2021 IEEE 29th International Requirements Engineering Conference (RE) | 978-1-6654-2856-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/RE51729.2021.00043

Text Mining for Standardized Quality Criteria of Natural-Language IT-Requirements

1st Erik Buchmann
Hochschule für Telekommunikation
04288 Leipzig, Germany
buchmann@hft-leipzig.de

2nd M.Sc. Serda Hauser
Leipzig University
04109 Leipzig, Germany
serda.hauser@studserv.uni-leipzig.de

***Abstract*—Without a precise specification, an IT project might not remain on time and on budget constraints, or it might lead to a different outcome than desired. A number of established standards define how requirements must be written to avoid such issues. This paper describes our ongoing work to derive a comprehensive set of standardized criteria that IT-requirements must meet in accordance with IEEE 1233-1996 and ISO/IEC/IEEE 29148-2011. We also use a text-mining approach to identify IT-requirements that violate these standards. Our preliminary results are promising: In our biased dataset, we can use text features that are easy to compute, to filter out requirements that do not comply with the standards. Our beneficiaries are auditors, developers, Scrum teams, customers and other stakeholders whose projects are highly dependent on extensive IT-requirements specification.**

***Index Terms*—requirements quality; text mining;**

I. Introduction

The foundation of a successful IT project is an elaborate, precise requirements specification. A requirements specification contains functional and non-functional requirements for an IT component. IT-requirements are often defined in natural language. This leaves room for mistakes or specification errors. For example, different authors might describe the same requirement in different words, requirements might overlap or contradict with each other, aspects could be omitted that are deemed obvious, or the used language is hard to understand for non-native speakers. Since a specification is generated at the beginning of an IT project, such errors in the specification can result in a wide range of expensive problems at latter stages of the project. To avoid those issues, standards like IEEE 1233-1996 [1] and ISO/IEC/IEEE 29148-2011 [2] demand that requirements must fulfill certain quality criteria. Those criteria include consistency, feasibility, readability or traceability.

A large number of Natural language processing (NLP) approaches have been proposed to assess the quality of natural-language requirements (see [3], [4] for an overview). One approach is to apply data mining on text features, e.g., the number of adjectives or passive phrases, to recognize defects like ambiguity, variability or writing quality in requirements. Alternatively, neural networks [5] can be used. Other approaches identify best practices [6]. Tools like QuARS [7] implement quality models [8] automatically identify potential linguistic defects, and requirements that seem to be incomplete or inconsistently written. The approaches support an author to avoid common mistakes when writing requirements. However, it is unclear yet how to quantify whether requirements fulfill a standard or not. This is because standards aim for requirements that are manageable and applicable in a large organization, which is different from searching for defects and anti-pattern.

In this paper, we identify a complete set of common criteria for IT-requirements that are brought forward by the standards. We want to use this set of criteria to build a score for the quality of the requirements specification. We also want to automatically identify IT-requirements which do not meet those criteria, i.e., which violate the standards. This is challenging. Standards are defined on different levels of abstraction, and it is difficult to find common grounds for requirements criteria. It is also not obvious if a requirement that meets one standard fulfills another one as well. Consider Example 1 and 2:

Example 1 *"To request an access token, the client obtains authorization from the resource owner. The authorization is expressed in the form of an authorization grant, which the client uses to request the access token. OAuth defines four grant types, authorization code, implicit, resource owner password credentials, and client credentials."*

Example 2 *"Emerged vulnerabilities in software and hardware of a system must be fixed or protected against misuse. Prior to installation of a software or hardware component, users must check whether any vulnerability has been discovered and published for the version they are installing."*

Example 1 is a part of the OAuth 2.0 Authorization Framework specification [9]. Example 2 is a fragment of the security requirements [10] from Deutsche Telekom. Intuitively, the first example is better specified than the second one. To name one aspect, Example 2 does not specify in which way the users are expected to find vulnerabilities. Example 1 fulfills IEEE 1233-1996 and ISO/IEC/IEEE 29148-2011, while Example 2 does not. Now assume an auditor needs to find out which requirements from a vast body of documents do not fulfill such standards. In this context, we define our problem statement as follows:

Given a large set of IT-requirements. Which text mining techniques allow to identify requirements that are likely to violate standardized quality criteria for requirements?

To tackle this problem, (I) we strive for a high recall for requirements that violate standards, (II) we prefer features and algorithms that are computationally inexpensive, and (III) we

978-1-6654-2856-9/21/$31.00 ©2021 IEEE
DOI 10.1109/RE51729.2021.00043

seek for classifiers that can be explained to an auditor. In this paper, we make the following contributions:

- We describe the set of 15 criteria for requirements specifications we have extracted from the standards.
- To identify requirements that violate those criteria, we train a classifier with text features.
- We test this approach by means of experiments with 30 requirements from three sources, and we compare the assessments of an expert with our classifier.

Our preliminary results indicate that this approach is promising. We can identify up to 100% ill-written requirements in a data set where 80% of the requirements are well specified. We obtained those results without computationally expensive approaches like comparing semantic similarities of words or extracting contextual features of sentences.

Section II reviews related work. Section III describes our research method. Sections IV and V contain our first results towards a generic set of quality criteria and a data mining approach that uses them. Sections VII concludes the paper.

II. Related Work

This section outlines existing work on standards for requirements and automated assessment of requirements quality.

A. Requirements Specifications

Quality requirements, also known as non-functional requirements, are among the ten most significant risks in requirements engineering (RE) [11]. Appearance requirements specify the customer's ideas about the product's final impression [12]. **Quality criteria** for requirements specify "what" is needed, not "how". Requirements should state what is needed for the system of interest and not include design decisions. Vague and general terms should be avoided because they result in difficult or impossible requirements to verify or allow for multiple interpretations [2]. To achieve this, quality must be generated in the first place, which is why quality-oriented process models are necessary [13].

To approach at requirements serve its purpose in large organizations with multiple different stakeholders involved, a number of standards have been developed. **IEEE 1233-1996** [1] ("Guide for Developing System Requirements Specifications") and **ISO/IEC/IEEE 29148-2011** [2] ("Requirements for Systems and Software Engineering") complement each other in describing characteristics of requirements. Both standards define the scope and boundaries of system/software requirements [14], and provide measures for requirements engineers. ISO/IECIEEE 29148-2011 can be considered the mother of all requirements standards. It contains a comprehensive description of RE's field, and is adapted by 12 other standards. It explains what a well-formed requirement is, contains templates for textual requirements, and provides characteristics of good requirements specifications [15]. Observe that the standards were developed to foster requirements management. This includes aspects such as versioning, or verifying and tracing implements back to its specification. Thus, existing quality models [8] that focus on specification defects are not sufficient to find out if a requirement conforms to a certain standard.

B. Automated Requirements Assessment

A broad range of automated **NLP** approaches has been proposed to assess the quality of natural-language requirements. The approaches we are aware of focus on aspects such as ambiguity [16], [17], completeness [18], equivalence [19], variability [7], writing quality [20], [21], templates based on best practices [22], [6] or anti-pattern referring to bad practices [23], [24]. Alternatively, **neural networks** can be trained to single out badly specified requirements [5] without having to provide an exhaustive definition for "badly specified". See [3], [4] for a comprehensive view of NLP approaches on requirement specifications.

A number of **tools** implement such approaches, e.g., QuARS [7], SREE [25] or CIRCE [26]. However, to the best of our knowledge, there is no approach to comprehensively mapping well-established standards on requirements quality to natural-language specifications.

C. Text Features

To make texts accessible for data mining, well-structured features must be derived from the less-structured text corpus. We focus on four categories of features:

Text statistics calculate the average number of words per sentence, the proportion of very long words, or in our context the total length of a requirement. Such statistics are assigned with hypotheses like "requirements that are too short could be incomplete" or "requirements that make use of many long words can be difficult to understand".

Features from **part-of-speech tagging** include the average number of adjectives, modal verbs, passive forms or filler words per sentence [27]. Our hypothesis behind such features is that, say, passive forms like "something must be done" leave open which role is assigned with a task, rendering a requirement incomplete. Similarly, modal verbs like "can", "may" or "would" make a requirement unspecific.

The **sentiment analysis** assigns sentences with positive or negative mood [28]. Requirements are expected to be neutral. We expect that a requirement might have issues, if it uses a positive or negative wording. Thus, it is sufficient to search for words that carry a sentiment – it is not necessary to interpret cases such as double negatives ("I do not dislike...").

Good requirements are clearly understandable and do not leave room for mistakes. **Readability metrics** assess the understandability of a text. The Flesch-Reading-Ease (*FRE*) maps the average length of a sentence ASL and the average number of syllables per word ASW to a scale: $FRE = 180 - ASL - (ASW \cdot 58.5)$. Numbers below 30 indicate texts that are very difficult to read [29]. The Swedish Lasbarhetsindex (*LIX*) is calculated from the average length of a sentence ASL and the percentage FLW of words with more than six characters: $LIX = ASL \cdot FLW$. Numbers above 60 indicate very difficult texts [30]. The Gunning Fog Index (*FOG*) is calculated from the average sentence length ASL and the

percentage CPW of words with three or more syllables: $FOG = 0.4 \cdot (ASL + CPW)$. Numbers above 16 indicate very difficult texts [29]. All constants have been gauged for English texts, i.e., they would be different for other languages.

III. Research Method

To systematically identify ill-defined requirements among a set of numerous specifications, we need (1) an exhaustive list of quality criteria, (2) test data labeled with those criteria, and (3) a concept to apply text mining. To this end, we propose the following research process:

1) Browse IEEE 1233-1996 [1] and ISO/IEC/IEEE 29148-2011 [2] to create a catalog of quality criteria for requirements specifications.
2) Consolidate this catalog by grouping criteria with the same meaning, and by rephrasing criteria that have been specified on different levels of abstraction.
3) Derive thresholds from the standards to determine if a certain requirement is either specified, partially specified or unspecified according [1] and [2].
4) Create a database of requirements as test data. Manually assess to which degree each requirement fulfills each criterion, and provide the labels "specified", "partially specified" or "unspecified" according to the thresholds.
5) Compute a feature vector for each requirement in the database by using common measures from Part-of-Speech tagging, Sentiment Analysis, and Readability Analysis.
6) Find out which features are linked with which criteria, and are suitable to label the requirements automatically.
7) Test the labeling with real-world data. Develop new features if its accuracy is insufficient to ease the quality assessment of a vast share of requirements.

This process resembles the Design Science [31] approach. The first four steps are the relevance cycle, the next two steps are the design cycle and the last step is the research rigor. The catalog of quality criteria allows deciding if a requirement conforms to standards objectively. We want to learn if text features allow to classify requirements with the same accuracy, i.e., we declare success if our approach automatically classifies all ill-defined requirements as "unspecified" or "partially specified". We also want to know which text features are correlated with ill-defined requirements.

IV. Quality Criteria for Requirements

This section introduces our quality criteria and the set of requirements we use as test data.

A. Obtaining quality criteria

To approach a complete set of quality criteria that conform to well-established standards, we extracted all demands for a requirements specification from IEEE 1233-1996 [1] and ISO/IEC/IEEE 29148-2011 [2]. Both standards are described at different levels of abstraction. Thus, we unified requirements that appeared in both standards in different ways. For example, "a requirement shall be consistent with other requirements for the same subject" has the same meaning as "a requirement shall not contradict other requirements." We obtained 15 different requirement criteria, as shown in Table I. The sorting of the criteria corresponds to [1].

TABLE I
Quality criteria for requirements

Criterion	*Description*
Unique set	A requirement shall not exist twice in the same specification catalog.
Normalized	A requirement shall not depend on other requirements in the same specified catalog.
Linked set	A requirement shall have a connection to the specified subject.
Complete	A requirement describes all features and characteristics of the subject in a measurable form.
Consistent	All requirements shall be described at the same level of detail.
Bounded	A requirement shall identify the boundaries, scope, and context.
Modifiable	The requirement shall be modifiable. Clarity and non-overlapping requirements contribute to this.
Configurable	Versioning allows the maintenance of requirements over time and across instances.
Granular	The requirement shall be the abstraction level for the system to be defined.
Unambiguous	Each requirement shall be understandable easily and clear, so it can be interpreted in only one way.
Singular	The requirement statement contains only one requirement, with no use of conjunctions.
Feasible	The role who is assigned with the implementation of a requirement is indeed able to implement it.
Traceable	The creation date and the creator of a requirement shall be traceable.
Verifiable	A requirement shall be verifiable so that the system satisfies the specification.
Affordable	A requirement shall be affordable from the point of budget and schedule.

B. Test Data

We test this set of criteria with software specifications related to security. For a detailed target/actual-comparison between our approach and the standards, an expert in Requirements Engineering had to determine 15 different quality criteria for each requirement manually. Thus, our data set must be restricted to a manageable size. To include a wide range of strategies to write requirements, our test data includes 30 requirements from three different data sources:

R1 An example for a standardized specification is RFC 6749 "OAuth 2.0 Authorization Framework" [9] by the Internet Engineering Task Force. This document defines a protocol to obtain authorized access to an Internet service. Our test data contains the first 10 requirements from Section 3 "Protocol Endpoints", including Example 1.

R2 A specification from a large company is "Security Requirement – Client Applications" [10] by Deutsche Telekom Group. It contains requirements for software products rolled out to services. Our test data contains the first 10 requirements from Section 2 "General security requirements for apps", including Example 2.

R3 "GParted Requirements for Version 0.6.0-1" [32] is an example of an Open Source specification made by volunteers. GParted is a Linux tool to partition and format

TABLE II
AVERAGE QUALITY ASSESSMENT

	R1	*R2*	*R3*
Unique set	100%	100%	100%
Normalized	85%	100%	100%
Linked set	100%	80%	100%
Complete	97%	80%	100%
Consistent	92%	80%	94%
Bounded	93%	73%	93%
Modifiable	100%	100%	100%
Configurable	0%	0%	0%
Granular	80%	60%	90%
Unambiguous	90%	70%	100%
Singular	10%	20%	30%
Feasible	97%	80%	100%
Traceable	0%	0%	0%
Verifiable	80%	50%	100%
Affordable	90%	80%	100%
specified	8	7	9
partially specified	1	1	1
unspecified	1	2	0
total	10	10	10

hard disks. Our test data contains the first 10 requirements from Section 3 "System Features."

To prepare the data set, we extracted plain text from PDF and HTML sources, we only considered complete, grammatically correct sentences, we replaced abbreviations with their long-form, and we removed examples and annotations that were not part of the specification. In total, we have obtained a set of 30 requirements consisting of 420 sentences, 8,104 words and 51,151 characters.

C. Requirements Quality

To find out if our set of quality criteria distinguishes standard-compliant requirements from ill-written ones, we let an expert manually evaluate each criterion for every one of our 30 requirements. The expert can reflect on many IT projects with approximately ten years of experience in his full-time job as a requirements engineer. Table II contains the average assessment for our three sources R1, R2 and R3. We assume all criteria to be equally important, and each criterion contributes 1/15 to the total quality of a requirement. Example 1 scored a quality of 87%, and Example 2 obtained 26%.

We assign each requirement with a label "specified", "partially specified" or "unspecified", to train a classifier in the next section. By following practical experiences from our expert, we assume a requirement is *unspecified*, if it fulfills the set of criteria by less than 60%, i.e., it fulfills less than 9 of our 15 criteria. According to our expert, a requirement can be assumed as *specified*, if it meets at least 75% of our criteria. The requirements between this range are labeled as *partially specified*. The last block of Table II shows how many requirements have been assigned with which labels. Only three requirements were labeled "unspecified", as it can be expected for requirements that were in practical use.

V. AUTOMATED LABELING

This section contains our text mining setup to classify the requirements, together with our preliminary results.

TABLE III
TEST DATA

Features	*R1*	*R2*	*R3*
Words per requirement	164,8	499,6	148
Characters per word	5,49	5,22	4,56
Ratio of long words	0,00	0,01	0,00
Sentences per requirement	7,50	24,8	10,20
Words per sentence	21,97	20,14	14,50
Ratio of long sentences	0,16	0,13	0,01
Adjectives per sentence	1,41	1,79	0,83
Adverbs per sentence	0,86	0,75	0,59
Modals per sentence	0,2	0,72	0,39
Conjunctions per sentence	0,32	0,16	0,13
Passives per sentence	0,37	0,11	0,27
Filler words per sentence	0,00	0,11	0,04
Positive words per sentence	0,00	0,00	0,00
Negative words per sentence	0,00	0,00	0,00
FRE	28,78	35,10	68,46
FOG	15,63	15,69	9,53
LIX	55,95	51,95	35,52

A. Text Features

We have implemented the text features described in Section II to complete our training data for the classifier.

We calculated **text statistics** by using the R library "koRpus". The thresholds of 15 characters for long words and 20 words for long sentences stem from readability metrics [29]. As Table III shows, our data sources are very different. Even though R2 tends to spend more text for each requirement, no set of requirements uses overly long or complex sentences with many long words.

We used the R libraries "NLP", "openNLP", "stringr" and "koRpus" [27] to calculate features from **part-of-speech tagging**. The detection of passive forms requires a variant of "be" and a past participle. To detect filler words like "absolutely", "generally" or "simply", we used a list of 71 prominent filler words [33]. Table III indicates that our sources of requirements use such terms to a varying extent. Note that criterion "Singular" only forbids conjunctions that put two different requirements together, while text feature "Conjunctions per sentence" also finds allowed conjunctions.

In our context, it is sufficient to count words with a positive or negative mood for **sentiment analysis** [28]. To detect positive feelings, we used a list of 187 words, such as "terrific", "splendid", "brilliant" from [34]. Similarly, we detected negative feelings with a list of 218 words like "dire", "stubborn" or "touching" from [34]. Table III shows that no requirement in our data set expressed feelings.

We calculated the **readability metrics** FRE, FOG and LIX by using the R library "koRpus". The average readability of our three sources of requirements is shown in Table III. On average, R3 (GParted) needs a middle reading competence, while the other requirements are complex or very hard to read.

B. Data Mining Setup

Our objective is to find out which requirements do (not) comply with standards on requirements specification. To this end, we use a supervised learning approach to train a classifier

with the text features and the labels "specified", "partially specified" or "unspecified" obtained from our expert. We evaluate the classifier as follows: The **Accuracy** is the proportion of instances that are correctly labeled. A **Kappa** of 1 means that our classifier is always correct, and 0 means that it performs like guessing randomly. We also calculate the **True Positive rate** (TP) for each label. Our test data contains 30 instances, and 24 of them have been labeled as "specified" by our expert. A classifier which always labels "specified" without considering features would have an accuracy of 80%, a Kappa of 0, a TP=100% for the label "specified" and a TP=0% for "partially specified" and "unspecified". To evaluate our approach, we want to compare the classifier model with the decisions of our expert. Thus, we favor a deterministic classifier, and we have decided on the C4.5 decision tree algorithm [35] implemented in Weka [36]. We use the same data for training and evaluating the classifier, and we allow a single instance per leaf. To avoid overfitting, we use a confidence factor of 50% for pruning. All other parameters are Weka's standard values.

C. First Mining Results

Table III shows that no requirement refers to positive or negative sentiment. Thus, we test only text statistics, part-of-speech features and readability metrics. To improve our set of text features, we need to know how strong each category of features is linked to the requirements quality. Thus, we train a classifier from each category in isolation.

TABLE IV
LABELING ACCURACY

	Statistics	*Part-of-Speech*	*Readability*
Tree size	5	7	1
Accuracy	90%	87%	80%
Kappa	0.62	0.46	0.00
TP specified	100%	100%	100%
TP partially	0%	0%	0%
TP unspecified	100%	66%	0%

The last column of Table IV shows that FRE, FOG and LIX were not correlated with our test data labels. Thus, the readability features do not single out ill-written requirements in our test data. The generated decision tree consists of only one node which always says "specified". In contrast, a classifier from part-of-speech features allows assigning 87% of the data with the correct label. An analysis reveals that the decision tree uses features "Filler words per sentence" and "Passives per sentence". This is in line with our expectations: A requirement that uses filler words and passive forms is always less precise than a short active form. Other features were below 50% confidence, and pruned from the decision tree. The classifier from text statistics was the most accurate one. It tells us that the label "unspecified" requirements are either very long or use many complicated long words.

We are also interested in finding out how well a classifier can identify single quality criteria that have not been met. We say that a requirement does not fulfill a criterion, if our expert has asseded it less than 100%. From Table II follows that "Unique set" and "Modifiable" are always fulfilled, while "Configurable" and "Traceable" are not fulfilled by any requirement of our test data. We re-use our decision tree setup for the remaining criteria, but we use the entire set of features.

TABLE V
IDENTIFYING QUALITY ISSUES

	Tree size	*TP issues*	*Accuracy*	*Kappa*
Normalized	7	100%	100%	1.00
Linked set	3	100%	100%	1.00
Complete	3	66%	97%	0.78
Consistent	11	66%	90%	0.73
Bounded	3	40%	90%	0.52
Granular	3	28%	83%	0.38
Unambiguous	3	50%	93%	0.63
Singular	7	100%	96%	0.89
Feasible	3	50%	93%	0.63
Verifiable	3	28%	83%	0.38
Affordable	3	66%	97%	0.78

Table V reports the size of the tree, the true positive rate of requirements with quality issues, the total accuracy and Kappa, which tells us if our classifier is better than guessing. We deem our results positive. Without computationally expensive NLP approaches like comparing semantic word similarities, measuring term frequency–inverse document frequency from word vectors, or extracting contextual features, our approach has identified five quality criteria with a Kappa > 0.75. However, a tree size of 3 indicates a classifier that uses only one feature, e.g., the ratio of modal verbs or very long words in a requirement. Experience shows that classifiers based on one feature are less applicable to other data sources written in a different style. Thus, we see much potential for future research.

VI. DISCUSSION

We have evaluated our approach with a data set that was inevitably restricted to a manageable size. However, we have tested requirements from three different specifications, which were in use for some time and already have achieved a high specification quality.

We have observed a true positive rate of 100% for the label "unspecified" with this data set. Our approach has found a clear correlation between computationally-inexpensive text features (cf. Table III) and the decisions of our expert (cf. Table II). Thus, we declare success, according to our objectives from Section III. With a larger data set, we expect even more correlations. However, with less mature requirements specifications, other text features such as the readability metrics might be correlated with non-conformant requirements, too.

A promising candidate for a larger data set is the set of 8706 active RFCs [37] that contain the requirements for many Internet protocols: The RFCs are publicly available, have undergone a strict editing process and have been carefully maintained and improved over the years. The number of RFCs is also sufficiently large to test Deep Learning approaches.

VII. Summary

In this paper, we have outlined our work in progress towards identifying IT-requirements that do not meet common quality criteria from IEEE 1233-1996 and ISO/IEC/IEEE 29148-2011. We have tested our approach with a restricted data set of 30 requirements from three different sources. This is because a target/actual-comparison between our approach and the standards requires an expert to manually determine and compare 15 different quality criteria for each requirement. With our data set, we have observed that our set of criteria is suitable to assess the standard conformity of requirement specifications objectively. In the future, we will develop weights to consider that, say, verifiability is more mission-critical for a requirement than having a version number.

Text mining is promising to label requirements that are likely to violate the standards. With our 30 requirements from three different sources, we have observed a true positive rate of 100% for the label "unspecified". Thus, our approach finds all three badly-specified requirements among 30 others. Assume a person needs to ensure that a vast body of requirements meets IEEE 1233-1996 or ISO/IEC/IEEE 29148-2011. Our preliminary results recommend examining those requirements first which are either very long, or use many long words, passive sentences or filler words. Note that these features are computationally inexpensive, i.e., can be applied to large data sets.

References

[1] IEEE 1233-1996, Guide for Developing System Requirements Specifications. IEEE Standards Association, 1996.

[2] ISO/IEC/IEEE 29148-2011, Systems and software engineering – Life cycle processes –Requirements engineering. IEEE Standards Association, 2011.

[3] T. Iqbal, P. Elahidoost, and L. Lucio, "A bird's eye view on requirements engineering and machine learning," in 2018 25th Asia-Pacific Software Engineering Conference. IEEE, 2018.

[4] L. Zhao, W. Alhoshan, A. Ferrari, K. J. Letsholo, M. A. Ajagbe, E.-V. Chioasca, and R. T. Batista-Navarro, "Natural language processing (nlp) for requirements engineering: A systematic mapping study," arXiv preprint arXiv:2004.01099, 2020.

[5] J. Winkler and A. Vogelsang, "Automatic classification of requirements based on convolutional neural networks," in 24th IEEE International Requirements Engineering Conference Workshops, 2016.

[6] H. Femmer, "Requirements engineering artifact quality: definition and control," Ph.D. dissertation, Technische Universität München, 2017.

[7] A. Fantechi, S. Gnesi, and L. Semini, "Applying the quars tool to detect variability," in Proceedings of the 23rd International Systems and Software Product Line Conference, 2019.

[8] D. M. Berry, A. Bucchiarone, S. Gnesi, G. Lami, and G. Trentanni, "A new quality model for natural language requirements specifications," in Proceedings of the international workshop on requirements engineering: foundation of software quality, 2006.

[9] Internet Engineering Task Force, "RFC 6749 OAuth 2.0 Authorization Framework," https://tools.ietf.org/html/rfc6749, Accessed 10/11/2020, 2012.

[10] Deutsche Telekom Group, "Security requirement – client applications, version 2.1, oct 30, 2016," https://www.telekom.com/en/corporate-responsibility/data-protection-data-security/security/details/privacy-and-security-assessment-process-358312, Accessed 10/11/2020, 2016.

[11] B. Lawrence, K. Wiegers, and C. Ebert, "The top risk of requirements engineering," IEEE Software, vol. 18, no. 6, 2001, pp. 62–63.

[12] J. Robertson and S. Robertson, Volere – Requirements Specification Templates. Atlantic Systems Guild Limited, 2000.

[13] K. Pohl, "The three dimensions of requirements engineering: a framework and its applications," Information systems, vol. 19, no. 3, 1994, pp. 243–258.

[14] G. Bochmann, "Purpose and structure of requirements specifications (following ieee 830 standard)," University of Ottawa, slide only, 2010.

[15] F. Schneider and B. Berenbach, "A literature survey on international standards for systems requirements engineering," Procedia Computer Science, vol. 16, 2013, pp. 796–805.

[16] V. Gervasi and D. Zowghi, "On the role of ambiguity in RE," in International Working Conference on Requirements Engineering: Foundation for Software Quality. Springer, 2010.

[17] U. S. Shah and D. C. Jinwala, "Resolving ambiguities in natural language software requirements: a comprehensive survey," ACM SIGSOFT Software Engineering Notes, vol. 40, no. 5, 2015, pp. 1–7.

[18] A. Ferrari, F. dell'Orletta, G. O. Spagnolo, and S. Gnesi, "Measuring and improving the completeness of natural language requirements," in International Working Conference on Requirements Engineering: Foundation for Software Quality. Springer, 2014.

[19] D. Falessi, G. Cantone, and G. Canfora, "A comprehensive characterization of nlp techniques for identifying equivalent requirements," in Proceedings of the 2010 ACM-IEEE international symposium on empirical software engineering and measurement, 2010.

[20] O. Ormandjieva, I. Hussain, and L. Kosseim, "Toward a text classification system for the quality assessment of software requirements written in natural language," in 4th international workshop on software quality assurance, 2007.

[21] B. Sateli, E. Angius, S. S. Rajivelu, and R. Witte, "Can text mining assistants help to improve requirements specifications," in Workshop on Mining Unstructured Data, 2012.

[22] C. Arora, M. Sabetzadeh, L. Briand, and F. Zimmer, "Automated checking of conformance to requirements templates using natural language processing," IEEE transactions on Software Engineering, vol. 41, no. 10, 2015, pp. 944–968.

[23] H. Femmer, D. M. Fernández, S. Wagner, and S. Eder, "Rapid quality assurance with requirements smells," Journal of Systems and Software, vol. 123, 2017, pp. 190–213.

[24] A. Ferrari, G. Gori, B. Rosadini, I. Trotta, S. Bacherini, A. Fantechi, and S. Gnesi, "Detecting requirements defects with NLP patterns: an industrial experience in the railway domain," Empirical Software Engineering, vol. 23, no. 6, 2018, pp. 3684–3733.

[25] S. F. Tjong and D. M. Berry, "The design of SREE —- A prototype potential ambiguity finder for requirements specifications and lessons learned," in International Working Conference on Requirements Engineering: Foundation for Software Quality. Springer, 2013.

[26] V. Ambriola and V. Gervasi, "On the systematic analysis of natural language requirements with CIRCE," Automated Software Engineering, vol. 13, no. 1, 2006, pp. 107–167.

[27] H. Schmid, "Probabilistic part-ofispeech tagging using decision trees," in New methods in language processing, 2013.

[28] P. Gonçalves, M. Araújo, F. Benevenuto, and M. Cha, "Comparing and combining sentiment analysis methods," in Proceedings of the first ACM conference on Online social networks, 2013.

[29] E. Robinson and D. McMenemy, "To be understood as to understand," Journal of Librarianship and Information Science, vol. 52, no. 3, 2020, pp. 713–725.

[30] J. Anderson, "Analysing the Readability of English and Non-English Texts in the Classroom with Lix." Darwin, August (ED 207 022), 1981.

[31] A. Hevner and S. Chatterjee, "Design science research in information systems," in Design research in information systems. Springer, 2010.

[32] B. Karatzidis, "Software requirements specification for gparted," https://gparted.org/docs/sw-req-spec/C/Software_Requirements_Specification_-_GParted.pdf, Accessed 10/11/2020, 2010.

[33] Grammarly Inc., "Real-time suggestions, wherever you write. what are filler words, and how do you cut them?" https://www.grammarly.com/blog/how-we-use-filler-words/, Accessed 10/11/2020, 2019.

[34] H. Possel, "Synonyms for feelings," https://www.smart-words.org/words-for/feelings/, Accessed 10/11/2020, 2020.

[35] R. Quinlan, C4.5: Programs for Machine Learning. Morgan Kaufman, 1993.

[36] Machine Learning Group at the University of Waikato., "WEKA - The workbench for machine learning," https://www.cs.waikato.ac.nz/ml/weka/, Accessed 10/11/2020, 2020.

[37] RFC Production Center, "RFC Editor," https://www.rfc-editor.org/, Accessed 05/06/2021, 2021.