement metrics that are based on objectively obtainable, quantitative data. These metrics were selected by answering the following questions:

- What are the key characteristics of requirements that we need to be able to monitor?
- Which of these can we objectively quantify and which require subjective assessment?

The authors have defined 8 metrics to be applied on requirement documents but only for three of them more detailed information is presented. The data must be collected from all requirement sources and regardless which kind of language is being used to write the requirements; the following conditions should be followed:

1. **Countable requirements**: Each requirement should be individually countable.
2. **Internally/Externally consistency**: Each requirement should be internally consistent and consistent with all other requirements in the specification
3. **Annotation available**: Each requirement should carry annotations in use by the program: likelihood of change, implementation priority, applicable build, requirements baseline, etc.
4. **Consistent convention**: Each requirement should be consistent with the convention used to specify the requirements.
5. **Requirement Traceability Matrix (RTM)**: Each specification should have a requirement traceability matrix that indicates the parent and subordinate requirements for each requirement in the specification.
6. **Electronically stored**: Requirements specifications should be electronically stored in a form that permits individual requirements to be identified and traced to requirements in higher and lower level specifications. This allows increasing the cost effectiveness and accuracy of data collection.

Note that the author do not present any metric to be applied in order to assure if a given SRS accomplish the above attributes. The above mentioned attributes are more pre-requisites that should be fulfilled by an requirements document.

4.3.2 Metrics

The author presents a list of metrics that can be applied on the requirements document. Table 10 presents the purpose and overview of those metrics.

METRIC	OVERVIEW	PURPOSE
Volatility	Indicates changes (additions, deletions, modifications) and reasons for changes to requirements.	Provides insight into system maturity and stability. Aids in predicting future requirements, design, and code volatility. Essential for interpreting other metrics.
Traceability	Indicates degree to which development organization maintains accountability for meeting requirements at each stage of life cycle via a requirements traceability matrix (RTM).	Provides quantitative means for determining whether all required relationships/ dependencies are addressed. Assists in exposing incompletely specified, overly specified and complex areas of system. Essential for interpreting other metrics
Completeness	Indicates completeness of all sections of requirements specifications, whether all allocated higher-level requirements are addressed, and the degree of decomposition of allocated higher-level requirements.	Assists in determining readiness to proceed to design.
Defect Density	Indicates number of requirements defects that are initially detected during an inspection or walkthrough. Classified by type, criticality, and source.	Provides early insight into quality, assists in cost/schedule estimation, and indicates effectiveness and extent of inspection / walk-through process. Useful in predicting product /process volatility. Essential for interpreting other metrics.
Fault density	Indicates number of requirements faults that are initially detected during test execution or posttest analysis. Classified by type, criticality, and source.	Assists in determining effectiveness of software process and the extent and effectiveness of testing. Useful in predicting product / process volatility and quality. Essential for interpreting other metrics.
Interface consistency	Indicates consistency and completeness of interface information at each level of specification.	
Problem report/action item/issue	Indicate number of requirements problems detected/issues raised via any process; as such, include requirements defects and faults. Characterize problems by source, product, type of problem, finding activity, severity, criticality, age, status, and primary reason for closure.	Number and status of requirements problem reports indicate quality of requirements engineering and inspection processes. Essential for interpreting other metrics.
Integrated progress	Indicates overall requirements progress. Encompasses measures of volatility, traceability, completeness, defect density, fault density, test coverage, and problem reports/ action items as appropriate for phase under consideration.	

Table 10 Metrics for the requirements document Engineering

Based on the definition of the metrics the authors defined formulas how to measure each of the metrics. Moreover, the authors define the purpose why the metric should be measured and give a reference value to support the interpretation of an obtained metric value.

QUALITY ATTRIBUTES / METRIC	FORMULA	PURPOSE	REFERENCE VALUE
Volatility	None presented	They consistent of counts of requirements changed (added, deleted and modified) in each specification over a given time interval, classified by reason for change.	None presented, however if there are high number of changes it may be desirable to plot the reasons for change at a lower level of detail and gather data to plan for possible future volatility and also it could indicate that insufficient effort has been applied in the requirements definition process.
Traceability[9]	Next-level traceability (COV)	This includes: COVup: It consists of the number of requirements that trace consistently to the next level up COVdown: It consists of the number of requirements that trace consistently to the next level down.	None presented.
	Full depth and height coverage (DHCOV)	This evaluate traceability to the highest (HCOV) and lowest (DCOV) levels of specification.	
	Linkage statistics	They measure the complexity of traceability data by providing counts of the number of higher / lower level requirements to which each requirement in a specification is traced.	
	Inconsistent Traceability Matrix (ITM)	This includes the number of requirements in a specification that have at least one inconsistent traceability link upward (ITMup) or downward (TIMdown)	
	Undefined Traceability Metrics (UTM)	It includes the number of requirements in a specification that have no traceability links upward (UTMup) or downward (UTMdown).	
Completeness[10]	Requirement decomposition (RD)	They provide insight into whether allocated requirements have been sufficiently decomposed for next phase that will use the information from that specification	None presented, however, the following degrees suggest how to evaluate the results: **Low degree of decomposition**: It may mean that a requirement is simple or specifies a very limited function or behavior. It also may mean that the analysis of the requirement was superficial and the requirement needs to be decomposed further. **High degree of decomposition**: It may represent a complex function.

QUALITY ATTRIBUTES / METRIC	FORMULA	PURPOSE	REFERENCE VALUE
	Requirements Specification Development (RSD)	This provides information on work accomplished versus work remaining for a given specification. RSD tacks the number of higher-level requirements allocated to a lower level specification that are reflected in one or more requirements in the lower level specification.	None presented.
	Requirement Specification "to be completed " (TBC)	This provides information on the number of items in the specification for which issue resolution is required and the number of higher-level requirements affected by unresolved issues. Two sub metrics are defined: **STBx**: number of incomplete requirements + number of blank and omitted sections + the number of references to nonexistent and incomplete materials. **TTSTBx**: It is the number of higher-level requirements allocated to the specification that trace to one or more incomplete requirements in the specification.	None presented

Table 11 Volatility, Traceability and Completeness metrics

This article is the first of a series of documents about metrics on the different phases of the development life-cycle the authors intended to write. However, no further article was found during the survey.

With the presented approach the authors define important quality attributes of a SRS but only a few metrics are defined. In future research more metrics should be developed to measure the remaining metrics.

4.3.3 Requirements Language

The article does not focus on a specific notation for the requirements.

9 The traceability metric is composed of five different types of metrics. They are listed on the column "FORMULA" but none quantitative formula is presented on the article.

10 Same as for traceability, the completeness metrics is composed of three metrics but none formula is presented for them.

4.3.4 Tools

None specified

4.3.5 Related Authors References

No references from the same authors could be found. The most relevant reference mentioned in the article are summarized in Section 4.1.

4.4 Application of Linguistic Techniques for Use Case Analysis

Written by A. Fanechi, S. Gnesi, G. Lami, A. Maccari
IEEE Joint International Conference on Requirements Engineering (RE'02)
Reference code: [FGLM02]

4.4.1 Overview

The metrics described in the paper are derived from linguistic techniques (lexical/morphological and syntactic analysis of natural language text) to detect ambiguities in the requirements. The existence of ambiguities in natural language text could cause interpretation problems, which the authors have grouped in three main categories. These categories include other important quality attributes of a SRS that are affected by ambiguity problems. Therefore, this article deals with more than just one quality attribute. The defined categories are:

- ***Expressiveness***: Characteristics dealing with the understanding of the meaning of Use Cases by humans. This category includes the following topics:
  - Ambiguity mitigation: Detection and correction of linguistic ambiguities in the sentences.
  - Understandability improvement: Evaluation of the understandability level of a requirements document and indication of the parts that need improvement.
- ***Consistency***: Characteristics dealing with the presence of semantic contradictions and structural incongruities in the natural language requirement document.
- ***Completeness***: Characteristics dealing with the lack of necessary parts within the RSD.

The author have addressed the problem of detecting linguistic inaccuracies in Use Cases starting from the definition of a set of metrics related to quality

characteristics which belong to the ***Expressiveness*** category. These metrics have been derived based on quality properties and quality indicators of three different tools: QuARS, ARM, SyTwo (See Section 4.4.3 for a more detailed description regarding these tools). Using these tools allowed the authors to derive some metrics, which are described in Section 4.4.2

To address the other two categories, ***Consistency*** and ***Completeness*** it is necessary to analyze the whole use case. The authors present two possible techniques:

- ***Using Functional Dependency Grammar:*** These are parsers that are able to enrich text with functional dependencies. These dependencies have information about sentence-level relations and functions between words and linguistic structures.
- ***Use Cases as structured in three semantic layers:*** Use Cases can be seen in three semantic layers:
  - ***Layer I - Specification and Use Cases:*** This layer is composed by the specification and a set of use cases where each use case defines a relation between main actor(s) and secondary actors
  - ***Layer II - Use Case Structure:*** This layer is composed by the use case itself and its internal structure where the different scenarios and its extensions defines the system's operation and a sequential control flow. The scenario extensions define the exceptions of that control flow
  - ***Layer III - Scenario or extension sentence:*** This layer belongs to the linguistic structure that each scenario or extension sentence has. This structure defines a relation between actors (Layer I) and system operations (Layer II).

According to the authors, it is on the third layer that the linguistic analysis technique has immediate application.

The definition of a relational structure combining the results of the linguistic analysis of such sentences and the structure implied by the other layers is still a research topic.

4.4.2 Metrics

The metrics described by the authors are the results of the use of three different tools. Each tool has a quality model that describes quality attributes for natural language requirements. All these tools are aids for *writing the* ***requirements right*** not *writing the* ***right requirements.***

Table 12 contains the 14 metrics derived by the authors. Each metric is mapped to a certain category that are sub-categories of the above mentioned ones.

Also, a formulas are provided in order to measure a certain metric and a rationale provides the reason why the measurement of a certain metric is of importance.

METRICS	CATEGORY	FORMULA	RATIONAL
Coleman-Liau Formula	Readability	5.89 * (letters / words) - 0.3 * (sentences / (words / 100)) - 15.8	It measures the difficulty in reading the document
Average number of words per sentence	Readability - Understandability	Nw / Ns where Nw= number of words; Ns = number of requirement sentences	Short sentences make the requirements document more readable/ understandable
Continuance Index	Traceability - Maintainability	Ncon/Nreq, where Ncon= number of continuances in sentences; Nreq = number of requirement sentences. Continuances are phrases as "the following:" that follow an imperative verb and precede the definition of lower level requirement specification.	The use of continuances indicates a well structured document, but too many continuances indicate multiple, complex requirements
Comment Frequency	Understandability	Nc / Ns where Nc= number of comment sentences; Ns = number of requirement sentences	The comments within the requirements document reduce the risk of misinterpretations
Directives Frequency	Understandability	Nd / Ns where Nd= number of directives; Ns = number of requirement sentences	Directives (i.e. words or phrases that indicate examples or other illustrative information) make the document more understandable.
Multiplicity	Understandability	Nmul / Nreq where Nmul = number of sentences having more than one main verb or more than one direct or indirect complement that specifies its subject; Nreq = number of requirement sentences.	The presence of multiple sentences makes the requirements document more difficult to read and understand
Vagueness	Ambiguity	NVag / Nreq where NVag = number of sentences including words holding inherent vagueness, i.e. words having a non uniquely quantifiable meaning; Nreq = number of requirement sentences.	The presence of vague sentences increases the level of ambiguity of the requirements document
Subjectivity	Ambiguity	Nsub / Nreq where Nsub = number of sentences that refer to personal opinions or feeling; Nreq = number of requirement sentences.	The presence of subjective sentences increases the level of ambiguity of the requirements document

METRICS	CATEGORY	FORMULA	RATIONAL
Optionality	Ambiguity	Nopt / Nreq where Nopt = number of sentences containing an optional part (i.e. a part that can or cannot be considered); Nreq = number of requirement sentences.	The presence of optional sentences increases the level of ambiguity of the requirements document. Optional phrases are defined by the authors.
Weakness	Ambiguity	Nwea / Nreq where Nwea = number of sentences containing a weak main verb; Nreq = number of requirement sentences.	The presence of weak sentences increases the level of ambiguity of the requirements document. Weak phrases are defined by the authors.
Underspecification	Specification Completion	Nusp / Nreq where Nusp = number of sentences having the subject containing a word identifying a class of objects without a specifying this class; Nreq = number of requirement sentences.	The presence of underspecification makes the requirements document not fully specified and therefore hard to understand and incomplete.
Implicity	Understandability	Nimp / Nreq where Nimp = number of sentences having a subject that is generic rather than specific; Nreq = number of requirement sentences.	The presence of implicit sentences makes the requirements document prone to be misunderstood

Table 12 Requirements metrics derived by using linguistic techniques

These metrics should be collected in order to gain an overview of the overall quality of the requirements document. However, the authors do not specify how the metrics (i.e. the measured quality attributes) are related to each other. Therefore, it is difficult to evaluate the overall quality, as it is not clear how to combine the individual measures.

4.4.3 Requirements Language

The article is focused on English as a specific natural language. The article focuses on natural language used in use cases.

4.4.4 Tools

In the following table the tools mentioned in the article that support the analysis of a requirements document regarding certain quality aspects are described.

Tool	Availability	Quality Model	Text Analysis Process	Characteristics
Quality Analyzer for Requirements Specifications (QuARS)	No. It is a research project	It is composed of a set of high-level quality properties for NL requirements to be evaluated by means of syntactic and structural indicators.	It receives as input a text document It indicates to the user those phrases that, according to the quality model, are defective. The tool does not force any corrective action. The sentences are also analyzed taking into account the particular application domain using targeted dictionaries.	- Quality model defined using literature and experience on RE and SPA according to SPICE (ISO/IEC 15504) model. - Includes a significant part of lexical and syntax-related issues of requirements documents. - The tool is designed to be adaptable
Automated Requirement Measurement Tool (ARM)	http://satc.gsfc.nasa.gov/tools/arm/	The quality model is similar to that defined by QuARS. The quality indicators have been defined on the basis of words, phrases and linguistic structures that were considered related to quality attributes. These indicators are grouped according to their indicative characteristics.	ARM scans a text file in search of default indicators.	- The user can supply indicators if domain-dependent quality indicators are known.
SyTwo	www.yana.net/sytwo/index.html	Partially adopts the QuARS quality model	It analyze the English text both to check its conformance to the rules of the Simplified English, and to detect some defect specifically conceived for evaluating the quality of requirements. The output is composed of an error code related to a predefined type of defect and where it is found.	- Web application performing the linguistic analysis of an English text - It provides the value of the Coleman-Liau metrics for readability evaluation.

Table 13 Tool to support the analysis of a requirements document

4.5 Advanced Use Case Modeling

Chapter 19: Ensuring a Successful Use Case Modeling Effort
Written by Armour, Frank. Miller, Granville
Addison-Wesley, 2000
Reference code: [AG00]

4.5.1 Overview

The purpose of the book-chapter is to discuss the quality attributes of a good use case model. The chapter focuses on the question whether the use case

model is being developed "right". The authors argue that no matter which model (very formal or informal) is used to create the use cases, the basic behavior of the system must be captured, selecting the right level of detail and to know when to stop with the analysis. This is to know when the quality set of requirements, which can be successfully specified, validated and implemented have been captured. The authors do not present quantitative metrics but questions that guide to know whether or not a quality attribute exists in the specification or in the process of managing requirements.

Furthermore, the authors suggest to select a Use Case model to capture a broad understanding of the requirements and to keep the very detailed business logic and interface specification out of the use case model. They base their quality model on the IEEE [IEEE830] definition of a well-formed requirement and Alan Davis book, Software Requirements: Objects, Functions, and States.

4.5.2 Metrics

As mentioned above, the authors do not present quantitative formulas but just questions which answers are indicators of the quality of the requirements specified with use cases. Table 14 presents a guideline to identify whether or not a use cases is being built under sufficient level of detail.

Topic	Criteria	Suggestion
Sufficient Level of Detail	Can analysis objects and their behaviors be realized relatively smoothly from the use cases or are some behaviors difficult to model with objects due to functionality that the use cases failed to model?	If no, continue use case modeling until it helps clear up the ambiguity
	Is the high-priority functionality (i.e., the functionality is expected to be developed in the first increments or releases) modeled to the point where solid, validateable requirements can be derived?	If no, more analysis needed
	Could system or acceptance test scripts be generated easily from the use cases?	If yes, the use cases are probably detailed enough.
	Can a complete set of verifiable and unambiguous requirements be derived from the use cases?	

Table 14 Criteria to decide whether a use cases is on a sufficient level of detail

In addition, the authors have selected nine quality attributes that characterize a good use case model. However, according to the author a use case should not have all the quality attributes, therefore they can be customized for specific use case modeling efforts and serve as a guideline to carry out use case walkthroughs and review sessions. That is, in each project or companies context it is necessary to decide which quality aspects should be addressed.

QUALITY ATTIBUTE	DEFINITION	QUESTIONS	TECHNIQUE
Correct	A use case model is correct if every use case contained in the model represents a piece of the system to be built.	To what extent have the users/customers been involved in the use case process? Is each use case an accurate reflection of the users'/customers' need? Why or why not? Have the users/customers given careful consideration to each use case? Are the use cases traced back to their source? Does each use case in the model trace to a business use case, BPR model, interview, or requirements workshop? Have domain experts who are independent from the users/customers reviewed the use cases? Has the project management determined that the use cases in the model are feasible and implementable? Have applicable industry standards, government regulations, and laws affecting the system's functionality been considered?	Make sure that all the use cases are reviewed and agreed on by all the stakeholders.
Unambiguous	A use case model is *unambiguous* if every use case in it has only one interpretation.	Is there a system glossary that is used in the use case model? Are all terms with multiple/unknown meanings defined (e.g., in the glossary)? Are use case activities quantifiable and verifiable? Can acceptance measures or testing criteria be assigned to each use case?	Iterative approach to developing, elaborating, and implementing the use case model is a good way to deal with ambiguity

QUALITY ATTIBUTE	DEFINITION	QUESTIONS	TECHNIQUE
Complete	A use case model is a *complete* representation of the requirements if it includes: - all significant requirements relating to behaviors, performance, design constraints, and external interfaces - for each use case, the definition of the system outputs to all actor inputs for both valid an invalid flows - full labeling and references on all diagrams, figures, tables, and other documentation, and - a definition of all terms and units of measure used	Are any behaviors missing form the use cases? Business processes? Do the system use cases map to the business use cases or business processes? Do the system use cases map to the domain and analysis object models? Are CRUD[11] or other mapping approaches used? Are all the actors participating in the use cases clearly defined? Has an interface analysis been performed? Is each interface between the system and the actors clearly defined (including information passed)? Are the expected input and output values defined for each actor interacting with the system? Are the exceptions and alternative flows documented? Are there any TBD (to be done) references? Are there any undefined terms or references? Are all sections of the use case model complete? Have performance considerations been addressed? Have security considerations been addressed? Have reliability considerations been addressed? Have capacity considerations been addressed?	Completeness does not typically occurs in a waterfall approach, therefore completeness should be judged within the context.
Verifiable	A use case model is *verifiable* if every use case stated is verifiable. A use case is verifiable if a person or machine can check in a cost-effective way that the implemented system meets the behavior contained in the use case.	Are there non-verifiable words or phrases in the use cases, such as "works well", "fast", "good performance", and "usually happen"? Are use case activities stated in concrete terms and measurable quantities? Are the preconditions and postconditions stated in a manner that can be tested?	Create acceptant test for it. If not possible the use case is ambiguous and non-verifiable.

[11] CRUD (create, read, updated, delete) matrix is used to capture interaction between objects:
- *Create:* Does the use case create an instance of this object?
- *Read:* Does the use case access this object for the purposes of reading one or more of its attributes?
- *Update:* Does the use case update any attributes of the object?
- *Delete:* Does the use case delete this object instance?

QUALITY ATTIBUTE	DEFINITION	QUESTIONS	TECHNIQUE
Consistent	A use case model is *consistent* if no two individual use cases described contain conflicting behaviors. Type of conflicts are: - a common behavior specified in two different use cases has conflicting descriptions - the characteristics of a behavior or thing conflicts. - There is a time-related conflict between two specified behaviors. - Two or more use cases describe the same behavior or object but use different terms	Is the use case model organized in a manner that encourages consistency? Have include relationships been used to factor out common behaviors? Have business function packages and dependency streams been used to organize the model? Do the preconditions and postconditions of individual use cases match up? Do any behaviors conflict? Do statements about what a behavior does contradict each other? If two behaviors conflict, are they actually two different behaviors that need to be specified? Do any terms have conflicting definitions? Are there any temporal inconsistencies? Are all use cases modeled at a consistent level of detail with respect to importance?	Inconsistencies are very often introduced by: multiple teams working on the use case model, updates to the model that create unseen conflicts, updating the model and not taking or having the time to review it for consistency, formally not defining terms. Therefore, it is important to keep good review, integration and validation practices in the use case modeling effort.
Understandable by customers and Users	Use cases should be self-explanatory. That is, the notations, models, and format of the use cases should allow the stakeholders to review, understand, and validate the use cases with a minimum amount of effort.	None mentioned	Write the use cases in the customer's and user's "language", using terms that are used in the business domain.
Extensible and Modifiable	A use case model is modifiable if changes to it can be made easily, completely, and consistently while retaining the structure and style	Is there a table of contents, glossaries, and an index? Are common use cases or behaviors, such as the same behavior appearing in multiple places in the document, factored into include relationships and cross-referenced? Is a CASE tool used to assist in maintaining the use case?	
Traceable	A use case model is *traceable* if the origin of each use case is clear and if it facilitates the referencing of each use case in future development or in enhancement documentation		It is facilitated by uniquely identifying each use case, including each extending and included use case.

QUALITY ATTIBUTE	DEFINITION	QUESTIONS	TECHNIQUE
Prioritized	A use case model is *prioritized* if the relative importance of each use case is understood.	Is each use case identified and its priority noted? Are the criteria used to prioritize clearly defined? How are the use case priorities specified (e.g., essential, conditional, optional)? Are the priorities being used to drive schedule and cost?	

Table 15: Quality attributes of use cases and possible questions to verify their fulfillment

The questions presented in the table can be used in an inspection to verify the fulfillment of the quality attribute. However, no concrete guidance is given to judge whether the attributes are fulfilled.

The quality attributes presented by the authors can be used as a baseline to define how a good use case model should look like. In a second step metrics need to be defined that measure the fulfillment of the attributes. Some of the metrics presented in the other reports can be used to measure a sub-set of the quality attributes.

4.5.3 Tools

None mentioned in this chapter

4.5.4 Requirement Language

The examples are presented in English natural language.

4.5.5 Related Authors References

There are several references that mention this book as very important to be used as a guideline to develop high quality SRS using Use Case model.

5 Conclusions and Future Work

As mentioned in the introduction section of this document, the literature survey was carried out in order to find metrics that allow to analyse the quality of the software requirements specifications. After reading quickly more than 50 references, it was clearly observable that there is still a lot of work to do regarding the complete definition of a metric program.

First of all, there is not a common agreement regarding the quality attributes of a software requirement specification. Davis et al., [DOJ93], who presented 24 quality attributes that should exist in the SRS, did the most extensible work in this field. Even the work was done 10 years ago; most of the current references still base their work on Davis's research. Second, there are many references that focus in trying to measure or improve the presence in the SRS of just one quality attribute. Ambiguity, e.g. [Lui01], [Mic01], is the attribute by excellence that most of the researches try to control and minimize. The use of linguistic technique is taking relevant important to detect and avoid ambiguity in the SRS. On the other hand, testability, traceability and correctness are also quality attribute which presence in the SRS is very important. Third, regardless if the reference focuses on one or more quality attributes, the metrics presented are very often ambiguous. The authors rarely present the meaning of the metric, the reference values and the data interpretation. Additionally, the benefit of applying certain metrics for the different point of views (stakeholders, developers, managers) is almost never mentioned. Finally there are few references that really make a difference regarding the quality evaluation of requirements written using natural language or Use Cases. In fact, the quality analysis of the use cases is done following the same quality attributes that apply for natural language: ambiguity, correctness, traceability, etc.

As per above, there is still much to do regarding metrics for software requirement specification. However, the current work done is a good foundation to create a Metric Model for Software Requirement Specification. This new model could be done following several steps: A metrics model should be developed in a company to represent its special context. Thus, the following steps should be considered always with respect to the context of the company in order to create a useful metrics model.

In an initial step the presented quality attributes and metrics should be summarized in a way that shows what are common quality aspects of a good SRS and which metrics are already at hand to measure these attributes. This overview can then be used as an input for a gap analysis; that is, it needs to be decided which quality aspects are missing and in addition which quality attributes are

not supported by a metric. In detail the metric model should be developed in the following steps. Note that with respect to other approaches, the described steps are similar to the Goal Question Metrics (GQM) approach.

1. Select and define quality attributes: This step involves the integration of all quality attributes that are desirable to find in a SRS. This step corresponds to the "Goal-Definition-Step" of the GQM approach; that is, the goals (quality aspects) that should be evaluated with the metrics should be defined. This step could also include the gap analysis; that is, compare the required quality attributes to the already existing ones and decide where new definitions are needed and how existing definitions should be tailored to the context.
2. Refine the quality attributes into sub-attributes to get a better understanding. Here the Empress-Method to refine quality aspects of a software product into more detailed aspects can be used [Empress]. This step can be compared with the "Question-Step" in the GQM approach. The questions refine the goals and ask how the goal can be achieved.
3. Consolidate current metrics for the refined quality attributes: This step involves the consolidation of all current quantitative metrics available for each quality attribute identified in the previous step. This represents a comparison with the state of the art to prevent a reinvention of the wheel. Also a gap analysis is performed, that is, an analysis where we need additional metrics.
4. Consolidate current processes, techniques to maximize the presence of the quality attributes; that is to ensure the fulfilment of the quality attribute. This step involves the analysis of all processes and techniques that are currently suggested to assure and maximize the presence of the attributes defined on the first and second step. Again, this step is important in order to become aware of the already existing techniques and processes to see what is at hand.
5. Identify tools that support the realization and measurement of the quality attributes: Many articles refers to tools that support the identification of quality attributes; therefore, this step consists of selecting all those tools and associate them to the different quality attributes.
6. Select or create new metrics for the SRS: This step involves the selection and creation of new metrics for the SRS. Table 16 presents the minimum information that should be obtained for each metric selected.

Information	Definition
Metric name	Brief title of the metric to be used
Description	Description of the metric and pre-conditions or attributes the SRS must have in order to use the metric (standard of requirements, available data, etc.)
Purpose of the metric	Specific purpose of the metric (what wants to be measured with this metric). That is, which refined quality attribute (question according to GQM) can be measured with the metric.
Which benefits will be obtained from applying the metric?	Which benefits can be obtained from applying the metric? That is, which goal can be achieved having the metric
SRS Notation	If the metric is only applicable to SRS that are written using a certain notation (Use Cases, natural language or formal specification languages), it should be specified with this element.
Formula	Formula to be applied to gain a quantitative result.
Phases	When can this metric be applied? E.g.: - RE Elicitation phase - RE Modeling phase - RE Validation phase - RE Evolution phase - RE-Analysis phase
Techniques	Which techniques can be used to obtain the data for the quantitative metric?
Reference values	Defines the values to be used in order to interpret the quantitative data obtained, e.g.: what does it mean if a document has 1000 words?
Who may be interested in the metric?	Who is interested in having this information? E.g.: DT: Developer Team TT: Testing Team RT: Requirement Team PM: Project Managers STH: Stakeholders (all persons involved in the project: users, customers, developers, etc.)
Tools	Which tool does support the presence of the quality attribute? Which tool can be used to support the applicability of the metric?

Table 16 Metrics Minimum Information

The Steps 3 – 6 are comparable the "Metrics-Step" in the GQM approach, as concrete metrics to evaluate the questions (refined quality attributes) are defined.

1. Validate the metrics: In order to know if the metrics really are meaningful and useful, it is important to validate the metrics defined in the previous step. It is important to validate them using real software requirements specifications, especially one which product has been already implemented and have met user's expectations. This is an inherent step when developing new metrics.

Besides the Metric Model, it is also worthwhile to validate the current tools used to manage requirements. These tools usually provide statistic, metrics and process that can be used to augment the quality of the SRS. The analysis of this tool can also support the selection and definition of metrics, which will support

the analysis and measurement of the quality of software requirements specifications.

Another interesting question that need to be addressed in future research activities is, how requirements metrics can be used to steer the quality assurance processes in a project. It needs to be investigated which metrics can be collected, for example in a requirements inspection and how the information gained in this process can be used to plan further quality assurance activities.

6 Literature Survey References

Below is the list of references found during the literature research regarding metrics of requirement specification.

[AF00]	Anderson, Stuart; Felici Massimo. Controlling Requirements Evolution an Avionics Case Study. Safecomp 00, LNCS1943
[AM00]	Armour, F. and Miller, G.. Advanced Use Case Modeling. Addison-Wesley, 2000
[AM02]	Anderson, Stuart; Felici, Massimo. Quantitative Aspects of Requirements Evolution. COMPSAC02, IEEE Computer Society
[ASJ01]	Anda, B., Sjoberg, D.I.K. ad Jorgensen, M.. Quality and Understandability in Use Case Models. ECOOP'2001, June 18-22, 2001, LNCS 2072 Springer-Verlag, pp. 402-428
[Bac99]	Bach, James. Risk and Requirements-Based Testing. IEEE Computer Society
[BS02]	Anda, Bente; Sjoberg, Dag I.K.. Towards an Inspection Technique for Use Case Models. Proceedings of the 14th international conference on Software engineering and knowledge engineering
[CC00]	Councill, Bill and Councill, Carol. Automating Requirements Traceability. STQE Magazine
[CL95]	Costello, Rita J.; Liu, Dar-Biau. Metrics for Requirements Engineering. J. Systems Software, 1995 Elsevier Science Inc.
[Coc00]	A. Cockburn. Writing Effective Use Cases. Addison-Wesley, 2000
[Dan01	Daneva, M.. An Assessment of the Effects of Requirements Reuse Measurements on the ERP Requirements Engineering Process. In: Dumke/Abran: New Approaches in Software Measurement, LNCS 2006, Springer Publ., 2001, pp. 172, 182
[DOJ93]	Davis, Alan; Overmyer, Scott; Jordan, Kathleen. Identifying and Measuring Quality in a Software Requirements Specification. Proceedings of the First International Software Metrics Symposium, Baltimore, May 21-22, 1993, pp. 141-152
[Dra99]	Drabick, Rodger D.. On-track Requirements. Software Testing & Quality Engineering
[DRC01]	Donzelli, P.; Rus, I.: Cantone, G.. Integrating Quality Modeling with Requirements Engineering. Proc. Of the ESCOM 2001, April 2001, London, pp. 45-53
[EO92]	Ebert, C.; Owwald, H.:. Improving Specification with Complexity Measures. Proceedings of the 3rd Int. Workshop on Rapid System Protoyping, June 23-25, research Triangle Park, NC, 1992
[Far90]	Farbey, B.. Software Quality Metrics: Considerations about requirements and requirement specifications. Information and Software Technology, 32 (1990), 1 pp. 60-64
[FFGL01]	Fabbrini, F.; Fusani, M.; Gnesi, S.; Lami, G.. An Automatic Quality Evaluation for Natural Language Requirements. 7th International Workshop on RE: Foundation for Software Quality REFSQ0'01
[FGL03]	Fantechi, A.; Gnesi, S.; Lami, G.. A Relation-based Approach to Use Case

Analysis. Ninth International Workshop on Requirements Engineering: Foundation for Software Quality. In conjunction with CAiSE'03

[FGLM02] A. Fantechi, S. Gnesi, G. Lami, A. Maccari. Application of Linguistic Techniques for Use case Analysis. IEEE Joint International Conference on Requirements Engineering (RE'02), September 09 - 13, 2002

[FN00] Fenton, Norman; Neil, Martin. Software Metrics: Roadmap. The Future of Software Engineering (Editor: Anthony Finkelstein) 22nd International Conference on Software Engineering, ACM Press ISBN 1-58113-253-0, pp.357-370, 2000

[GW89] D. Gause and G. Weinberg. Exploring Requirements: Quality Before Design. Dorset House, New Zork, 1989

[Hci02] HCi Consulting. Simple metrics for documentation. HCi Journal, March 2002

[Hur97] Hurlbut, R.R.. A Survey of Approaches for Describing and Formalizing Use Cases. Technical Report: XPT-TR-97-03, Expertech, Ltd., 1997

[IE830] IEEE Std 830-1993. Recommended Practice for Software Requirements Specification. IEEE Std 830-1993

[IE982.1] IEEE Std 982.1. IEEE Standard Dictionary of Measures to Produce Reliable Software. IEEE Std 982.1

[IE982.2] IEEE Std 982.2. IEEE Guide for the Use of IEEE Standard Dictionary of Measures to Produce Reliable Software. IEEE Std 982.2

[ISMS] ISMS. International Software Metrics Symposium. Proceedings from 3rd Conference to 7th Conference

[ISO] ISO/IEC 9126. Information Technology Software Quality Characteristics and Metrics; Part 1: Quality Model, Part 2: External Metrics, Part 3: Internal metrics. ISO/IEC 9126

[KA01] Kececi, Nihal; Abran, Alain. An Integrated Measure for Functional Requirements Correctness. Software Engineering Management Research Laboratory

[KAH00] Kantorowitz, E.; Arzi, L.; Harel, A.. A method for evaluation of the performance of requirements inspections. Proceedings of the 11th ESCOM (European Software Control and Metrics Conference) , Munich 2000

[KB96] P. Kar and M. Bailey. Characteristics of Good Requirements. INCOSE Requirements Working Group Information Paper, INCOSE Szmp., 1996

[Ken96] Kenett, R.S.. Software Specification Metrics: a quantitive approach to assess the quality of documents. 19th Convention of Electrical and Electronics Engineers in Israel, 5-6 Nov 1996

[KG99] D. Kulak and E. Guiney. Use Cases: Requirements in Context. ACM Press, New York, 1999

[Kov99] B.L. Kovitz. Practical software Requirements. Manning Greenwich, UK, 1999

[KP00] E. Kamsties, B. Paech. Taming Ambiguity in Natural Language Requirements. ICSSEA 2000, Paris, December, 2000

[Kro98] Krogstie, John. Integrating the understanding of quality in requirements specification and conceptual modeling. ACM SIGSOFT Software Engineering Notes, Volume 23 Issue 1

[Lui01] Luisa, Mich. On the use of Ambiguity Measures in Requirements Analysis. 6th International Workshop on Applications of Natural Language for Information Systems

[LV00] Le Vie, Donald S. Jr.. Documentation Metrics: What do you really want to measure?. Intercom

[MG00] Mich, L.; Garigliano, R.. Ambiguity Measures in Requirement Engineering.

	International Conference on Software Theory and Practice, ICS 2000, Beijing, China
[Mic01]	Mich, Luisa. Ambiguity identification and resolution in software development: a linguistic approach to improving the quality of systems. Seventh IEEE Workshop on Empirical Studies of Software Maintenance
[RA98]	Rolland C, Anchour C Ben.. Guiding the construction of textual use case specification. Data and Knowledge Engineering 1998; 25: 125 - 160
[RHHHW98]	Linda Rosenberg, Ph.D. , Lawrence Hyatt , Theodore Hammer , Lenore Huffman , William Wilson. Testing Metrics for Requirement Quality. QWE98
[RR99]	S. Robertson and J. Robertson. Mastering the requirements Process. Addison-Wesley, Boston, 1999
[Rul01]	Rule, P.G.. Using Measures to Understand Requirements. Proc of the ESCOM 2001, April 2001, London, pp. 327-335
[SS98]	I. Sommerville and P. Sawyer. Requirements Engineering: A good practice Guide. John Wiley & Sons, New York, 1998
[Uni-Magd]	Reiner Dumke. Suggested bibliography in Software Metrics. University of Magburg
[Whi90]	Whitty, R.W.. Research in Specification Metrics. IEEE Colloquium on Software Metrics, 5 Jan 1990
[Wie]	Karl E. Wiegers. Chapter 14: Verifying Requirements Quality. Software Requirements
[Wie99]	Wiegers, Karl. Writing Quality Requirements. Software Development, vol. 7, no. 5 (May 1999).
[Wie99]	K. Wiegers. Software Requirements. Microsoft Press, Redmond, Wash.,1999
[WRH97]	Wilson, William M.; Rosenberg, Linda H.; Hyatt, Lawrence E.. Automated Quality Analysis of Natural Language Requirement Specifications. In Proc. Int Conf. On SW Engineering, May 1997 ????

7 References

[AM00] Armour, F. and Miller, G.. Advanced Use Case Modeling. Addison-Wesley, 2000

[CL95] Costello, Rita J.; Liu, Dar-Biau. Metrics for Requirements Engineering. J. Systems Software, 1995 Elsevier Science Inc.

[DOJ93] Davis, Alan; Overmyer, Scott; Jordan, Kathleen. Identifying and Measuring Quality in a Software Requirements Specification. Proceedings of the First International Software Metrics Symposium, Baltimore, May 21-22, 1993, pp. 141-152

[IEEE SW May/June 2003] IEEE Software. *Software Engineering, Glossary.* May/June, 2003.

[Lui01] Mich, Luisa. Ambiguity identification and resolution in software development: a linguistic approach
to improving the quality of systems. Seventh IEEE Workshop on Empirical Studies of Software Maintenance

[Mic01] Mich, Luisa. Ambiguity identification and resolution in software development: a linguistic approach
to improving the quality of systems. Seventh IEEE Workshop on Empirical Studies of Software Maintenance

[RR99] S. Robertson and J. Robertson. Mastering the requirements Process. Addison-Wesley, Boston, 1999

Document Information

Title:	Software Requirement Metrics A literature survey on measurement approaches for Requirement Specifications
Date:	September 25, 2003
Report:	IESE-096.03/E
Status:	Final
Distribution:	Public