

# REFSQ'2001 PRELIMINARY WORKSHOP PROGRAM

## Seventh International Workshop on Requirements Engineering: Foundation for Software Quality

June 4-5, 2001, Interlaken, Switzerland  
(before CAISE\*2001)

Below is the program for REFSQ'2001. The times and appointments of session chairs and discussants are very tentative at this stage. Note, however, that you will probably have to arrive in Interlaken on Sunday evening, as we start early in the morning on Monday.

In case you would like to prepare as a discussant or facilitator already now, we have made the draft versions of all the papers available from this page. We will replace them with the final papers as they arrive.

Printing PDF is sometimes problematic when Adobe Acrobat is run inside your browser. If this happens, instead try to save the link directly to a file and open it outside your browser.

Noone should have to be the discussant of more than three papers. Send us an email if this has happened, and we will correct!

## MONDAY JUNE 4 2001

0900-0910 **Welcome** (*Andreas L Opdahl*)

0910- **Session 1: THE REQUIREMENTS ENGINEERING PROCESS**  
(*Chair: Andreas L Opdahl*)

*P01: Managing the Requirements Engineering Process*

L. Nguyen, P.A. Swatman

Full paper: *15 minutes' presentation and 25 minutes' discussion.*

Discussants: *Presenters of papers P02, P03 and P04 (in that order.)*

*P02: Supporting Informal Meetings in Requirements Engineering*

Andreas Braun, Bernd Bruegge, Allen Dutoit

Full paper: *15 minutes' presentation and 25 minutes' discussion.*

Discussants: *Presenters of papers P03, P04 and P01 (in that order.)*

1030-1050

**Break.**

*P03: There's More to the Story: Using Narrative Analysis to Examine Requirements Engineering As a Social Process*

Rosio Alvarez

Full paper: *15 minutes' presentation and 25 minutes' discussion.*

Discussants: *Presenters of papers P04, P01 and P02 (in that order.)*

*P04: MapExecutor: A Dynamic Enactment Support to Specify and Execute Methods with Maps*

Fernando Vélez

Full paper: *15 minutes' presentation and 25 minutes' discussion.*

Discussants: *Presenters of papers P01, P02 and P03 (in that order.)*

1210-1240

**Session discussion 1.** (30 minutes, facilitator: N.N.)

1240-1350

**Lunch.**

1350-

**Session 2: USE CASES AND SCENARIOS** (Chair: Matti Rossi)

*P05: Integrating Requirements and Architecture Information: A Scenario and Meta-Model based Approach*

Klaus Pohl, Mathias Brandenburg, Alexander Gülich

Full paper: *15 minutes' presentation and 25 minutes' discussion.*

Discussants: *Presenters of papers P07, P08 and P09 (in that order.)*

*P06: Developing Guidance and Tool Support for Rationale-based Use Case Specification*

Allen H. Dutoit, Barbara Paech

Full paper: *15 minutes' presentation and 25 minutes' discussion.*

Discussants: *Presenters of papers P08, P09 and P05 (in that order.)*

*P07: Comparing Use Case Writing Guidelines*

Karl Kox, Keith Phalp, Martin Shepperd

Full paper: *15 minutes' presentation and 25 minutes' discussion.*

Discussants: *Presenters of papers P09, P05 and P06 (in that order.)*

1550-1610

**Break.**

*P08: Scenario Networks: A Case Study of the Enhanced Messaging System*

Thomas Alspaugh, Annie I. Anton

Full paper: *15 minutes' presentation and 25 minutes' discussion.*

Discussants: *Presenters of papers P05, P06 and P07 (in that order.)*

*P09: Templates for Misuse Case Description*

Guttorm Sindre, Andreas L. Opdahl

Full paper: *15 minutes' presentation and 25 minutes' discussion.*

Discussants: *Presenters of papers P06, P07 and P08 (in that order.)*

1730- **Session discussion 2.** (Around 30 minutes, facilitator: N.N.)  
 Later... **Workshop excursion.**

## TUESDAY JUNE 5 2001

0900- **Session 3: REQUIREMENTS QUALITY AND VALIDATION** (Chair: *Camille Ben Achour - Salinesi*)

*P10: Round-Trip Prototyping for the Validation of Requirements Specifications*

Andreas Homrighausen, Hans-Werner Six, Mario Winter

Full paper: *15 minutes' presentation and 25 minutes' discussion.*

Discussants: *Presenters of papers P11, P14 and P15 (in that order.)*

*P11: An Automatic Quality Evaluation for Natural Language Requirements (Draft)*

F. Fabrini, M. Fusani, S. Gnesi, G. Lami

Full paper: *15 minutes' presentation and 25 minutes' discussion.*

Discussants: *Presenters of papers P10, P12 and P13 (in that order.)*

*P12: Variety and Quality in Requirements Documentation*

Norah Power

Position paper: *10 minutes' presentation and 15 minutes' discussion.*

Discussants: *Presenters of papers P13 and P11 (in that order.)*

1045-1105

**Break.**

*P13: Categorizing Viewpoint Contents for Discrepancy Management in Viewpoint-Based Requirements Engineering*

Andrés Silva

Position paper: *10 minutes' presentation and 15 minutes' discussion.*

Discussants: *Presenters of papers P14 and P11 (in that order.)*

*P14: Grounding Scenarios in Frame-Based Action Semantics*

Atsushi Ohnishi, Colin Potts

Position paper: *10 minutes' presentation and 15 minutes' discussion.*

Discussants: *Presenters of papers P15 and P10 (in that order.)*

*P15: Requirements Management in a Life-Cycle Perspective - A Position Paper*

Åsa Grehag

Position paper: *10 minutes' presentation and 15 minutes' discussion.*

Discussants: *Presenters of papers P12 and P10 (in that order.)*

1220-1250

**Session discussion 3.** (30 minutes, facilitator: N.N.)

1250-1400

**Lunch.**

1400-

**Session 4: MARKET-DRIVEN REQUIREMENTS ENGINEERING**  
(Chair: Klaus Pohl)

*P16: Evaluating Automated Support for Requirements Similarity Analysis in Market-Driven Development (Draft)*

Johan Natt och Dag, Bjørn Regnell, Par Carlshmare, M. Andersson, Joachim Karlsson

Full paper: 15 minutes' presentation and 25 minutes' discussion.

Discussants: Presenters of papers P17, P19 and P18 (in that order.)

*P17: Linking the Selection of Requirements to Market Value: A Portfolio-Based Approach*

Siv Sivzattian, Bashar Nuseibeh

Full paper: 15 minutes' presentation and 25 minutes' discussion.

Discussants: Presenters of papers P16, P18 and P19 (in that order.)

1520-1550

**Session discussion 4.** (30 minutes, facilitator: N.N.)

1550-1610

**Break.**

1610-

**Session 5: REQUIREMENTS ENGINEERING FOR NEW TECHNOLOGIES** (Chair: N.N.)

*P18: Why is RE for Web-Based Software Development Easier?*

Didar Zowghi, Vincenzo Gervasi

Position paper: 10 minutes' presentation and 15 minutes' discussion.

Discussants: Presenters of papers P19 and P16 (in that order.)

*P19: Requirements Engineering for Mobile Information Systems*

John Krogstie

Position paper: 10 minutes' presentation and 15 minutes' discussion.

Discussants: Presenters of papers P18 and P17 (in that order.)

1700-1730

**Session discussion 5.** (30 minutes, facilitator: N.N.)

1730-1800

**Final Discussion and Closing.** (30 minutes, facilitators: Matti Rossi and Camille Ben Achour - Salinesi.)

<http://www.ifi.uib.no/conf/refsq2001/>  
Last updated 2001.04.28  
Comments to [andreas@ifi.uib.no](mailto:andreas@ifi.uib.no)

# An Automatic Quality Evaluation for Natural Language Requirements

F. Fabbrini, M. Fusani, S. Gnesi, G. Lami  
Istituto di Elaborazione dell'Informazione del C.N.R. – Pisa, Italy

## Abstract

*This paper presents a tool called QuARS (Quality Analyzer of Requirements Specification) for the analysis of natural language software requirements. The definition of QuARS has been based on a quality model for software requirements. The quality model aims at providing a quantitative, corrective and repeatable evaluation of software requirement documents. To validate the quality model several real software requirements documents have been analyzed showing interesting results.*

## 1. Introduction

The achievement of software requirement quality is the first step towards software quality. The process leading to requirement quality starts with the analysis of the requirements expressed in natural language and continues with their formalization and verification (for example by using formal methods).

The outcome of the requirement process is a document usually composed of:

- Natural language description of the functional and non-functional requirements of the software we are developing;
- Rigorous description of these requirements by means of formal/semi-formal methods [WIN99], [MEY85] (e.g. UML, SDL, etc.).

Our experience on the formal specification and verification of software requirements has shown that many problems occur in the passage from informal requirements, expressed in natural language, to semi-formal/formal models. These problems are mainly due to the inherent ambiguity originating from different interpretations of natural language descriptions. [FAN94]

In order to reduce the risks associated to this passage, first it is necessary to have methods and tools to analyze natural language descriptions of the software/system under development. This analysis is aimed at detecting those problems that may affect the transformation of natural language requirements towards formal models.

Our purpose is to present an automatic tool for the analysis of natural language requirements documents to be used for detecting and removing defects that could cause errors in the transition to formal models.

The first step of our approach has been to define a quality model against which the requirements could be checked in order to remove ambiguities and incompletenesses. The quality model is composed of quality properties to be evaluated by means of quality indicators.

Then a tool has been realized to implement the quality model and mechanize the analysis. The tool has been called QuARS (Quality Analyzer for Requirements Specifications).

We based our work on the study of natural language requirement documents taken from industrial projects. Moreover, we took into account several works in the domain of the quality analysis of natural language requirements [LEH93], [KRO95], [MAC93], [MAT87], [MAN98]. In particular, Kansties and Paech [KAN00], starting from the consideration that

ambiguity in requirements is not only a linguistic-specific problem, proposed a check list addressing not only linguistic ambiguity but also the ambiguity related to the particular domain of the application under development. In their paper, Kansties and Paech identified the principal deficiencies of the solutions to avoid ambiguity in natural language requirements:

1. Lack of acceptance by intended users and unfeasibility: extensive lists of rules are usually ignored;
2. Lack of specificity: current inspection techniques relying on checklist-based reading are unspecific. Usually checklists are ambiguous too.
3. Unidimensionality: they help to reveal linguistic ambiguities but not ambiguities respect other requirements, the application domain and the system domain.

The methodology we present in this paper, together with the associated tool, overcomes the deficiencies pointed out by Kansties and Paech since it is easy to use (1.), points out specific potential defects in the requirements document (2.) and detects not only linguistic ambiguities but also domain-dependent ambiguities (3.).

Also Wilson and others [WIL96] face the quality evaluation of natural language software requirements. Their approach is similar to ours since they define a quality model composed of quality attributes and quality indicators, and develop an automatic tool for performing the analysis against the quality model. Nevertheless, their work does not address some important issues as the adaptability of the tool to different domains and domains. Moreover, even if their quality model takes into account both syntactic and structural aspects of the requirements document, they don't consider the semantic aspects of the quality of requirements for their analysis.

Our tool includes and mechanizes all the aspects of the requirements quality considered in the two previous works, besides, it introduces the important issue of the adaptability/tailorability to different application domains of the quality analysis.

This paper is organized as follows: in Section 2. the quality model for natural language software requirements is described. In Section 3. the functionalities and the principal characteristics of the QuARS tool are described. In Section 4. we show how QuARS can be included into the requirement process. In Section 4. we describe the adaptability characteristic by means of an example of the adaptation to a particular application domain: the security domain. Finally, in section 5, conclusions are presented.

## 2. A NL SRS Quality Model

As any other evaluation process, the quality evaluation of natural language software requirements has to be conducted against a Model. The Quality Model we defined for the natural language software requirements is aimed at providing a way to perform a quantitative (i.e. that allows the collection of metrics), corrective (i.e. that could be helpful in the detection and correction of the defects) and repeatable (i.e. that provides the same output against the same input in every domains) evaluation.

The adopted approach has been mainly top-down. In fact, starting from the identification of a set of representative and meaningful high level properties to be evaluated, we defined quality indicators directly detectable and measurable on the requirement document.

The high level properties of the Quality Model are:

- **Testability:** the capability of each Requirement to be assessed in a pass/fail or quantitative manner.
- **Completeness:** the capability of the Requirements to refer precisely identified entities.

- **Understandability:** both the capability of each Requirement to be fully understood when used for developing software and the capability of the Requirement Specification Document to be fully understood when read by the user.
- **Consistency:** the capability of the Requirements to avoid potential or actual discrepancies.

The above properties don't cover all the possible quality aspects of software requirements but they are sufficiently specific for being applied (with the support of an automatic tool) to comparing and verifying the quality of requirement documents. Furthermore, this set of properties is sufficient to include a large part of the syntax-related issues of a requirement document along with a number of interesting semantics-related issues.

The above properties can be automatically evaluated by means of tangible Indicators. Indicators are syntactic or structural aspects of the requirement specification documents that provide information on a particular property of the requirements themselves. Several Indicators have been included in the Quality Model, each associated to a Property (see Table 1.).

The Indicators can be counted/detected during the parsing of the requirement document. The detection of an Indicator in the requirement document points out a potential problem related to the correspondent Property so that corrective actions may be easily done.

For doing that both the single sentences of the Requirement Document and the whole Requirement Document structure are evaluated. In fact, some Quality Indicators investigate the quality of the Requirement Document, while others investigate the quality of the single component of the document itself (i.e. the requirement sentences). In other words, the Quality Indicators of the Quality Model differ for their scope: some of them need to analyze single sentences for being calculated, others need to analyze the whole Requirement Document. Figure 1 shows that the object of the Quality Evaluation is the Requirement Document intended both as a whole and a set of sentences. The final Quality Evaluation is the union of the found defects regarding the Requirement Document structure and the single sentences.

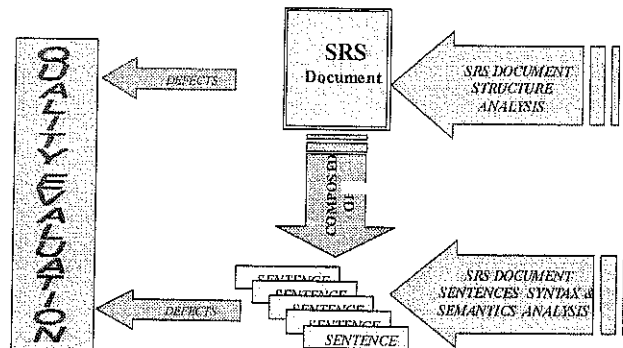


Figure 1. The Quality Evaluation of a Requirement Document

The following tables describe the quality model, in particular:

Table 1 describes the Indicators related to the Quality Properties along with examples of the keywords to be used for detecting potential defects in the NL SRS.

Table 2 provides some examples of requirements sentences containing defects detectable by using our quality model.

Table 3 shows the scope of each Quality Indicator of the Quality Model.

The Indicator keywords used for detecting Indicators in the requirement document have been defined after the analysis of a number of requirement documents taken from industrial

projects and on the basis of our experience in the Requirement Engineering [FAB98], [FAB00], [FAN97], [LAM99].

PROPERTY	INDICATOR	DESCRIPTION	NOTES
TESTABILITY	Optionality	An Optionality Indicator reveals a requirement sentence containing an optional part (i.e. a part that can or cannot be considered)	Optionality-revealing words: <i>possibly, eventually, if case, if possible, if appropriate, if needed, ...</i>
	Subjectivity	A Subjectivity Indicator is pointed out if sentence refers to personal opinions or feelings	Subjectivity-revealing wordings: <i>similar, better, similarly, worse, having in mind, take into account, take into consideration, as [adjective] as possible</i>
	Vagueness	A Vagueness Indicator is pointed out if the sentence includes words holding inherent vagueness, i.e. words having a non uniquely quantifiable meaning	Vagueness-revealing words: <i>clear, easy, strong, good, bad, efficient, useful, significant, adequate, fast, recent, near, far, close, in front, ...</i>
	Weakness	A Weakness Indicator is pointed out in a sentence when it contains a weak main verb	Weak verbs: <i>can, could, may</i>
COMPLETENESS	Under-specification	An Under-specification Indicator is pointed out in a sentence when the subject of the sentence contains a word identifying a class of objects without a modifier specifying an instance of this class	This indicator deals with the syntactic and semantics of the sentence under evaluation
UNDERSTANDABILITY	Implicitity	An Implicitity Indicator is pointed out in a sentence when the subject is generic rather than specific.	Subject expressed by means of: • Demonstrative adjective ( <i>this, these, that, those</i> ) or Pronouns ( <i>it, they, ...</i> ) Subject specified by means of: • Adjective ( <i>previous, next, following, last, ...</i> ) or Preposition ( <i>above, below, ...</i> )
	Multiplicity	A Multiplicity Indicator is pointed out in a sentence if the sentence has more than one main verb or more than one direct or indirect complement that specifies its subject	Multiplicity-revealing words: <i>and, or, and/or, ...</i>
	Comment Frequency	It is the value of the CFI (Comment Frequency Index). [CFI= NC / NR where NC is the total number of Requirements having one or more comments, NR is the number of Requirements of the RSD]	-
	Readability Index	It is the value of ARI (Automated Readability Index) [ARI= WS + 9*SW where WS is the average words per sentence, SW is the average letters per word]	-
	Directives Frequency	It is the rate between the number of SRS and the pointers to figures, tables, notes, ...	-
	Unexplanation	An Unexplanation Indicator is pointed out in a RSD (Requirement Specifications Document) when a sentence contains acronyms not explicitly and completely explained within the RSD itself	-
CONSISTENCY	Under-reference	An Under-reference Indicator is pointed out in a RSD (Requirement Specifications Document) when a sentence contains explicit references to: • not numbered sentences of the RSD itself • documents not referenced into the RSD itself • entities not defined nor described into the RSD itself	-

Table 1: Quality Properties and their Indicators

INDICATOR	NEGATIVE EXAMPLE
Implicitly	• the <u>above</u> requirements shall be verified by test
Optionality	• the system shall be such that the mission can be pursued, <u>possibly</u> without performance degradation
Subjectivity	• <u>in the largest extent as possible</u> , the system shall be constituted by commercially available software products
Vagueness	• the C code shall be <u>clearly</u> commented
Weakness	• the results of the initialization checks <u>may</u> be reported in a special file
Underspecification	• the system shall be able to run also in case of <u>attack</u>
Multiplicity	• the mean time needed to remove a faulty board <u>and restore service</u> shall be less than 30 minutes
Undereference	• the software shall be designed <u>according to the rules of the Object Oriented Design</u>
Unexplanation	• the handling of any received valid TC packet shall be started in less than 1 CUT

Table 2: Examples of requirement sentences containing defects

Quality Indicator	Requirement Document Sentences	Whole Requirement Document
Comment Frequency		•
Directives Frequency		•
Implicitly	•	
Multiplicity	•	
Optionality	•	
Readability Index		•
Subjectivity	•	
Under- specification	•	
Under-reference		•
Unexplanation		•
Vagueness	•	
Weakness	•	

Table 3. Scope of the Quality Indicators

### 3. QuARS: Quality Analyzer for Software Requirement Specifications

In order to make the analysis of natural language software requirements automatic, a tool has been implemented at CNR-IEI. The tool is named QuARS (Quality Analyzer of Requirements Specifications) (see figure 3) and has been realized to parse requirement sentences written in Natural Language and to point out a number of potential sources of errors into the requirements themselves.

QuARS has been developed in the framework of a project<sup>1</sup> that aims at transferring technology to small and medium enterprises (SME) for improving their software development process.

The QuARS tool is composed of the following main logic modules:

- Lexical Analyzer
- Syntax Analyzer
- Quality Evaluator
- Special purpose grammar
- Dictionaries

The phases of the SRS Quality Evaluation made by the QuARS tool are described below:

- The files containing the SRS Document are analyzed by the lexical Analyzer in order to verify if a correct English Dictionary has been used.
- The output of the Lexical Analyzer (i.e. the lexical category associated to each word of the sentences) is the input of the Syntactical Analyzer. It, using a special purpose grammar, builds the derivation trees of each sentence.
- The set of derived trees is the input of the Quality Evaluator module of the QuARS tool. The Quality Evaluator module receives also the special Dictionaries as input. These Dictionaries contain the words and the syntactical elements that allow the detection of inaccuracies in the SRS sentences (see for example the Notes column in the Table 1). The Quality Evaluator module, according to the rules of the Quality Model and by reading the dictionaries, performs the evaluation of the sentences. Finally, it provides the user with Warning Messages, that are able to point out those sentences of the SRS Document having potential defects.

Figure 2. depicts the operation of the QuARS tool.

QuARS has been designed with the aim to match mainly the following characteristics:

- Ease to use: the tool has to be usable with little effort both in terms of people training needed and time consumed.
- Generality: the tool shall run independently of the format of the SRS document.
- Flexibility/tailorability: the tool has to be modifiable in order to make it more effective for particular application domains

---

<sup>1</sup> The LINK project (MURST L.488/92), with the partial support of SSSUP S.Anna, Pisa, Italy.

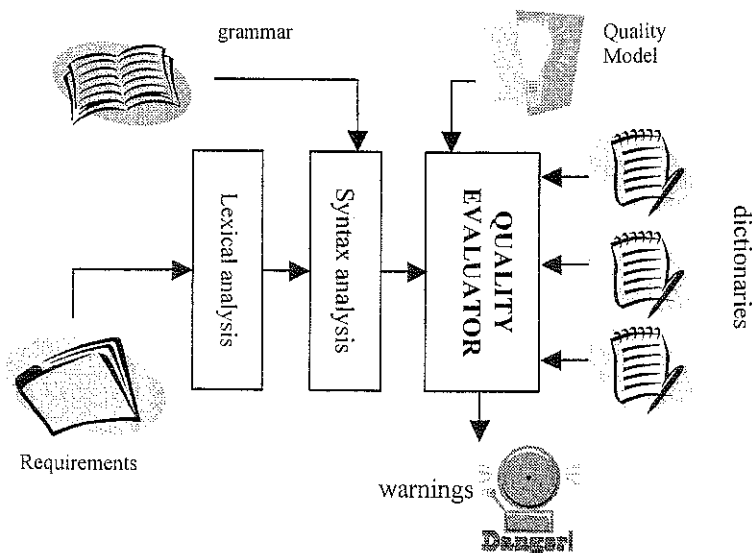


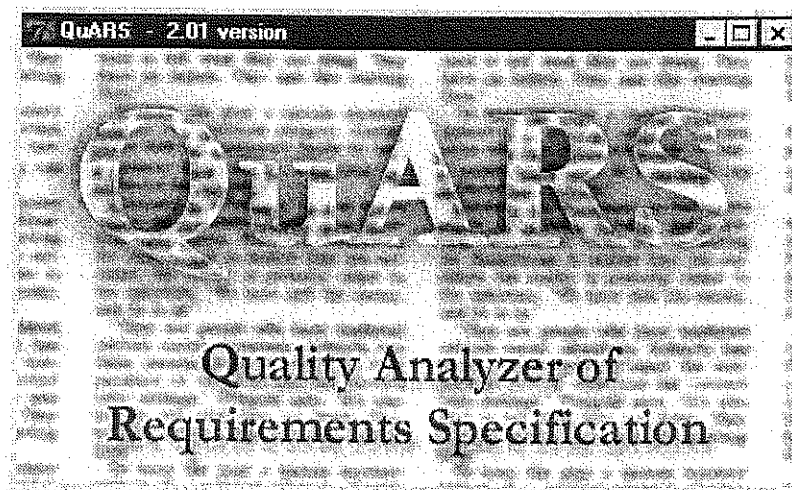
Fig. 1 Scheme of the QuARS tool operation

In the following we will describe how the main target characteristics (ease to use, generality and flexibility) have been implemented.

*Ease to use*

Despite of QuARS has his own text interface, a TCL/TK [BRE97] graphical interface has been released to simplify the use of the tool and to enhance his characteristics (see figure 4).

The script language TCL/TK has been used because of his portability and the very fast way to develop, verify and debug the code: this fits good with the building a not integrated



interface for a tool with his own independent behavior and environment.

Fig. 3: The QuARS tool logo

The QuARS GUI (Graphic User Interface) allows the user:

- to do an easy and fast navigation in the file system for selecting the requirements file to analyze,
- to edit and save the specification file to analyze,
- to remember the latest working directory,
- to edit and save all the QuARS dictionaries using the simple but complete built-in editor,
- to easy choose the work session to perform,
- to save into file the results clearly visualized in the wide central window for a future use,
- to see the statistics about the current session work,
- to illustrate him, an on line help, the analysis purpose and the functionality of the graphical interface itself.



Fig. 4: QuARS: the user interface

### Generality

The expected format of the requirements sentences to be evaluated is the text format. This format allows to achieve an high generality because it is always possible to save every format as text format.

The risk associated to this strategy is to left some information related to the particular layout or formatting of the SRS document. For example, if multilevel indented bullet lists are used, after the transformation in text format, the hierarchy represented by means of the different

levels of indentation will be lost. On the other hand, it can be assumed that in every requirement documents the hierarchy of the requirements isn't be established by means of the formatting of the text, but it has to be defined by means of appropriate numeration of the sentences. Then the possible lack of information doesn't compromise the validity of the textual requirement document.

### *Flexibility/Tailorability*

The flexibility/tailorability target characteristic is very important for the use of QuARS in industrial domain. In fact, it is important to adapt QuARS to different projects and different application domains.

The way QuARS does it is by allowing the used to evolve and modify the dictionaries. Dictionaries are directly used for the detection of several Indicators, and in some cases the content of the dictionaries is strictly dependent of the application domain of the requirements document under evaluation. Some Indicators are most dictionary-sensitive than others (e.g. Vagueness and Under-specification)

For these Indicators, the possibility to adapt the related dictionary is necessary to assure a meaningful evaluation.

## **4. The QuARS tool in the Requirements Process**

In this section we discuss how the QuARS tool could be integrated within the software requirement process and in particular the software requirements evaluation process.

A typical evaluation of a requirement document is performed by a team of expert reviewers who, by analyzing the SRS documents, could detect ambiguities, inconsistencies or, in general, inaccuracies. Other actors may be involved in setting up the requirement document, but at evaluation (review) time the reviewer is the key actor.

QuARS can be helpfully included in the software requirements production and quality evaluation process (see figure 5).

The QuARS tool, for its nature, can be tuned up in order to make it more adequate to particular application domains or types of software requirements. Then the inclusion of the QuARS tool in the software requirements quality evaluation process allows the reviewer to change the Quality Model if necessary in order to make it more respondant to the particular SRS under evaluation. The conceptual changes on the Quality Model can be easily transferred on QuARS for its flexibility. Finally, the corrective actions are made on the SRS document and the process can be re-started on the modified document. When the requirement document passes the judgment of the Requirements Quality Evaluation Team it will be approved and injected in the development process.

We run QuARS on real requirement documents taken from industry software projects.

The requirements documents run with the QuARS tools are given from different application domains:

- S1. Business Application: Functional Requirements of a Transaction and Customer Service (TACS) Check Cashing module;
- S2. Space Software Application: Functional Requirements of a sub-system of a space vehicle;
- S3. Telecommunication Application: Requirements Specification of a project aiming for a new generation STM switches;

S4. Security Application: Functional Security Requirements for an Application Level Firewall Protection Profile

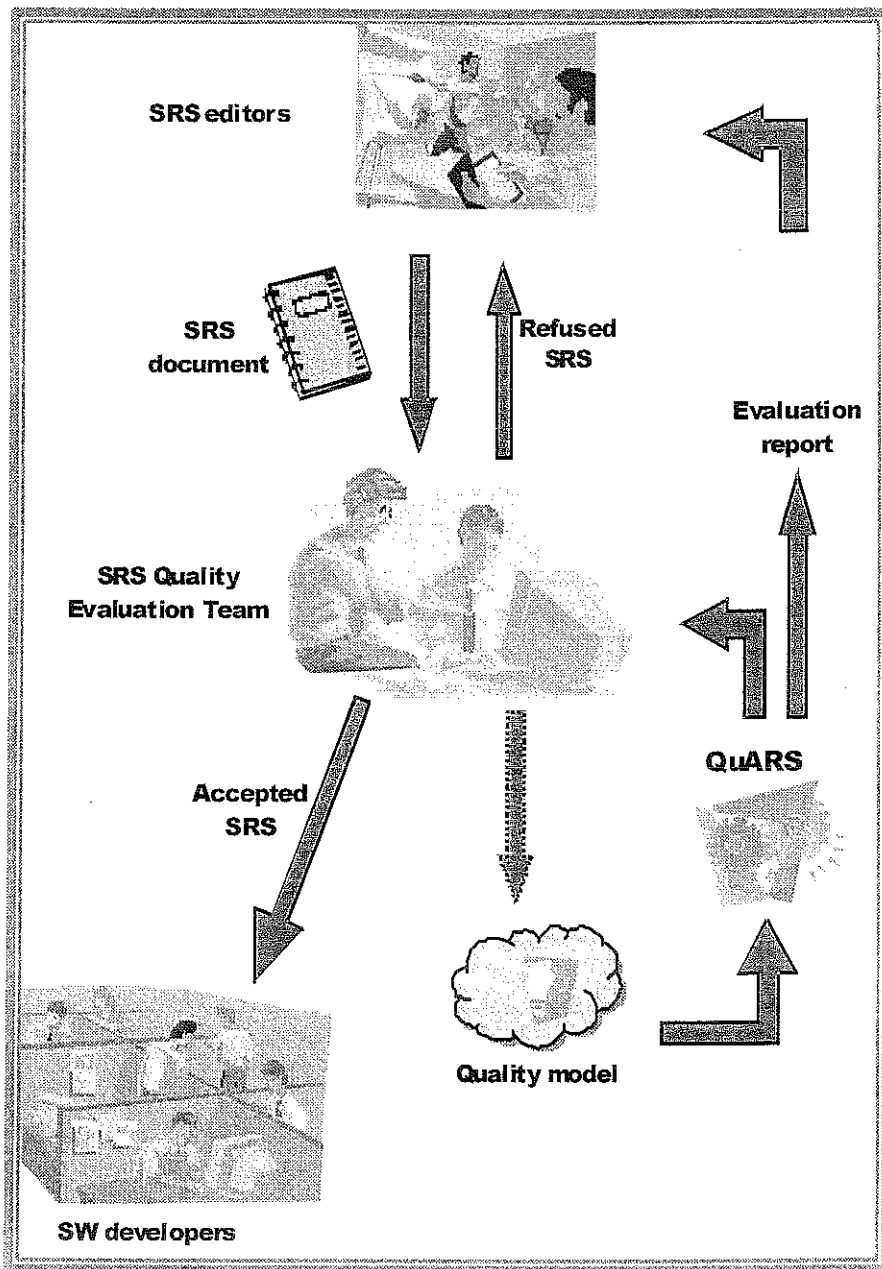


Fig. 5: The use of QuARS into the SRS process

The outcomes of this kind of validation have been interesting. They are shown in the Figure 6 and Figure 7. Figure 6 shows the rate of defects occurrences on the total requirements for each evaluated documents. Figure 7 shows the percentage distribution of defects types detected for each of the four requirements evaluated.

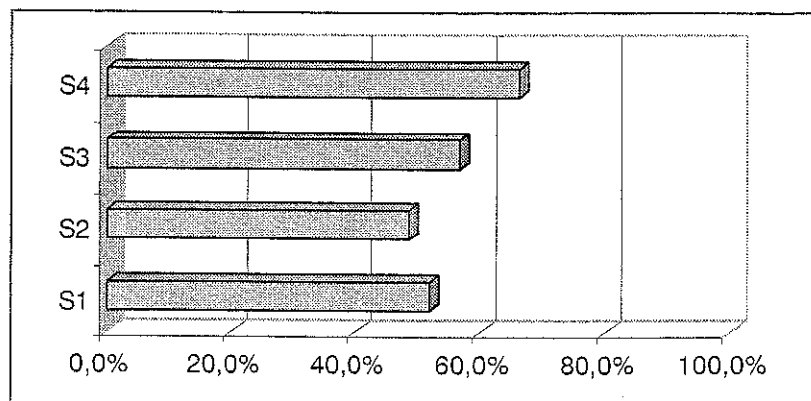


Fig. 6. Rate of defects occurrences on the total requirements

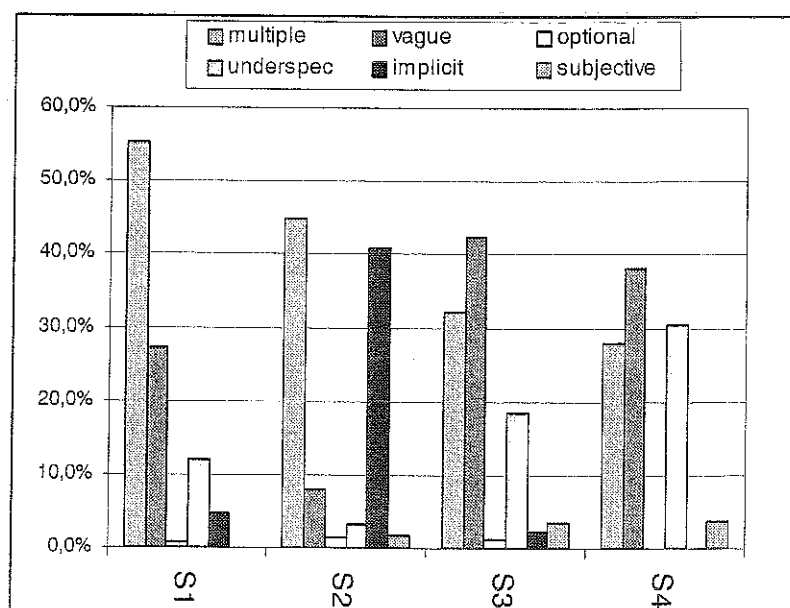


Fig. 7: Percentage distribution of defects types detected

What we can note is, first, that the rate of defects detected is at least the 50% for each requirement document evaluated. It means that the Quality Model (and the QuARS tool that implements it) can be considered effective in defect detection. It is also possible to observe that the multiplicity and vagueness and under-specification indicators are most frequently pointed out.

Requirements engineers from the involved industries found these results useful and interesting for improving their requirement documents

## 5. An Example of Tailoring of the QuARS tool: Analysis of Software Security Requirements

Before to perform an evaluation of a new requirement document it is necessary to adjust the QuARS tool in order to make it more suitable for evaluating the new requirements.

In order to verify if the expected flexibility of this tool is really applicable to requirements dealing with a particular application domain, QuARS has been used for evaluating requirements dealing with a particular application domain: security applications.

With this aim some Special Dictionaries of QuARS have been reviewed and modified in order to make them more suitable to an environment where IT Security requirements are essential and then to make QuARS more effective for this environment.

The changes made for tailoring the QuARS tools regarded in particular the dictionary associated to the Under-specification Indicator. Some special words, indicating classes of objects that in a security environment needed a modifier for being uniquely specified, have been added.

The security-tailored QuARS tool has been then used for analyzing the SI. Requirements Document "U.S. Government Application-Level Firewall Protection Profile for Low-Risk Environments" [USG98] document, that defines the basic security requirements of U.S. Government organizations handling unclassified information in low-risk environment. This analysis aims to check if this document, that is conform to Common Criteria Part 2 and Part 3 [CCr98], contains lacks of quality according to our Quality Model that can cause errors in the subsequent phases of the software development.

The changes made for adapting QuARS to a security domain consist in the addition of special words in the Under-specification-related dictionary. In the following a sample of the security-related words added to the Under-specification dictionary is provided:

#### ***Key***

This word in a security environment means: "a long string of seemingly random bits used with cryptographic algorithms to create or verify digital signatures and encrypt or decrypt messages and conversations. [ABR95]

Nevertheless, in this environment it is necessary to specify which type of key we are considering. In fact, different types of key exist: private key, public key, secret key, etc.

#### ***Access***

This word in a security environment means: "A specific type of interaction between a subject and an object that results in the flow of information from one to the other. [DoD85]

Nevertheless, in this environment it is necessary to specify which type of access we are considering. In fact, different types of access exist: write access, remote access, authorized access, unauthorized access, etc.

#### ***Security policy***

It means: "A set of laws, rules, and practices that regulate how an organization manages, protects, and distributes sensitive information". [NCS88]

Nevertheless, in this environment it is necessary to specify the domain a security policy is referred to. In fact, different security policy domains may exist: Corporate security policy, System security policy, Technical security policy.

#### ***Attack***

This word in a security environment means: "The act of trying to bypass security controls on a system. An attack may be active, resulting in the alteration of data; or passive, resulting in the release of data." [NCS88].

Nevertheless, in this environment it is necessary to specify which type of attack we are considering. In fact, different types of attack exist: physical attack, simple attack, complex attack, etc.

It is now possible to understand why the under-specification of these words can cause bad interpretation of the requirements and then introduce risks in the project. This is the reason why the QuARS tool points out the occurrences of these words when they have a loss of specification.

## 5. Conclusions

This paper first described a Quality Model defined for the evaluation of Natural Language Software Requirement Specifications composed of linguistic quality properties of Requirement Documents.

This Quality Model has been defined for generic SRS but with the characteristic of being highly adaptable to different domains. Some Security Requirement Specification documents have been analyzed using the Quality Model defined.

Many problems are usually present in SRD due to an imprecise use of natural language and the presented tool helps in revealing a consistent number of them in an easy way.

## 6. References

- [ABR95] M. Abrams, S. Jajodia, H. Podell, eds, *Information Security – An integrated Collection of Essays*, IEEE Computer Society Press, January 1995.
- [BRE97] *Practical Programming in Tcl and Tk*” second edition Prentice Hall 1997.
- [CCr98] *Common Criteria for Information Technology Security Evaluation*, CCIB-98-026 Version 2, May 1998.
- [DoD85] Dept. Of Defense Standard, Department of Defence Trusted Computer System Evaluation Criteria, DOD 5200.28-STD, GPO 1986-623-963, Dec. 26, 1985.
- [FAB00] F.Fabbrini, M.Fusani, S.Gnesi, G.Lami “Quality Evaluation of Software Requirement Specifications”, *Proc of Software & Internet Quality Week 2000 Conference.*, San Francisco, CA May 31-June 2 2000, Session 8A2.
- [FAB98] F.Fabbrini, M.Fusani, V.Gervasi, S.Gnesi, S.Ruggieri. On linguistic quality of Natural Language Requirements. In 4<sup>th</sup> International Workshop on Requirements Engineering: Foundations of Software Quality REFSQ’98, Pisa, June 1998.
- [FAN97] A. Fantechi, M.Fusani, S.Gnesi, G.Ristori. Expressing properties of software requirements through syntactical rules. Technical Report. IEI-CNR, 1997.
- [FAN94] A. Fantechi, S. Gnesi, G. Ristori, M. Carenini, M. Vanocchi, P.Moreschini, "Assisting requirement formalization by means of natural language translation", *Formal Methods in System Design*, vol 4, n.3, pp. 243-263, Kluwer Academic Publishers, 1994.
- [GNE00] S.Gnesi, D.Latella, G.Lenzini, A “Brutus” Model Checking of Spi-Calculus Dialect, *Proc.of the Workshop on Formal Methods and Computer Security*, FMCS 2000, 2000.
- [KAN00] E.Kamsties, B.Peach Taming Ambiguity in Natural Language Requirements. ICSSEA 2000-5.
- [KRO95] J. Krogstie, O.I. Lindland, G. Sindre. Towards a deeper understanding of quality in requirements engineering. In 7<sup>th</sup> International CAiSE Conference, vol. 932 of *Lecture Notes in Computer Science*, pages 82-95, 1995.
- [LAM99] G. Lami. Towards an Automatic Quality Evaluation of Natural Language Software Specifications. Technical Report. B4-25-11-99. IEI-CNR, 1999.

- [LEH93] F.Lehner. Quality control in software documentation based on measurement of textcomprehension and text comprehensibility. *Information Processing & Management*, vol; 29, No. 5, pp 551-568, 1993.
- [MAC93] B.Macias, S.G. Pulman. Natural Language processing for requirements specifications. In Redmill and Anderson, *Safety Critical Systems*, pages 57-89. Chapman and Hall, 1993.
- [MAN98] M.Mannion, B.Keepence, D.Harper. Using viewpoints to define domain requirements. *IEEE Software*, January-February 1998, pages 95-102.
- [MAT87] K.Matsumura, H.Mizutani, M.Arai. An application of structural modelling to software requirements analysis and design. *IEEE Transactions on Software Engineering*, vol. SE-13, No.4, April 1987.
- [MEY85] B.Meyer. On formalism in specifications. *IEEE Software*. January 1985, pages 6-26.
- [NCS88] Nat'l Computer Security Center, Trusted Network, Glossary of Computer Security Terms, NCSC-TG-004, Oct. 1988.
- [USG98] U.S. Government Application-Level Firewall Protection Profile for low-Risk Environment, Version 1.b, September 1998..
- [WIL96] W.M.Wilson, L.H. Rosenberg, L.E. Hyatt. Automated quality analysis of Natural Language requirement specifications. *PNSQC Conference*, October 1996.
- [WIN99] J.M. Wing, J. Woodcock, J. Davies (eds.) *FM'99 – Formal Methods*, vol. I and II LNCS 1708, 1709, Springer.

# REPUBBLICA ITALIANA



## IN NOME DELLA LEGGE

NOI PROF. DOTT. ING. ARRIGO VALLATTA  
RETTORE DEL POLITECNICO DI MILANO

VEDUTI GLI ATTESTATI DEGLI STUDI COMPIUTI

DA **PELLEGRINI MARCO**

NATO A FIDENZA (PARMA) L'11 GIUGNO 1961

VEDUTO IL RISULTATO DELL'ESAME DI LAUREA DA LUI SUPERATO CON LODE IN QUESTO POLITECNICO IL 7 APRILE 1986  
GLI CONFERIAMO LA LAUREA DI

## DOTTORE IN INGEGNERIA ELETTRONICA

IL PRESENTE DIPLOMA DI LAUREA VIENE RILASCIATO A TUTTI GLI EFFETTI DI LEGGE

REGISTRATO AL N. 52057 E AL N. 4298

DATO A MILANO L'11 APRILE 1986

IL DIRETTORE AMMINISTRATIVO  
(GESUALDA ASSANTE)

*Gesualda Assante*

IL RETTORE

(ARRIGO VALLATTA)

*Arrigo Vallatta*

IL PRESIDE DELLA FACOLTA

(EMILIO MASSA)

*Emilio Massa*