Information and Software Technology 67 (2015) 180–195

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

A methodology for the classification of quality of requirements using machine learning techniques

Eugenio Parra *, Christos Dimou, Juan Llorens, Valentín Moreno [1], Anabel Fraga [1]

Knowledge Reuse Group, University Carlos III de Madrid, Av. de la Universidad, 30, 28911 Leganés, Madrid, Spain

ARTICLE INFO

Article history:
Received 18 January 2015
Received in revised form 1 June 2015
Accepted 13 July 2015
Available online 20 July 2015

Keywords:
Software engineering
Requirements engineering
Requirements quality
Machine learning

ABSTRACT

Context: One of the most important factors in the development of a software project is the quality of their requirements. Erroneous requirements, if not detected early, may cause many serious problems, such as substantial additional costs, failure to meet the expected objectives and delays in delivery dates. For these reasons, great effort must be devoted in requirements engineering to ensure that the project's requirements results are of high quality. One of the aims of this discipline is the automatic processing of requirements for assessing their quality; this aim, however, results in a complex task because the quality of requirements depends mostly on the interpretation of experts and the necessities and demands of the project at hand.
Objective: The objective of this paper is to assess the quality of requirements automatically, emulating the assessment that a quality expert of a project would assess.
Method: The proposed methodology is based on the idea of learning based on standard metrics that represent the characteristics that an expert takes into consideration when deciding on the good or bad quality of requirements. Using machine learning techniques, a classifier is trained with requirements earlier classified by the expert, which then is used for classifying newly provided requirements.
Results: We present two approaches to represent the methodology with two situations of the problem in function of the requirement corpus learning balancing, obtaining different results in the accuracy and the efficiency in order to evaluate both representations. The paper demonstrates the reliability of the methodology by presenting a case study with requirements provided by the Requirements Working Group of the INCOSE organization.
Conclusions: A methodology that evaluates the quality of requirements written in natural language is presented in order to emulate the quality that the expert would provide for new requirements, with 86.1 of average in the accuracy.

1. Introduction

The development of any software project must be based on a high quality of requirements engineering process. Requirements engineering is a "systematic process of developing requirements through an iterative co-operative process of analyzing the problem, documenting the resulting observations in a variety of representation formats, and checking the accuracy of the understanding gained" [1]. Low quality of requirements can provoke errors during the development of a project, since they are considered to be the most costly to correct, if they are not detected on time.

Depending on the project, there could exist systems that need a large quantity of requirements and their specification could include different roles, such as: users or clients, design engineers, and development groups. For this reason, one of the approaches in Requirements Engineering is to establish standards and guides in order to facilitate the specification and improvement of quality of requirements. In [2] the authors "synthesized the different quantitative indicators of qualitative desirable properties of the requirements"; these properties are: verifiability, validability, modifiability, consistency, completeness, unambiguity, understandability, traceability, abstraction, precision and atomicity.

Although these characteristics are clearly defined, the quantification of the value of each characteristic for a requirement is a complex task. In this direction, various studies have been realized for the automatic calculation of these values, using quality metrics over the requirements and establishing quality levels for these requirements [2–4].

* Corresponding author.
E-mail addresses: eparra@kr.inf.uc3m.es (E. Parra), cdimou@kr.inf.uc3m.es (C. Dimou), llorens@kr.inf.uc3m.es (J. Llorens), vmpelayo@kr.inf.uc3m.es (V. Moreno), afraga@kr.inf.uc3m.es (A. Fraga).
[1] V.M. and A.F. should be considered as co last-author.

http://dx.doi.org/10.1016/j.infsof.2015.07.006
0950-5849/

However, one of the most decisive factors for the quality of requirement must be the evaluation of the involved domain experts, because their evaluation is strongly linked to the needs and necessities of the project. Today, the most widely used techniques for the analysis of requirements, that calculate quality metrics automatically, do not take into account the experts' interpretation of quality when they define the quality levels of the requirements, and they are, therefore, less flexible for adaptation at the needs of different projects.

In this work, we present a methodology for the evaluation of the quality of requirements in an automatic way, according to the quality criteria posed by the domain expert who uses this methodology. The objective of this methodology is the emulation of the prediction of the quality of new requirements that are entered in the system. In order to achieve this goal, the expert must contribute with an initial set of requirements that have been previously classified according to their quality and that have been chosen as appropriate for establishing the degree of exigency for the project. For each of the requirements in the given set, we extract metrics that quantify the value for the quality of the requirement. The methodology proposes the use of a Machine Learning technique, namely rule inference, for learning the value ranges for the metrics, as well as the way they are combined to result into the interpretation of requirements quality of the domain expert.

For the demonstration of the reliability of the proposed methodology, we examine a case study of a requirements system provided by the INCOSE's Requirements Working Group and we present two approaches of implementation of the methodology, obtaining different results in accuracy and efficiency in resolving problems.

2. State of the art

2.1. Machine learning techniques

In this paper, we use rule induction techniques which is part of the field of machine learning. The rule induction techniques receive as input a data set composed of the attribute group and a value of class. This set forms the set of learning examples. The purpose of the rule induction techniques is to generate a decision tree or a set of rules for determining the classification of new samples [5].

The reason to use these techniques is because they are robust against noise, caused by insufficient data or errors in both attributes and in the allocation class value. They have the ability to identify irrelevant attributes, as well as to detect discriminating, absent or empty attributes. Extracting rules of the results provided by these techniques allows easy interpretation, facilitating rule modification or elimination for a set [6].

The rule induction techniques used in this work are:

- *C 4.5:* A technique that generates rules from a previously generated decision tree [7].
- *PART:* This algorithm is a combination of C4.5 and RIPPER rule learning [8]. RIPPER implements a propositional rule learner that initially builds a rule set until all positive examples are covered and improves its fit to the training data through a pruning process over the set [9].
- *Boosting and Bagging:* these algorithms are a combination of homogeneous classifiers belonging to so-called "set of classifiers". Classifier sets aim to improve the accuracy of individual classifiers that are generated using machine learning algorithms [10]. Homogeneous classifiers are generated based on the same learning algorithms. In this paper the algorithms used are based on C4.5 and PART.

2.2. Management of requirements

Many of the errors in software development are caused by deficiencies in the elicitation, specification and analysis of requirements. These errors are the most expensive and most difficult to correct if they are not detected in the early phase of the project [11–15]. This demonstrates the importance of quality of requirements in the development of a project.

One of the phases in the requirement engineering process is specification. The specification of any requirement must be in a complete, consistent and correct form. To ensure that requirements comply with the quality metrics, there are several approaches based on conceptual modeling [16–18] in order to detect errors and provide a more formal specification improving the quality. But, the first step to perform the model is the specification of the requirement in natural language. Two advantages of this are: there is no limitation on the concepts that can be expressed; and sentences and grammatical structure provide means of tracing meaningful elements. [19].

Based on high quality requirements written in natural language, the conceptual models are performed with a good quality improving even more the final quality of the requirements.

In order to ensure the quality of requirement written in natural language, various books and articles have been published that the structure and quality metrics to be taken into account in each of the requirements [20–24]. International organizations, such as the IEEE and ISO, have issued several standards for specification of quality of requirements [25–27]. Moreover, Ivy F. Hooks published for NASA the "Guide for managing and writing requirements" [28] and the European Space Agency (ESA) has published a similar guide for the specification of requirements [29]. Finally, the International Council on Systems Engineering [30] dedicated to the development and progress of the systems engineering, published in 2012 another guide for writing quality of requirements [19].

Also with the aim of assisting in the creation and development of projects, there are software tools that allow the management of requirements [31–35].

The main features that provide the tools for the management of requirements are:

- Storage Management.
- Help is the specification.
- Traceability.
- Validation.
- Quality Management.

2.3. Automation in the quality assessment of requirements

Research in requirements engineering has led to current studies that assess in an automatic way the properties that affect the quality of the requirements by analyzing their language and composition. There are studies where attention is focused on the detection and evaluation of a single feature. Some papers detect ambiguities [36–39], inconsistencies [40] and conflicts [41].

In [42] authors present a set of automated analysis mechanisms to support the requirements engineers to detect and analyze modeling errors in contextual requirements models. They present a tool called Re-Context, which supports analysts by checking consistency and conflicts in the requirements models.

The work [43] presents a model-based requirements verification method, called NLtoSTD, to verify requirements documents. The method transforms natural language requirements into a State Transition Diagram that can help to detect and to eliminate ambiguities and incompleteness. The authors assess the approach

performing two controlled experiment and the results are compared against a standard fault-checklist-based requirement inspection method.

Other studies develop quantitative methods to evaluate the quality of requirements. Through processing language requirements can be formalized features that should or should not meet to assign a quality measure. Some of these studies present software tools that perform the quantification of these characteristics automatically [2–4]. Using quantitative methods, it is possible to improve the quality of the project by automatically prioritizing requirements [44,45].

The Software Assurance Technology Center of NASA proposes the conjunction of several tools developed by NASA, intended to the engineering of high quality of requirements [46]. These tools are: ARM (Automated Requirements Measurement), SCAT (Safety Critical Analysis Tool) and RUT (Requirements Use case Tool).

In the work [47], Ott presents an automatic method for the categorization of requirements presented in various documents. This method analyzes the text of a requirement and establishes a category that should match the theme of the document to which it belongs.

To solve the classification of requirements into various topics, the work [48] proposes a method which automatically classifies requirement sentences into each topic category using only topic words as representative of the analysts' views. Authors evaluate the effectiveness of the proposed technique by performing experiments using two real requirements data sets.

Some studies use artificial intelligence techniques to analyze information concerning the quality of requirements. A research work published in [49] proposes that it is possible to use Case Based Reasoning techniques and neural networks to improve the quality of requirements. The corresponding cases would include the analysis of quality performed by software specification requirements. Using neural networks, similar cases could be found for comparing the quality and thus provide a more accurate estimation of quality. However this work is only a proposal, as no real demonstration of the method is presented.

In the work [50], the authors proposed to use ambiguity detector tools over Software Requirements Specification (SRS) documents where the sentences and paragraphs were classified in function of their ambiguity. With the output of the tools and the values of the classification the machine learning techniques are used to make a decision-tree-based text classifier to detect ambiguities in the (SRS) documents. This work has similarity aspects to our work, but the difference is that it only detects ambiguous passages in SRS documents.

In the paper [51], the authors employ a statistical model using requirements quality metrics to predict system operational performance for government acquisition programs. The paper describes the expected results of a doctoral research to establish that the requirements quality is indeed a predictive factor for end system operational performance. The sources of data are Operational Requirements Documents and Operational Test Reports. In order to analyze the text of requirements, a tool is used to identify the linguistic quality problems. The results of the tool are a binary evaluation of the features: ambiguity, vagueness, testability and incompleteness. With this data the authors use a statistical software tool to make a logistic regression with the values of the features, determining a linear prediction model. The paper is a proposal and results are not published yet. This methodology has similarities with our work, it extracts quality metrics using tools and creating a model with them. The quality metrics employed in the paper are considered for us like features of the requirements and we use the quality metrics to evaluate these features. The goal of this paper is to show a statistically significant relationship between quality of requirements and system performance, in our work the objective is to learn a set of quality metrics in order to predict the quality that an expert would assign to requirements in order to emulate the decision of the expert.

The specification of natural language requirements with high quality is a complex issue which is closely linked to the needs of the project and expert opinion. The proposals presented to determine or to improve the quality of requirements automatically do not reflect the necessary flexibility to include information provided by the expert, tailoring it to each project. The evaluation of the features requirements is based on the interpretation of the authors. In this paper, we present a methodology that analyzes expert information by means of a set of classified requirements with good and bad quality. Machine learning techniques are employed over a set of metrics in order to estimate the quality of new requirements, predicting the classification that would be given by the expert.

The main contribution of our work is that the methodology learns and adapts the information provided by the expert, learning the characteristics that the expert considers when he decides the classification of the quality of the requirement and then estimate the quality of new requirements. This methodology is thus flexible to the quality demands in different projects.

2.4. Search Based Software Engineering

Search Based Software Engineering (SBSE) is an approach of Software Engineering in which Search Based Optimization is applied to Software Engineering [52].

In [53] the term SBSE was coined and the authors establish that software engineering is ideal for the application of metaheuristic search techniques, such as genetic algorithms, simulated annealing and machine learning, and among others.

SBSE have been used in many problems in Software Engineering. In [54] the authors address the problem of Multi-Objetive Next Release Problem (MONRP), in which they extend the simple function fitness over the Next Release Problem (NRP) to solve the Multi-Objetive problem. The (NRP) is an example of a Feature Subset Selection search problem directly related with SBSE.

To define problems of Software Engineering in SBSE it is necessary to choose a representation of the problem and define one or more fitness function. We can establish that our work is a methodology designed to find a fitness function that minimizes the difference between the quality estimated by the classifier and the quality provided by the expert over the classified requirement corpus. After Section 3 we present a formal definition of the problem, the representation and the function fitness.

3. Methodology

As mentioned above, the aim of the presented methodology is to determine the quality of a set of given requirements. Nevertheless, quality is an ambiguous concept that depends on the evaluation of different criteria; different users may evaluate differently the quality of the same requirement. The purpose of this method is to estimate the quality that a quality expert would give to a requirement. To accomplish this, the methodology must be based on the evaluation of the quality over the set of requirements. This evaluation should be carried out by the quality expert of the project. It is over this set that a learning process will take place, in order to identify the criteria that the expert would use to determine themselves the quality of the requirements.

The quality criteria are characterized by a set of metrics that provide a tool of management of requirements *Requirement Quality Analyzer* (RQA) [34] presented in the work [2]. These metrics

represent by numerical values the quality of each requirement. Based on these values, the methodology builds a model that determines the values of the metrics that would later assess requirements as of good or of poor quality.

The proposed methodology uses a representation based on the metric values of each requirement in the requirements corpus, taking into account the quality classification given by the expert. The machine learning algorithms employed can then identify the relationship between the metrics of each requirement with the classification provided by the expert. This will result into a classifier that classifies requirements as of good or bad quality, depending on the values of the metrics that represent them.

3.1. Metrics

In order to measure the quality of requirements different proposal are presented such as [19,55,56], that they not only present the desirable properties (Unambiguity, Validability, Understandability, and so on) but quantify metrics to provide the possibility to measure this desirable properties.

A requirement with good quality must be correct, consistent and complete. There are three main clusters of metrics to determine the quality of requirements: metric for correctness, metric for consistency and metric for completeness. In this work, we are focusing in the metric for correctness.

The Table 1 arrays the metrics that have been used in the proposed methodology for representing requirements. These metrics are defined in [2], where the authors present "a framework to obtain low-level quality indicators for measuring quality in textual requirements, as well as a tool that computes quality measures in a fully automated way". The work summaries of different approach a set of quantify indicators to provide measures of the quality of requirement, and the tool presented can calculate the metric that provide values of these indicators automatically. The tool calculates the metrics following the Guide for Writing Requirements prepared within [19] by INCOSE. The guide focuses on the textual requirements expression in the context of System Engineering, also how to express requirements clearly and precisely as text once they have been discovered. In our work, the metrics of the requirements are achieved using this tool referenced that they called RQA [34]. For each of the metrics, it is explained how it affects the requirement and the reference of the section in the paper [2] with which the metric is defined.

3.2. Elements of the methodology

The word Methodology is used frequently within researches, but usually the concept are not distinguished between methodology, process, method and technique; and moreover its meaning and differences [58]. Methodology can be defined as a collection of related processes, tasks, methods, and tools that to a class of problems that all have something in common [59].

Based on this definition, in Fig. 1 we show the elements that compose the methodology presented.

Following we detail each element of the methodology

- *Task 1 – Generating classifiers:* Classifiers are generated from a set of metrics associated with requirements previously classified by an expert in according to their quality.
- *Task 2– Estimating quality of new requirements and evaluation:* in order to estimate the quality of new requirements, it is necessary to create instances with the metrics associated and use the classifiers generated in task 1. In this task, it is possible to check the reliability of the classifiers performing evaluation tests with requirements which the quality classification is known.

The following sections detail each method of the tasks of the methodology.

3.2.1. Task 1 – generating classifiers

Developing this task, provides as a result a set of classifiers that can estimate the quality of new requirements. Fig. 2 illustrates the methods of this task of the methodology.

- *Quality assessment on a corpus of requirements:* to begin the methodology, it is necessary to have to assessment of the quality by an expert over a corpus of requirements. Thus, each requirement of the corpus will have an associated quality assessment. The objective of this method is to have as knowledge source the requirement wrote in natural language and the quality associated.
- *Extraction of quality metrics:* we should extract values automatically of quality metrics for each requirement of the corpus. From this method each requirement will be represented by the values of quality metrics and the estimated assessment of the quality by the expert. From this point it is possible to associate the set of metric that represent each requirement and the quality value provided by the expert.
- *Creation of learning instances:* with the values of the set of metrics and evaluation of the quality associated to the requirements, learning instances are built that will serve to generate classifiers. Learning instances give a concrete structure to the information that will be processed by the machine learning algorithms. In Section 4, we explain two different ways to create the learning instances. The way information is represented in the learning instances affects both the accuracy and the computation time spent in generating the classifiers, the quality assessment can solve different issues. The goal of this step is to determine how to use the information delivered in the previous method to achieve the best accuracy in the prediction of the quality.
- *Generation of classifiers:* employing machine learning techniques, we use the learning instances to build classifiers that are able to estimate the quality of new requirements. This method is the result of the first task and the objective is to generate a classifier that it can hold the knowledge about the values of metrics of requirement and relationship itself, with the quality.

3.2.2. Task 2 – Estimating quality of new requirements and evaluation

To estimate the quality of requirements using the classifiers generated in the previous task, we must build an instance for each new requirement with the quality metrics used in the first task. The format of this instance depends on the format of the learning instances.

Fig. 3 shows graphically the methods of this task of the methodology.

- *Extracting quality metrics:* in order to estimate the quality of a new requirement with the classifiers, we must extract the values of their quality metrics. Metrics must be the same as those used in the corpus of requirements for the generation of classifiers.
- *Generating instances of each new requirement:* extracting the metric of a new requirement the classifiers can be used to estimate its quality. To do this, with the metrics information, it is necessary to build appropriate instances for the classifiers so that they recognize the information in order to predict the assessment of the quality. The objective of this method is to represent the quality of the new requirement based on the quality metrics in the same way as the representation used in the first task.

Table 1
List of quality metrics.

No. of ambiguous expressions	Using ambiguous expressions may render the requirement difficult to understand. The expressions are defined in a list of ambiguous expressions. Based on the number of these expressions the methodology proposed makes the relation between these and the quality ([2] – Section: 4.2. Lexical indicators)
Boilerplates representation	Indicates whether the requirement is represented by boilerplates in the system, that means the requirement has a correct semantic and syntactic structure [57]. There is not reference of this metric in [2] but the latest versions of the tool RQA provide it to ensure the quality of the structure
No. of expressions in the conditional	The requirement should be written in assertive way. ([2] – Section: 4.3.1. Verbal tense and mood)
No. of connectors	The number of connectors on the requirement text. Using multiple connectors may indicate different needs in the same requirement and therefore compromise the atomicity ([2] – Section: 4.2.1. Connective terms)
No. of dependencies	Number of links in or out of the requirement. A large number of references can hinder the understanding of the requirement ([2] – Section: 4.4.3. Number of dependencies of a requirement)
No. of design expressions	The requirement should focus a necessity instead of expressing a solution ([2] – Section: 4.2.3. Design terms)
No. of domain concepts used	A large number of domain concepts used in the same requirement can indicate over-specification ([2] – Section: 4.3.2. Domain terms)
No. of domain verbs used	Too many domain verbs may indicate that the requirement expressed too many needs ([2] – Section: 4.3.2. Domain terms)
No. of flow expressions	The requirement should avoid pseudocode and flow expressions to avoid specifying the solution to the problem ([2] – Section: 4.2.1. Connective terms)
No. of expressions in the imperative	The requirement should have at least one imperative verb ([2] – Section: 4.3.1. Verbal tense and mood)
No. of implicit expressions	The requirement should be explicit, it should be avoided the use of pronouns ([2] – Section: 4.3.1. Verbal tense and mood)
No. of incompleteness expressions	Some sentences such as "etc.", "not limited to", "as a minimum"... clearly demonstrate that the requirement has not a clear scope. Incompleteness expressions must be avoided because the requirement should be atomic. The expressions are defined by a set of incompleteness expressions. This metric is incorporate in RQA tool based on the chapter 2.1.4 C4 – COMPLETE of the document Guide for Writing Requirements [19]
No. of In-links	Number of links that the requirement is object (in-links). A high number of links can hinder the understanding of the requirements ([2] – Section: 4.4.3. Number of dependencies of a requirement)
No. of negative expressions	Include one or more negative expressions can make the requirement is hard to understand ([2] – Section: 4.2.1. Connective terms)
No. of hierarchy levels	Too many levels of hierarchy in the requirements may lead to difficulties in understanding the requirement ([2] – Section: 4.4.2. Degree of nesting)
No. of character between punctuation	It is incorrect to write several sentences without using punctuation marks because it hinders the readability of the requirement ([2] – Section: 4.1.3. Punctuation)
No. of Out-links	Number of links to other requirements (out-links). A high number of links can hinder the understanding of the requirements ([2] – Section: 4.4.3. Number of dependencies of a requirement)
No. of verbs in the passive voice	The use of verbs in passive voice may hinder understanding of the requirement ([2] – Section: 4.3.1. Verbal tense and mood)
No. of rational expressions	Avoid using justification in the requirement. This metric is not explicit cited in [2] but this metric is explained in the Guide for Writing Requirements [19] and the RQA tool incorporates the metric
Readability requirement	A bad readability requirement may cause confusion. Expressed numerically by a mathematical formula described in ([2] – Section: 4.1.2. Readability)
No. of speculative expression	RQA tool uses this metric because the use of speculative expressions may indicate that the real need of the requirement is not clear. RQA evaluates this metric by a list of speculative expressions
No. of subjective expressions	Avoid the expression of the author's view in the requirement ([2] – Section: 4.2.2. Imprecise terms)
No. paragraphs	The requirement should not be expressed in too many paragraphs to avoid over-specification, redundancy of information and express various needs in the same requirement ([2] – Section: 4.1.1. Size)
No. of words	The requirement should not have an excessive number of words to also prevent the problems mentioned in the metric *number of paragraphs* ([2] – Section: 4.1.1. Size)

- *Estimating the quality of new requirements:* estimating the quality of a new requirement depends on the format of the learning instances used in the generation of classifiers. This method is the objective of the methodology: to estimate the correctness quality of textual requirements.
- *Evaluation of the classifier:* before generating the classifiers, we must extract a set of requirements that are not part of the training set, but rather form the test set. This set will be used to evaluate the classifiers and determine the success rate. To perform this process, the set of metrics that represent each requirement of the test set is processed by each of the classifiers, as explained in the previous method, in order to obtain the quality estimates. Since we already know the classification assigned by the expert, it is possible to collate this with the estimate and thus be able to determine the success rate of each classifier.

3.3. Tools

The tools used in the methodology are: RQA [34], for extracting the metrics for each requirement; and WEKA [60], that provides the machine learning algorithms.

3.4. Example

We now show an example of use of the methodology that involves the estimation of two new requirements.

To estimate the quality of the new requirements, it is assumed for the example that the classifier has been generated by the algorithm PART, which we call C-PART, with a corpus of requirements classified according to the quality provided by the expert.

The new requirements are:

- *Requirement 1:* The user shall be able to select an order form.
- *Requirement 2:* A minimum order value in Dollar can be defined in an order form.

The process starts with the extraction of the metrics for each requirement. Using the RQA tool [2], it is possible to extract the values of the different metrics for each requirement. Table 2 shows the metrics and values, emphasizing the metrics where the two requirements differ:

With the representation of requirements through the values of the quality metrics, we can generate instances that will be processed by the classifier C-PART. The classifiers that the induction

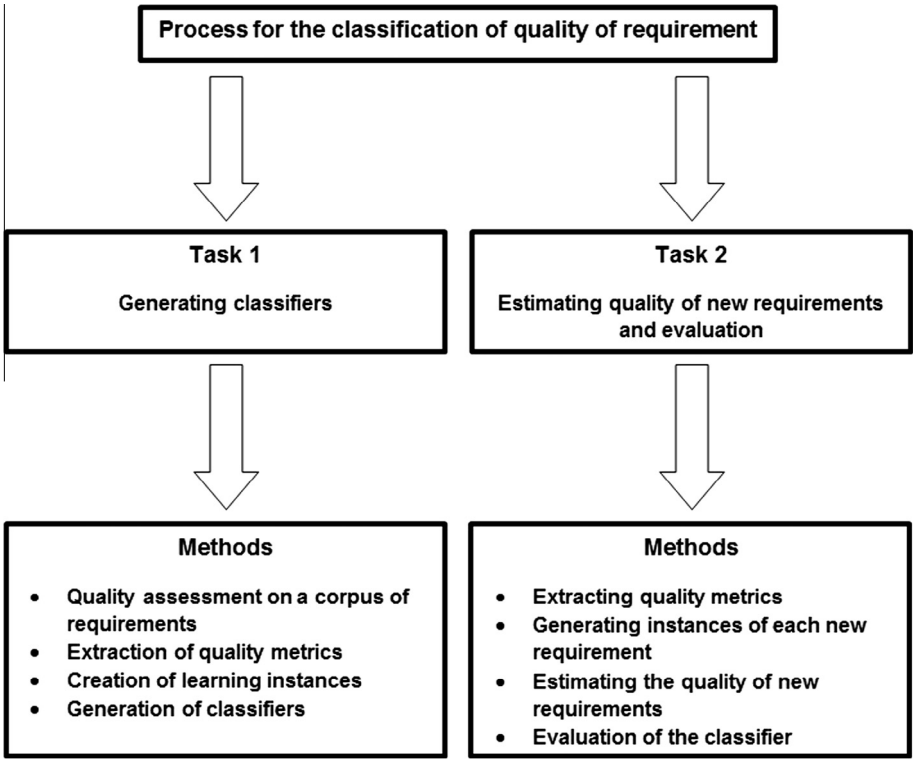

Fig. 1. Elements of the methodology.

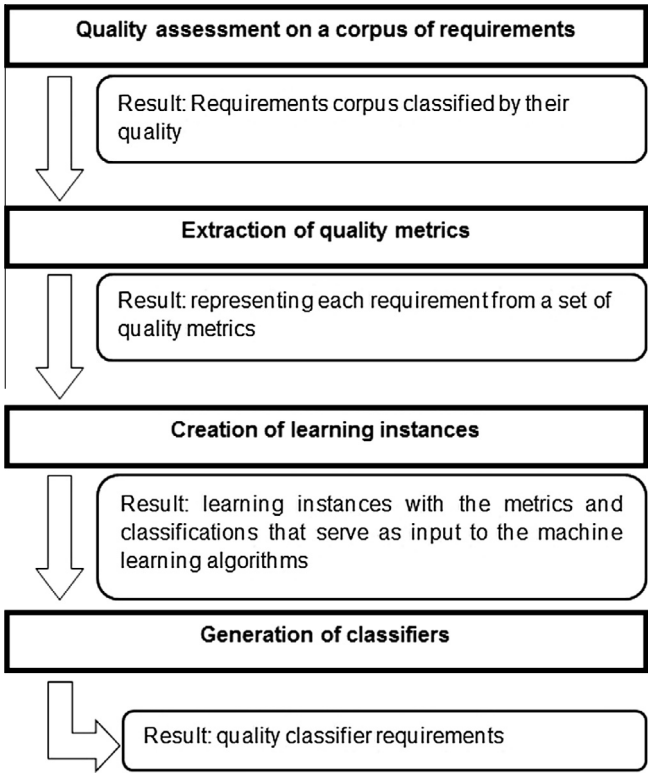

Fig. 2. Methods of the first task – Generating classifiers.

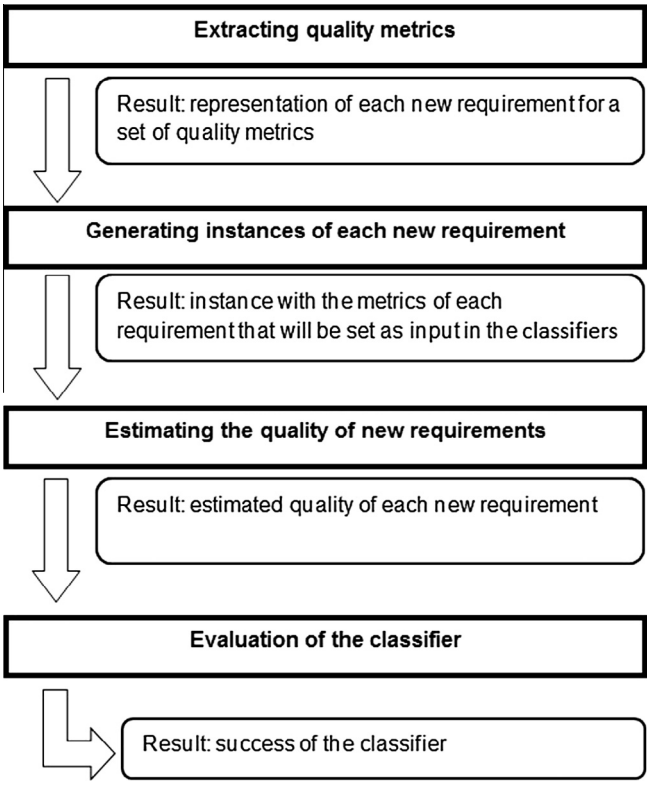

Fig. 3. Methods of the second task – Estimating quality of new requirements and evaluation.

rules algorithms produce are composed of a set of nested rules. If one rule is not met, then the next one is evaluated. Once one rule is met, it assigns the quality to the requirement.

When processing the instances of the two requirements cited in this example, a rule of the classifier is triggered based on the values of the metrics. This rule is the one that is fulfilled in the classifier.

Table 2
Values of the metrics of the example requirements.

Metrics	Requirement 1	Requirement 2
No. ambiguous expressions	**0.0**	**1.0**
Boilerplates representation	0.0	0.0
No. conditional expressions	**0.0**	**1.0**
No. connectors	0.0	0.0
No. dependencies	0.0	0.0
No. design expressions	0.0	0.0
No. domain concepts	1.0	1.0
No. domain verbs	**1.0**	**0.0**
No. flow expressions	0.0	0.0
No. imperative expressions	**1.0**	**0.0**
No. implicit expressions	0.0	0.0
No. incompleteness expressions	0.0	0.0
No. in-links	0.0	0.0
No. negative expressions	0.0	0.0
No. hierarchy levels	0.0	0.0
No. characters between punctuation	**46.0**	**63.0**
No. out-links	0.0	0.0
No. verbs in passive voice	**0.0**	**1.0**
No. rational expressions	0.0	0.0
Readability requirement	**3.0**	**8.0**
No. speculative expressions	**0.0**	**1.0**
No. subjective expressions	0.0	0.0
No. number paragraphs	1.0	1.0
No. number words	**10.0**	**13.0**
No. requirement changes	0.0	0.0

The bold values emphasize those metrics where the two requirements differ.

- For requirement 1, the following rule is triggered:
  Domain concepts > 1 AND
  Readability requirement <= 12 AND
  Imperative expressions <= 1 AND
  Verbs in passive voice <= 0 AND
  Domain verbs > 0 **THEN** Good quality
- For requirement 2:
  Verbs in passive voice <= 1 AND
  Imperative expressions <= 0 AND
  Characters between punctuation > 26 AND
  Speculative expressions > **THEN** Bad quality

Thus, according to the proposed methodology, the estimation of the quality provided by the expert who provided the corpus, would be: good for the first requirement and bad for the second.

3.5. Formal definition of the problem based on SBSE

Now, we formalize the problem as it is completed by the SBSE formalization, where a definition, a representation and the fitness function is represented.

Given:

- A set of textual requirements $R = \{r_1, \ldots, r_n\}$.
- A set of correctness metrics apply to requirements $M = \{m_1, \ldots, m_k\}$ such that $m_j : R \to \mathbb{R}(1 \leq j \leq k)$.
- The set of correctness quality assessments provided by the expert over the set of requirements $C = \{C_1, \ldots, C_n\}$, such that $C_i \in \{0, 1\}$ is the quality provided by the expert to the requirement $r_i(1 \leq i \leq n)$ and 0 represent bad quality and 1 represent good quality.

Problem:

To find a function $f : \mathbb{R}^k \to \{0, 1\}$ such that minimized $\sum_{i=1}^{n} |f(m_1(r_i), \ldots, m_k(r_i)) - C_i|$.

Representation:

The representation of each possible solution is a piecewise function. This function can be defined like a set of rules.

Fitness function:

The same summatory expression that appears in the definition problem can be the fitness function that should be minimized.

4. Implementation

In this section, we explain the representation and implementation details of the methodology methods proposed in this work. We have followed two approaches in order to present two different ways of representation depends on balance of the different kinds of requirement. In Section 5.4 we explain the correct selection of the representation in function of the balance. The two representation approaches are given by the different implementations of the learning instances. These two approaches focus on different priorities: building time of the classifier and accuracy rate. In the following sections, we present the implementation of these two approaches; each contains the implementation of the methods of the methodology. In both, there are methods that necessarily share the same implementation, such as the construction of the corpus of requirements or the extraction of quality metrics.

4.1. Approach 1: Simple model

4.1.1. Task 1 – generating classifiers

4.1.1.1. Quality assessment on a corpus of requirements. A prerequisite for the initiation of the methodology methods, is a corpus of requirements classified previously in function of their quality by an expert. In this work, we use a corpus of requirements supplied by the Requirements Working Group of INCOSE (International Council on Systems Engineering) [30]. This corpus contains 1035 requirements and each has been classified as of good or of bad quality by an expert. In the corpus there are 545 requirements with good quality and 490 with bad quality. This corpus is the initial basis for both solution approaches presented in this work.

4.1.1.2. Extraction of quality metrics. As we said previously, each requirement of the corpus has been processed by the tool RQA presented in [2] and then to use the representation of each requirement how a set of values metrics.

4.1.1.3. Creation of learning instances. Learning instances are stored in a format that is compatible to WEKA [60], a popular data analysis tool. Each learning instance consists of a set of attributes and a set of instances. The set of attributes contains the values of the quality metrics, as well as a class value which represents the two quality options (good quality and bad quality). The set of instances consists of the numerical values of the set of metrics of learning requirements and the quality values assigned by the expert.

The way metric values are combined to create the learning instances, determine different forms of training the algorithms, which results in the two solution approaches presented in this work.

The first approach is called "simple model" because each instance corresponds to the set of metric values of only a requirement, as well as the quality assigned by the expert (see Fig. 4). Thus, the number of instances that contain the simple model is the number of requirements that form the training set.

4.1.1.4. Generation of classifiers. To generate the classifiers that estimate the quality of requirements, we use machine learning algorithms, specifically the rule induction algorithms PART and C 4.5. With the intention of improving the accuracy, we also use combination of homogeneous classifiers using Bagging and Boosting techniques, based on the above algorithms.

The algorithms receive as cases learning input the instances created with the values of the metrics of the requirements. Based on these instances, the algorithms induce rules to determine the class values that represent quality values.

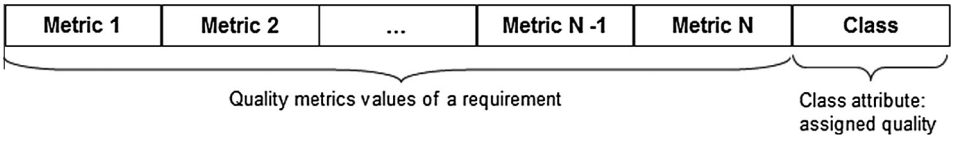

Fig. 4. Format of an instance in the simple model.

The approaches proposed in this work provide sets of rules that differ in their interpretation of the solution of the problem. In one approach, simple model, the rules determine the quality of a requirement in terms of the values that are the metrics representing it, i.e. the algorithm learns the value range and combination thereof for estimating the quality of a requirement. In approach 2, which will be explained in detail in Section 4.2, the rules determine an evaluation of comparing two requirements.

4.1.2. Task 2 – estimating quality of new requirements and evaluation

To use the classifiers for estimation of the quality of new requirements, it is necessary to generate instances with their corresponding metric values and using them as input cases in the classifiers.

4.1.2.1. Extracting quality metrics. In the implementation of this method of the methodology, the new requirements are processed by the RQA tool [2] used in the previous task.

4.1.2.2. Generating instances values for each new requirement. We then proceed to the generation of instances with the set of metrics that represent them. The format of the instances must be appropriate to the form of learning instances used in the construction of the classifier. The implementation of this method is different for both solution approaches used in this work.

To use the classifiers generated in approach 1, the format of the instances must contain the values of the quality metrics of the new requirements in the same order as in the learning instances. Based on these values, the classifier can estimate a quality value.

4.1.2.3. Estimating the quality of new requirements. The estimated quality by the classifiers using the simple model is now straight-forward. The classifier outputs a class value to the set of metrics of the new requirement; this value is the quality that the classifier predicts as the most probable expert estimation.

4.1.2.4. Evaluation of the classifier. In our effort to evaluate the classifiers generated from different algorithms, we use test sets which are not part of the learning set used to train the algorithms. We use test sets because we perform cross validation to evaluate the degree of accuracy of the algorithms. In Section 5, we explain the details regarding the tests used in the cross validation.

Each test set contains a stratified selection of 103 requirements, 54 of which have good quality and 49 bad quality. Therefore each training set contains 932 requirements, where 491 requirements are classified as of good quality and 441 as of bad quality.

4.2. Approach 2: Model with pairs of comparison

In this section, we detail the implementation of the second solution approach of this work. This approach is called "model with pairs of comparison" because the learning instances are the comparison of two requirements. As mentioned above, this approach has several methods in common with the first approach, which have been already covered previously.

4.2.1. Task 1 – generation of classifiers

4.2.1.1. Quality assessment on a corpus of requirements. Both the corpus of the requirements and the quality assigned by the expert are the same as in the first approach. It is necessary to have the same initial corpus to make a coherent comparison of the results of the experiments. As it has already been mentioned, the corpus contains 1035 requirements, 545 of which are of good quality and 490 of bad quality.

4.2.1.2. Extracting quality metrics. The representation of each requirement for the set of quality metrics is common in both approaches.

4.2.1.3. Creating instances of learning. In this method, the two approaches proposed are distinguished with respect to the format of learning instances. This solution approach of the problem is a requirements engineering adaptation to the methodology proposed in the doctoral thesis [61], which presents "a systematic method which can predict the position that would occupy a resource, between others sorted considered like competition, if its characteristics were changed". The authors developed a methodology to estimate documentary relevance assigned by the searchers in the WEB environment. The authors use machine learning algorithms over learning instances composed by pairs of comparison of SEO (search engine optimization) values in WEB searchers. In our study, the values that compose the instances are the quality metrics of the requirements, each instances is a comparison of two set of metrics belonging to two different requirements, plus a class value indicating which of the two sets of metrics has better quality. In Fig. 5, we show a graphic representation of an instance.

The value of the class attribute that we have established in the implementation is 1 if the first requirement has better quality assigned by the expert than the second, or 2 if the second requirement has better quality than the first one.

For each instance of pairs of comparison, we perform permutations on the order of the requirements, and consequently change the class value. For example, if the quality of a requirement A is better than the quality of a requirement B, the instances have the permutations depicted in Figs. 6 and 7.

The reason to perform this permutation is that the algorithms can clearly identify if a set of metrics of a requirement is better than another one, regardless of their position within the instance. The number of learning instances with pairs of comparison is the number of all possible pairs of requirements with different quality multiplied by two because of the permutations.

4.2.1.4. Generating the classifiers. We use the same machine learning algorithms to generate the classifiers as in the first approach, namely PART and C4.5, as well as a combination of homogeneous classifiers using Bagging and Boosting techniques, which are based upon the above.

In this approach, the algorithm learns which set of metrics corresponding to each requirement is decisive for the classification of requirements based on their quality. The result of the quality is not obtained directly as the output of the classifier; for this reason, it is

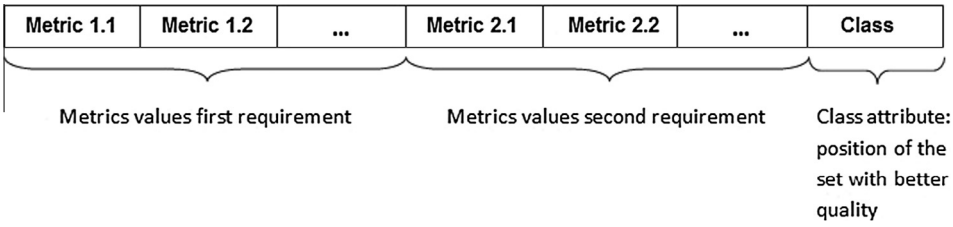

Fig. 5. Format of an instance in the model with pairs of comparison.

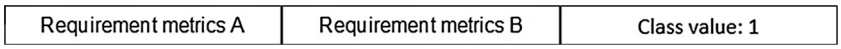

Fig. 6. Instance without permutation.

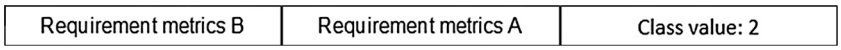

Fig. 7. Instance permuted.

necessary to interpret the results in order to determine the quality of a new requirement.

4.2.2. Task 2 – estimating the quality of new requirements and evaluation

4.2.2.1. Extracting quality metrics. The first method of this task is common in the two approaches proposed. It is about representing the new requirements with the same quality metrics that were used in the training set.

4.2.2.2. Generating instance for each new requirement. Since the output of the classifier is not the quality of the new requirement, it is necessary to create a method to estimate the quality of a new requirement by comparing requirements.

The method we have chosen is to compare each new requirement with the requirements used in the training set when the classifier was generated. Therefore, the format of each instance is the result of the comparison between the new requirement and a requirement of the training set. As in the case of training, for each instance, permutation is performed, so that the number of instances required for estimate the quality of a new requirement is the number of requirements of the training set, doubled because of the permutations. This set of instances is called evaluation instances.

The result of introducing the set of evaluation instances in the classifiers is that, for each instance, it is estimated if the new requirement or the learning on is of better quality. In Fig. 8, we show graphically the composition of the set of evaluation instances.

In Fig. 8, the requirements characterized with letters of the alphabet representing the requirements of the training set. Note that permutations are performed for all instances.

After entering the set of evaluation instances in the classifier, we obtain as result an assessment for each instances indicating which requirements have better quality: the new requirement or the requirement with which it is compared. In this work, we have created a set of rules that use this information to establish the quality of new requirements.

The set of rules uses as variables the percentage of requirements with good quality of the training set that were measured with better quality than the new requirement (called Percent_Good_Requirement); and vice versa, the percentage of requirements with poor quality that were measured with worse quality than the new requirement (called Percent_Bad_Requirement). These variables are used because the algorithms are trained by comparing requirements with different quality, so that the rate of success is higher.

Now in Fig. 9, we present the set of rules that we have established to determine the quality of a new requirement and after that we explain how this set has been performed.

Where:

- *Percent_Good_Requirement:* represents the number of requirements with good quality of the training set valued by the classifier with better quality than the new requirement.
- *Percent_Bad_Requirement:* represents the number of requirements with bad quality of the training set valued by the classifier with worse quality than the new requirement.
- *Estimated_Quality:* Quality assigned to the new requirement.

In order to define these rules, we have used the classifiers with the same requirements of the training set with which the algorithms have been trained. The learning instances to introduce in the classifiers are formed by comparing each requirements of the training set with all others. The result is an assessment of the quality of each requirement with all others. By gathering this information, it is possible to obtain for each requirement of the training set, the percentage of requirements with good quality that are assessed as of worse quality than the requirement of the training set treated; and vice versa, the percentage of requirement with bad quality that are assessed the requirement treated with better quality. Once done, it is possible to generate a set of leaning instances using for each training requirement the above mentioned percentage, as well as the quality assessment assigned by the expert. In Fig. 10 we show graphically an instance of one requirement treated:

The learning instances have been processed by the Bagging PART algorithm, resulting into an automatically generated classifier, that associate the quality estimated by the expert with the percentages of the requirements of the training set that assess a requirement with good or bad quality. The classifier is composed by the set of nested rules shown in Fig. 9. With this process we achieve to use the information of the training set to generate rules to estimate an assessment of the quality of new requirement in the second approach.

4.2.2.3. Evaluation of the classifier. The test sets in the two solution approaches must be the same in order to correctly compare the results. Recall that each test set contains 103 requirements, 54 of which have good quality and 49 bad quality.

New requirement metrics	Metrics of requirement **A**	Class Value: To be determined
Metrics of requirement **A**	***New requirement metrics***	Class Value: To be determined
New requirement metrics	Metrics of requirement **B**	Class Value: To be determined
Metrics of requirement **B**	***New requirement metrics***	Class Value: To be determined
. . .	. . .	. . .
New requirement metrics	Metrics of requirement **Z**	Class Value: To be determined
Metrics of requirement **Z**	***New requirement metrics***	Class Value: To be determined

Fig. 8. Set of evaluation instances.

```
If Percent_Good_Requirement = 100% And Percent_Bad_Requirement = 100%
    Estimated_Quality = Good_Quality
Else
    If Percent_Good_Requirement = 100%
        Estimated_Quality = Bad_Quality
    Else
        If Percent_Bad_Requirement = 100%
            Estimated_Quality = Good_Quality
        Else
            If Percent_Good_Requirement >= 99%
                Estimated_Quality = Bad_Quality
            Else
                If Percent_Bad_Requirement >= 99%
                    Estimated_Quality = Good_Quality
                Else
                    If Percent_Good_Requirement > 91.65% AndPercent_Bad_Requirement > 90.97%
                        Estimated_Quality = Bad_Quality
                    Else
                        If Percent_Bad_Requirement > 96.61%
                            Estimated_Quality = Good_Quality
                        Else
                            If Percent_Good_Requirement > 88.93%
                                Estimated_Quality = Bad_Quality
                            Else
                                If Percent_Bad_Requirement > 49.89%
                                    Estimated_Quality = Good_Quality
                                Else
                                    Estimated_Quality = Bad_Quality
```

Fig. 9. Set of rules to determine the quality of a new requirement.

Percentage of good quality of requirements that have evaluated the training requirement treated with worse quality	Percentage of bad quality of requirements that have evaluated the training requirement treated with better quality	Class value assigned by the expert to the training requirement treated

Fig. 10. Instance to generate the set of rules that evaluate the new requirements in the second approach.

5. Results

In this section, we cover the research questions and the related answers by the use of the methodology through the use case and we explain the results obtained for the two approaches presented in this paper.

5.1. Research questions

The principal research questions that the work pretend to answer are:

- It is possible to learn the features of the correctness quality of requirement through the processing corpus of requirement classified in function of the quality.
- Learning the features of the correctness quality is possible to predict the quality of others requirements.

To address the research questions, the methodology presented represents the features by a set of correctness quality metrics for each requirement and use machine learning techniques to learn the relation and values of the metrics in order to predict the quality of others requirements. To demonstrate the reliability of the methodology we present the result obtained by the use case using different algorithms in order to compare different ways to improve the fitness function.

5.2. Results

There are two key factors to evaluate the different algorithms: the accuracy percentage of the classifiers obtained on the test set and the time spent in generating classifiers.

The evaluation of these two factors is performed on each classifier generated by the different learning algorithms used in this work.

The evaluation of the classifiers is performed with the accuracy over a test set of requirement where the classifications are known. For a proper comparison of these approaches, we have obtained the results by cross-validation. Due to the fact that estimation of the quality of requirements in the approach 2 (model with pairs of comparison) is not the direct output of the classifiers, it has not been possible to use cross-validation, as provided by WEKA [60]. It was therefore necessary to manually create cross-validation sets and obtain the accuracy results per set. In order to determine the final accuracy of each learning algorithm, we have calculated the average of the results of all sets that form the cross-validation sets.

Although WEKA can perform cross-validation directly in approach 1 (simple model), we have decided to use the same sets employed in the approach 2 and to make the cross-validation manually in order to get a proper comparison of the results.

For cross-validation, we have created 10 sets stratified of the corpus of requirements. The algorithms are trained with training sets consisting of 9 of the 10 sets and the tenth set is used as a test set. This is done 10 times by selecting a different test set each time so that the algorithms will both learn and be evaluated against all requirements. The final result of cross-validation is the arithmetic average of the success rates of the different test sets.

5.2.1. Results of approach 1 (Simple model)

The accuracy results of the classifiers generated by the machine learning algorithms in the different cross-validation sets are shown in Table 3. This table contains the accuracy above 10 test sets requirements of the classifiers generated by machine learning algorithms in each of the sets which form the stratification for cross-validation.

It is noted in the table that the best accuracy is achieved by the algorithms Bagging PART and Boosting PART with 86.83% accuracy and with a median of 87%. The algorithm with the results closer to the median is Boosting PART with a standard deviation of 2.90, which means that this algorithm is the most stable.

In Table 4, we show the efficiency of these algorithms in the learning process. We study the efficiency because it is possible to have the necessity to make a classifier for each project and/or to include new constraints in the new requirements. The values in the table were calculated by the average of the time in seconds used by the algorithms for generating the classifiers.

It can be noted that the algorithms C4.5 and PART require less time in the generation of a classifier than the combination of the homogeneous classifiers Bagging and Boosting.

5.2.2. Results of approach 2 (model with pairs of comparison)

Table 5 presents the results obtained using the classifiers generated by different algorithms over sets created to perform cross-validation.

The best result is obtained with the C4.5 algorithm with 87.72% accuracy. However C4.5 algorithm obtains the highest standard deviation 7.12, rendering it the most unstable of the algorithms tested. Bagging-PART algorithm achieves the second best accuracy 87.02, the highest value with respect to the median 87.12, and it can be considered the most stable algorithm because it obtains the lowest result in standard deviation 2.44.

In Table 6, we show the efficiency of the algorithms for generating the classifiers, presented in seconds, minutes and hours.

It can be observed that the most efficient algorithm is C4.5 somewhat less than 5 min. The algorithms Bagging-PART and Boosting-PART takes several hours to complete the process.

The algorithms have been executed on a computer with an Intel microprocessor Core i3-3225 to 3.30 GHz and a RAM capacity of 8 GB.

In Fig. 11, we present a comparative graph of efficiency of the algorithms.

5.3. Comparison of results of the approaches 1 and 2

This section compares the results of the two approaches considered in this work.

Fig. 12 shows in a bar graph the results. For each algorithm, it shows the success rates of the two approaches thereby facilitating comparisons.

Fig. 13 shows the same results as Fig. 12 but with a radial plot. This type of diagram allows better analysis for the comparison of the approaches.

Table 3
Accuracy results for the classifiers with the simple model.

Test sets	PART	C4.5	Bagging PART	Bagging C4.5	Boosting PART	Boosting C4.5
Set 1	78.21	78.21	80.19	81.18	81.18	87.15
Set 2	83.16	85.14	87.12	90.09	87.12	86.13
Set 3	86.13	83.16	92.07	86.13	85.14	88.11
Set 4	90.09	89.10	91.08	90.09	92.07	88.11
Set 5	86.13	77.22	86.13	85.65	87.12	82.17
Set 6	83.16	86.13	85.14	86.13	85.14	86.13
Set 7	82.17	79.20	79.20	80.19	83.16	80.19
Set 8	88.11	84.15	90.0	87.12	90.09	85.14
Set 9	84.26	83.16	88.11	82.17	88.11	86.13
Set 10	86.13	86.13	89.10	84.15	89.10	90.09
Total (average)	84.76	83.16	86.83	85.29	86.83	85.94
Median	85.19	83.66	87.62	85.89	87.12	86.13
Standard deviation	3.33	3.84	4.32	3.41	3.26	2.90

Table 4
Average time in seconds used in the generation of the classifiers by the different algorithms with simple model.

	PART	C4.5	Bagging PART	Bagging C4.5	Boosting PART	Boosting C4.5
Time in seconds	1.56	1.44	5.26	5.10	4.99	4.82

Fig. 14 shows in a bar graph the time used in seconds by the algorithms of the two approaches in the generation of the classifiers. The chart appears in 3D perspective to distinguish the values obtained with the simple model.

5.4. Discussion

Observing the results, we have to emphasize that the model that achieves the best success rate is the model with pairs of comparison, with an improvement in four of the six algorithms. Although the C4.5 algorithm in this model gets the best success rate 87.72, it also has the highest standard deviation 7.12. The algorithm Bagging-PART of this second approach obtains the lowest standard deviation value of the results presented 2.44, this being therefore the most stable algorithm, and it has the second highest success rate of all results 87.02.

But even though the model with pairs of comparison achieves the best success rates, it does not achieve a significantly higher difference in most of the algorithms, even though it considerably increases the number of instances (435,026 instances in the model with pairs of comparison versus 934 in the simple model). This observation can be attributed to the fact that the instances of the model with pairs of comparison are formed by comparing the requirements with different quality, and in the evaluation process the new requirements are compared with all requirements belonging to the training set, causing comparing requirements with the same quality. This can cause wrong estimates that provoke a

Table 5
Accuracy results for the classifiers with the model with pairs of comparison.

Test sets	PART	C4.5	Bagging PART	Bagging C4.5	Boosting PART	Boosting C4.5
Set 1	80.19	81.18	85.14	84.15	79.20	81.18
Set 2	90.09	88.11	88.11	87.12	92.07	85.14
Set 3	85.14	84.15	88.11	86.13	86.13	85.14
Set 4	89.10	100	89.10	88.11	90.09	93.06
Set 5	84.15	88.11	86.13	87.12	86.13	86.13
Set 6	81.18	85.14	86.13	85.14	86.13	86.13
Set 7	82.17	79.20	83.16	78.21	85.14	78.21
Set 8	83.16	88.11	90.09	88.11	89.10	90.09
Set 9	82.17	83.16	84.15	88.11	85.14	82.17
Set 10	84.15	100	90.09	85.14	86.13	87.12
Total (average)	84.15	87.72	87.02	85.74	86.53	85.44
Median	83.66	86.63	87.12	86.63	86.13	85.64
Standard deviation	3.23	7.12	2.44	2.99	3.46	4.27

Table 6
Average time used in the generation of the classifiers by the different algorithms with the model with pairs of comparison.

	PART	C4.5	Bagging PART	Bagging C4.5	Boosting PART	Boosting C4.5
Time in seconds	1676.7	275.7	13094.6	3120.6	21985.5	4756.5
Time in minutes	27.945	4.595	218.235	52.01	366.425	79.275
Time in hours	0.465	0.0765	3.637	0.866	6.107	1.32

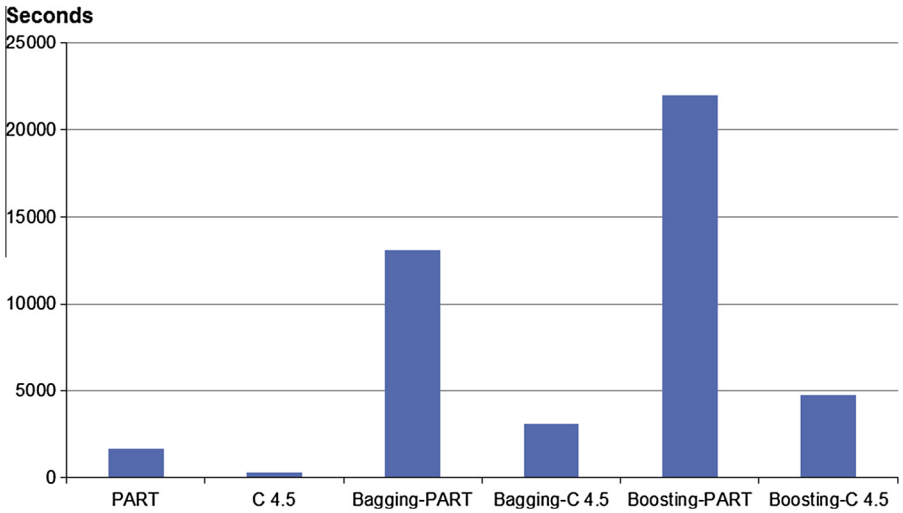

Fig. 11. Efficiency of the algorithms in seconds with the model with pairs of comparison.

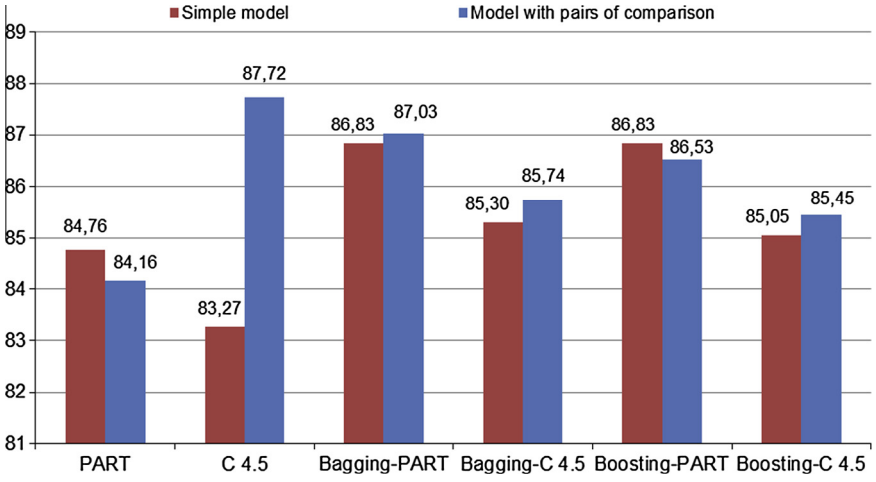

Fig. 12. Accuracy of the classifiers in the approach one and two.

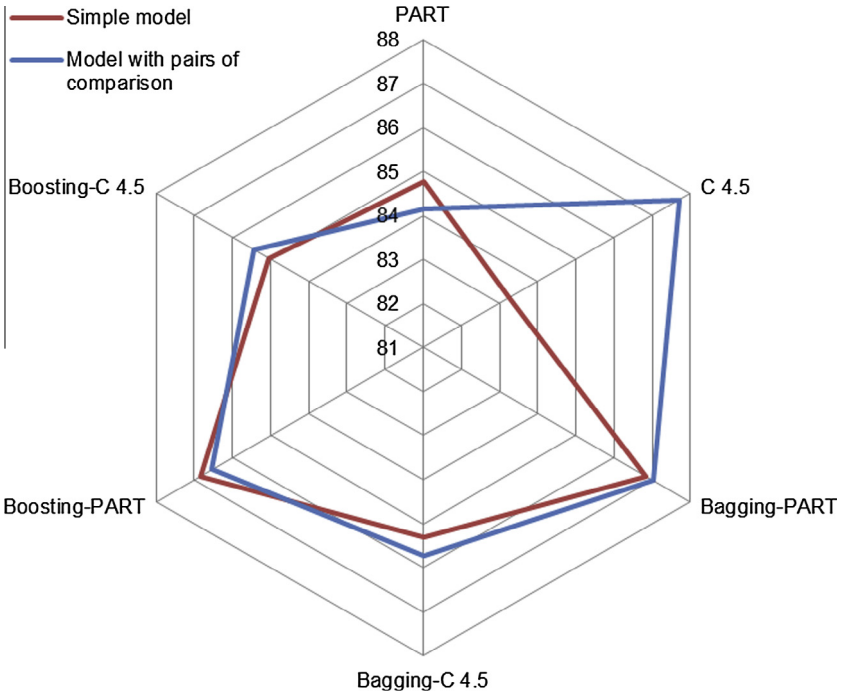

Fig. 13. Accuracy of the classifiers in the approach one and two.

decrease in the final accuracy of the experiments. The reason to perform the second approach is to show a different way to implement the methodology and to open new ways of research to improve the accuracy.

The incorrect categorization of the requirements by the classifier can be due to the fact that the metric values of these requirements are the same with others, having only different categorization as provided by the expert, since the expert could have considered another classification based on aspects that the metrics cannot measure. Another case for incorrect categorization is that the representation of the metrics of these requirements is not present in the training set.

The choice of implementation will not only be based on the accuracy and efficiency obtained in the experiments, it is also influenced by the balance of the different kinds of requirements:

- If the number of requirements with different classes is balanced, we can use the simple model which gives good measurements of accuracy and high efficiency.
- If the number of requirements with different classes is unbalanced, we must implement the model with pairs of comparison since it is achieved a large number of instances by comparing each requirement of a kind with all requirements of the opposite kind. Due to the large number of instances, the algorithms are able to generate classifiers that learn to distinguish the two classes even with the imbalance in training instances.

5.5. Answering the research questions

Both research questions established before are answered at this point. Once the methods that complete the first task of the process

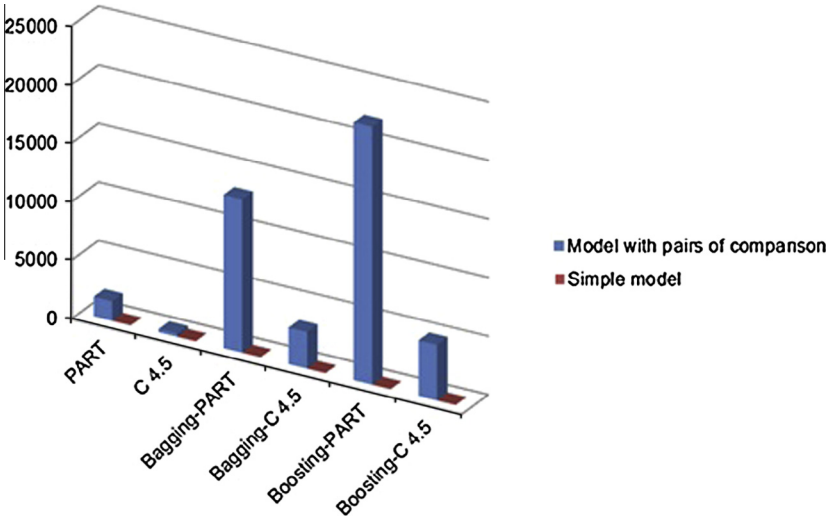

Fig. 14. Efficiency of the algorithms in seconds of the two approaches.

and then processed by the machine learning techniques are completed, a classifier is generated by the representation of the requirement using a set of correctness quality metrics. This classifier is composed by a set of linked rules that represent the learning of the features of correctness quality of requirements.

Once the second task is complete, the quality of new requirements is estimated obtaining in our experiments an accuracy in a range of 83.27 as a minimum accuracy and 87.72 as the maximum percentage of accuracy.

5.6. Threats in the validity

The principal threat in the validation of the methodology is the errors in the classification of the requirements by the experts, because the classifier can learn incorrect classification and replicate the error in the classification of new requirements. Trying to avoid this threat, we use induction rules algorithms that they are robust against noise, caused by insufficient data or errors in both attributes and in the allocation class value [6].

Another threat is the quality provided by the expert not focused on correctness but completeness or consistency. The set of metrics presented in this work is about correctness and with the result only probe the accuracy using this kind of metrics. To probe that the methodology is also works with another kind of metric is out of the scope of the paper but it is a good topic for future works.

6. Conclusions

In this work we have presented a methodology that evaluates the quality of requirements written in natural language. The method aims to generate classifiers using machine learning algorithms of induction rules only providing requirement sets classified by an expert according to their quality, in order to estimate the quality of new requirements. With this, the exigency and constraint in the quality of the futures requirement are established by the experts of the projects. We have proposed the automatic extraction of quality metrics to represent the quality of the requirements written in natural language, and these sets have been used by the algorithms to induce rules, also in an automatic way, for evaluating the quality of a requirement according to the metric that represents it. We have presented two solution approaches of problems, differentiated by the form of the learning instances used for the generation of classifiers obtaining different results, both in accuracy and efficiency of the algorithms. Finally we have demonstrated the reliability of the methodology examining a case study with a set of real requirement.

7. Future work

The accuracy of the classifiers can be improved by varying the learning instances or devising new forms of implementing them. A modification that could be performed in the approach 2 is to adapt more faithfully to the referenced work [61], where the classifiers are generated from the sorted resource models. The proposed changes are to use the classifiers to evaluate the same requirements that the algorithms have used in the training. All requirements will be compared with each other and it will be possible to establish a quality measure for each requirement, with respect to the group and then obtain ordered requirements, based on their quality. With the training set ordered, it is possible to generate new learning instances comparing each requirement with those that have a lower value in the set and then establish that this requirement is better than everyone it has been compared. Using machine learning algorithms, it is possible to generate a classifier that estimates the position of a new requirement and thus to establish a quality assessment based on that position in the set. The number of instances of the model would be all possible pairs of requirements of the training set, not only couples of different quality of requirements as in the presented work, so that the time spent in generating the classifier increases significantly.

In order to improve the efficiency in the generation of classifiers in the approach 2, it is possible to create a training set composed of only a selection of requirements that represent the entire corpus. This process can be performed choosing randomly two requirements with different quality and generating a classifier with them. Once the classifier is generated, it is possible to introduce the remaining requirements to estimate the quality. Those requirements that have their quality predicted accurately are removed from the set, and the process is repeated by adding another pair of requirements to generate a new classifier. This process is repeated until all requirements in the provided set are correctly classified, with the final result of a reduced set of requirements that will form the training set. In this manner it is possible to reduce the time of generating the classifiers.

An interesting future work is to probe that our methodology works with another kind of metrics like completeness and consistency and then cover completely the assessment of the quality of requirement written in natural language.

Acknowledgements

The research leading to these results has received funding from the European Union's seventh framework program (FP7/2007–2013) for CRYSTAL – CRITICAL SYSTEM ENGINEERING ACCELERATION joint undertaking under Grant Agreement No. 332830 and from specific national programs and/or funding authorities.

References

[1] P. Loucopoulos, V. Karakostas, System Requirements Engineering, McGraw-Hill, Inc., 1995.
[2] G. Génova, J.M. Fuentes, J. Llorens, O. Hurtado, V. Moreno, A framework to measure and improve the quality of textual requirements, Requir. Eng. 18 (2013) 25–41, http://dx.doi.org/10.1007/s00766-011-0134-z.
[3] F. Fabbrini, M. Fusani, S. Gnesi, G. Lami, The linguistic approach to the natural language requirements quality: benefit of the use of an automatic tool, in: Proceedings of the 26th Annual NASA Goddard Software Engineering Workshop 2001, 2001, pp. 95–105, http://dx.doi.org/10.1109/SEW.2001.992662.
[4] F. Fabbrini, M. Fusani, S. Gnesi, G. Lami, An automatic quality evaluation for natural language requirements, in: Proceedings of the Seventh International Workshop on RE Foundation for Software Quality REFSQ 2001, vol. 1, 2001, pp. 1–15 <http://fmt.isti.cnr.it/WEBPAPER/P11RESFQ01.pdf>.
[5] J. Hong, I. Mozetic, R.S. Michalski, AQ15: Incremental Learning of Attribute-Based Descriptions from Examples: The Method and User's Guide, Dep. Comput. Sci. Univ., Illinois Urbana-Champaign, 1986.
[6] J. Major, J. Mangano, Selecting among rules induced from a hurricane database, J. Intell. Inf. Syst. 4 (1995) 39–52, http://dx.doi.org/10.1007/BF00962821.
[7] J.R. Quinlan, C4.5: Programs for Machine Learning, Elsevier, 1993.
[8] E. Frank, I.H. Witten, Generating accurate rule sets without global optimization, Work. Pap., 1998 <http://hdl.handle.net/10289/1047>.
[9] W.W. Cohen, Fast effective rule induction, in: Proceedings of the Twelfth International Conference on Machine Learning, 1995, pp. 115–123.
[10] T.G. Ditterrich, Machine learning research: four current direction, Artif. Intell. Magaz. (1997) 97–136.
[11] F.P. Brooks, No silver bullet: essence and accidents of software engineering, Computer (Long. Beach., CA) 20 (1987) 10–19, http://dx.doi.org/10.1109/MC.1987.1663532.
[12] E.J. Braude, Software Engineering: An Object-Oriented Perspective, John Wiley & Sons, Inc., 2000.
[13] R.L. Glass, Facts and Fallacies of Software Engineering, Addison-Wesley Professional, Boston, 2003.
[14] P. Bourque, R. Dupuis, Guide to the Software Engineering Body of Knowledge 2004 Version, 2004, http://dx.doi.org/10.1109/SESS.1999.767664.
[15] T.S. Group, Chaos Manifesto 2013, 2013 <http://www.standishgroup.com/> (last accessed 20.05.05).
[16] T. Gorschek, C. Wohlin, Requirements abstraction model, Requir. Eng. 11 (2006) 79–101, http://dx.doi.org/10.1007/s00766-005-0020-7.
[17] Z. Racheva, M. Daneva, A. Herrmann, A conceptual model of client-driven agile requirements prioritization: results of a case study, in: Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering Measurements, vol. 39, 2010, pp. 39:1–39:4, http://dx.doi.org/10.1145/1852786.1852837.
[18] F. Alencar, G. Giachetti, O. Pastor, From i∗ requirements models to conceptual models of a model driven development process, in: Proceedings of the Second IFIP WG 8.1 Working Conference on PoEM 2009 on the Practice of Enterprise Modeling, Stock., Sweden, November 18–19, 2009, 2009, pp. 99–114, http://dx.doi.org/10.1007/978-3-642-05352-8_9.
[19] Requirements Working Group, International Council on Systems Engineering (INCOSE), Guide for Writing Requirements, 2012, 59 <http://www.incose.org>.
[20] S. Magee, L.L. Tripp, Guide to Software Engineering Standards and Specifications, Artech House, Inc., 1997.
[21] I. Hooks, Writing good requirements, in: Proceedings of the 3rd NCOSE International Symposium, vol. 2, 1993, vol. 2, 1993, pp. 1–11, http://dx.doi.org/10.1002/j.2334-5837.1994.tb01834.x.
[22] L.H. Rosenberg, H. Linda, Generating high quality requirements, in: Proceedings of the AIAA space 2001 conference and exposition, AIAA paper, 2001, pp. 28–30.
[23] I. Alexander, R. Stevens, Writing Better Requirements, Addison-Wesley, 2002.
[24] W. Turk, Writing requirements, (2006) 20–23.
[25] S.E.S. Committee, IEEE Recommended Practice for Software Requirements Specifications, 1998.
[26] J. Bøegh, A new standard for quality requirements, IEEE Softw. 25 (2008) 57–63, http://dx.doi.org/10.1109/MS.2008.30.
[27] ISO/IEC, Software Engineering—Software Product Quality Requirements and Evaluation (SQuaRE), Qual. Requir. Int'l Organ. Stand. ISO/IEC 25, 2007.
[28] I.F. Hooks, Guide for Managing and Writing by, 2000.
[29] E.S.A. (ESA), Guide to the Software Requirements Definition Phase, 1995.
[30] INCOSE (International Council on Systems Engineering), n.d. <http://www.incose.org/> (last accessed 01.05.15).
[31] Reqtify, 3DS DassaultSystèmes, 3DS DassaultSystèmes, n.d. <http://www.3ds.com/> (last accessed 20.05.02).
[32] Caliber, Borland, Borland, n.d. <http://www.borland.com/> (last accessed 01.05.15).
[33] IBM, Rational DOORS. Rational RequisitePro, IBM, n.d. <http://www.IBM.com> (last accessed 01.05.15).
[34] T.R. Company, RQA Requirements Quality Analyzer, Reuse Co., n.d. <http://www.reusecompany.com> (last accessed 01.05.15).
[35] V. Solutions, Visure Solutions, Visure Solut., n.d. <http://www.visuresolutions.com/> (last accessed 01.05.15).
[36] F.J. Chantree, A. De Roeck, B. Nuseibeh, A. Willis, Identifying Nocuous Ambiguity in Natural Language Requirements, Fac. Maths Comput. Doctor of, 2006, p. 203.
[37] D. Popescu, S. Rugaber, N. Medvidovic, D.M. Berry, Reducing ambiguities in requirements specifications via automatically created object-oriented models, in: Lecture Notes on Computer Science (including Subseries Lecture Notes on Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 5320 LNCS, 2008, pp. 103–124, http://dx.doi.org/10.1007/978-3-540-89778-1_10.
[38] N. Kiyavitskaya, N. Zeni, L. Mich, D.M. Berry, Requirements for tools for ambiguity identification and measurement in natural language requirements specifications, Requir. Eng. 13 (2008) 207–239, http://dx.doi.org/10.1007/s00766-008-0063-7.
[39] Y. Wang, I.L.M. Gutiérrez, K. Winbladh, H. Fang, Automatic detection of ambiguous terminology for software requirements, in: Natural Language Processing and Information Systems, Springer, 2013, pp. 25–37.
[40] T.C. de Sousa, J.R. Almeida, S. Viana, J. Pavón, Automatic analysis of requirements consistency with the B method, ACM SIGSOFT Softw. Eng. Notes 35 (2010) 1, http://dx.doi.org/10.1145/1734103.1734114.
[41] A. Sardinha, R. Chitchyan, N. Weston, P. Greenwood, A. Rashid, EA-Analyzer: automating conflict detection in a large set of textual aspect-oriented requirements, Autom. Softw. Eng. 20 (2013) 111–135, http://dx.doi.org/10.1007/s10515-012-0106-7.
[42] R. Ali, F. Dalpiaz, P. Giorgini, Reasoning with contextual requirements: detecting inconsistency and conflicts, Inf. Softw. Technol. 55 (2013) 35–57, http://dx.doi.org/10.1016/j.infsof.2012.06.013.
[43] D. Aceituna, G. Walia, H. Do, S.-W. Lee, Model-based requirements verification method: conclusions from two controlled experiments, Inf. Softw. Technol. 56 (2014) 321–334, http://dx.doi.org/10.1016/j.infsof.2013.11.004.
[44] R. Thakurta, A framework for prioritization of quality requirements for inclusion in a software project, Softw. Qual. J. 21 (2013) 573–597, http://dx.doi.org/10.1007/s11219-012-9188-5.
[45] C.E. Otero, E. Dell, A. Qureshi, L.D. Otero, A quality-based requirement prioritization framework using binary inputs, in: AMS2010 Asia Modelling Symposium in 2010–4th International Conference on Mathematical Modelling, Computation and Simulation, 2010, pp. 187–192, http://dx.doi.org/10.1109/AMS.2010.48.
[46] J.R. McCoy, NASA software tools for high-quality requirements engineering, in: Proceedings of the 26th Annual NASA Goddard Software Engineering Workshop, 2001, p. 20771, http://dx.doi.org/10.1109/SEW.2001.992657.
[47] D. Ott, Automatic requirement categorization of large natural language specifications at Mercedes-Benz for review improvements, Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 7830 LNCS, 2013, pp. 50–64, http://dx.doi.org/10.1007/978-3-642-37422-7_4.
[48] Y. Ko, S. Park, J. Seo, S. Choi, Using classification techniques for informal requirements in the requirements analysis-supporting system, Inf. Softw. Technol. 49 (2007) 1128–1140, http://dx.doi.org/10.1016/j.infsof.2006.11.007.
[49] H. Jani, A. Islam, A Framework of Software Requirements Quality Analysis System using Case-Based Reasoning and Neural Network, Ieeexplore.Ieee.Org., n.d., pp. 152–157 <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6528619>.
[50] I. Hussain, O. Ormandjieva, L. Kosseim, Automatic quality assessment of SRS text by means of a decision-tree-based text classifier, in: Proceedings of the International Conference on Quality Software, 2007, pp. 209–218, http://dx.doi.org/10.1109/QSIC.2007.4385497.
[51] J.L. Dargan, E. Campos-Nanez, P. Fomin, J. Wasek, Predicting systems performance through requirements quality attributes model, Proc. Comput. Sci. 28 (2014) 347–353, http://dx.doi.org/10.1016/j.procs.2014.03.043.
[52] M. Harman, P. McMinn, J. de Souza, S. Yoo, Search Based Software Engineering: Techniques, Taxonomy, Tutorial, Springer, Berlin Heidelb., 2012, pp. 1–59, http://dx.doi.org/10.1007/978-3-642-25231-0_1.
[53] M. Harman, B.F. Jones, Search-based software engineering, Inf. Softw. Technol. 43 (2001) 833–839, http://dx.doi.org/10.1016/S0950-5849(01)00189-6.
[54] Y. Zhang, M. Harman, S. Mansouri, The Multi-Objective Next Release Problem, 2007, pp. 1129–1136, http://dx.doi.org/10.1145/1276958.1277179.
[55] CESAR, CESAR Project, n.d. <http://www.cesarproject.eu/> (last accessed 01.05.15).
[56] ARTEMIS, CRYSTAL, n.d. <http://www.crystal-artemis.eu/> (last accessed 01.05.15).

[57] E. Hull, K. Jackson, J. Dick, Requirements Engineering, Springer Science & Business Media, 2010.
[58] A. Fraga, A Methodology for Reusing Any Kind of Knowledge at Low Cost: Universal Knowledge Reuse, University Carlos III de Madrid, 2010.
[59] J. Bloomberg, R. Schmelzer, Service Orient or be Doomed!: How Service Orientation Will Change Your Business, John Wiley & Sons, 2006.
[60] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The WEKA data mining software, ACM SIGKDD Explor. 11 (2009) 10–18, http://dx.doi.org/10.1145/1656274.1656278.
[61] V. Moreno, Análisis de los criterios de relevancia documental mediante consultas de información en el entorno WEB, University Carlos III de Madrid, 2010.