

Using NLP Tools to Detect Ambiguities in System Requirements - A Comparison Study

Aleksandar Bajceta¹, Miguel Leon¹, Wasif Afzal¹, Pernilla Lindberg² and Markus Bohlin¹

¹*Mälardalens universitet, Box 325, 631 05 Eskilstuna*

²*Alstom Sweden, 721 14 Västerås*

Abstract

Requirements engineering is a time-consuming process, and it can benefit significantly from automated tool support. Ambiguity detection in natural language requirements is a challenging problem in the requirements engineering community. Several Natural Language Processing tools and techniques have been developed to improve and solve the problem of ambiguity detection in natural language requirements. However, there is a lack of empirical evaluation of these tools. We aim to contribute the understanding of the empirical performance of such solutions by evaluating four tools using the dataset of 180 system requirements from the electric train propulsion system provided to us by our industrial partner Alstom. The tools that were selected for this study are Automated Requirements Measurement (ARM), Quality Analyzer for Requirement Specifications (QuARS), REquirements Template Analyzer (RETA), and Requirements Complexity Measurement (RCM). Our analysis showed that selected tools could achieve high recall. Two of them had the recall of 0.85 and 0.98. But they struggled to achieve high precision. The RCM, an in-house developed tool by our industrial partner Alstom, achieved the highest precision in our study of 0.68.

Keywords

Requirements engineering, Natural language requirements, Ambiguity, Natural language processing

1. Introduction

Good requirements engineering is the backbone of every successfully developed system. It is a process where engineers define the stakeholders' needs and their transformation into clearly described systems' behavior. Many projects have failed due to a poor requirements engineering process. According to [1], bad requirements engineering is the cause in 71% of software project failures.

This paper considers a comparison study for ambiguity analysis applied to the requirements in a safety-critical application in the railway domain. The development of safety-critical systems, such as those used in the railway industry, must follow international standards, and they demand that requirements documents should be complete, clear, precise, unequivocal, verifiable, testable, maintainable, and feasible [2].

NLP4RE 2022: 5th Workshop on Natural Language Processing for Requirements Engineering @ REFSQ, Birmingham, UK (CEUR workshop): <https://nlp4re.github.io/2022/>

✉ aleksandar.bajceta@mdh.se (A. Bajceta); miguel.leonortiz@mdh.se (M. Leon); wasif.afzal@mdh.se (W. Afzal); pernilla.lindberg@alstomgroup.com (P. Lindberg); markus.bohlin@mdh.se (M. Bohlin)



© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

Most of the requirements documents are written in some form of natural language [3]. It is easy to write and understand natural language requirements, but they can be more ambiguous and error-prone. The ambiguous requirement could be understood differently between developers, testers, and other stakeholders involved in project development. The occurrence of underspecified or abstract requirements that can be interpreted differently is one of the most common requirements engineering problems [4]. This kind of miscommunication could lead to project delay or failure. Because of that, the identification of ambiguities in natural language requirements plays a vital role in requirements engineering.

A system requirement is ambiguous if there is more than one interpretation. Analyzing requirements for defects and ambiguities is a time-consuming process, and it can benefit from an automated solution. Using Natural Language Processing (NLP) for requirements analysis has been a popular research topic. Researchers created several tools for identifying ambiguities or defects in requirements using pattern or rule-based approaches [2, 5–10]. However, there is still a lack of industrial evaluation in using these tools in the research papers [11]. Our paper aims to fill the research gap by evaluating several requirements analysis tools used for ambiguity detection in natural language requirements in the railway domain.

Tools that we selected are Automated Requirements Measurement (ARM) [8], Quality Analyzer for Requirement Specifications (QuARS) [9], REquirements Template Analyzer (RETA) [5], and Requirements Complexity Measurement (RCM) [10]. We selected the tools based on suitability for the type of systems considered in this paper and their availability. The three first tools were publicly available for use by anyone, and the fourth tool was developed in-house at Alstom, and it is also publicly available¹. Furthermore, they all use a similar approach for identifying ambiguities, making them a good candidate for comparison studies. A sample set of 180 requirements were provided to us by our industrial partner, Alstom Transport AB in Sweden. These requirements are used in one of the Alstoms' projects for defining specifications of a train propulsion system. After running selected tools on this set of requirements, we compared the tool's ambiguity identification results with ambiguities found by an expert responsible for requirements engineering in Alstom.

The remainder of the paper is structured as follows. The related work is presented in Section 2. In Section 3, we described the tools used in the analysis. The followed methodology is described in Section 4. Results of the study are presented in Section 5. We addressed validity threats in Section 6. Section 7 concluded the paper.

2. Related Work

We identified several papers that compare requirements analysis tools, whose focus is on reducing ambiguities and improving overall requirements quality.

In [12] researchers compared their unnamed tool that was in development with two additional requirements engineering tools, reconstructed ARM tool [8] and TIGER Pro 2.0². The study was performed on a set of requirements from two projects. Researchers concluded that their unnamed new tool is still not good enough and that it is identifying many false positive

¹<https://github.com/eduardenoiu/NALABS/tree/master/RCM>

²<http://www.therightrequirement.com/TigerPro/TigerPro.html>

examples. A similar comparison study was performed in [13], where the authors presented their experience report using three different tools for requirement analysis. They compared two commercial tools, Requirement Scout and QVscribe, and one academic tool, QuARS. The study was performed by running the tools on the set of requirements from a simple E-shop software. The conclusions were that all three tools are performing quite similarly and that the two newer tools, Requirement Scout and QVscribe, did not outperform QuARS which was developed almost 20 years ago.

Different techniques, tools, and their technologies are reviewed and discussed in [14–17]. In these papers, researchers were reviewing the tools and technologies, and they were not experimenting on an actual set of requirements. In [14] researchers performed a literature review focusing on the tools for automatic detection of ambiguities in the requirements. They extracted 25 tools developed between 2008 and 2018. Ten of the most popular tools were compared with the addressed ambiguity, technologies used, and approaches in detecting ambiguities. In [15] researchers reviewed different techniques used for ambiguity detection in the NL requirements. They identified three approaches in detecting ambiguities: manual, semi-automatic using natural language processing techniques, and semi-automatic using machine learning techniques. They classified the detection of ambiguities with patterns as a Semi-Automatic approach using NLP. Similar studies were done in [16], [17]. A literature review was performed on detecting and resolving ambiguities in the NL requirements. Researchers compared tools and approaches by different parameters such as used technologies, targeting ambiguity, user interaction, etc.

In our work, we introduce two tools that were not compared in the literature. Also, tools are evaluated using requirements that were assessed and used in an industrial project.

3. Selected Tools

The availability of tools for requirements engineering is poor, and most of them are not publicly available [11]. Public versions of several tools do not contain all features, such as the set of patterns needed for the tool's full functionality. We were also unable to run specific tools in our environment. For our study, we selected four tools. The tools ARM and QuARS are among the oldest tools for requirement analysis. But they are still quite popular, and they built a foundation for further research and tools development. RETA is a bit newer tool, and comparison with older tools is relevant to us. The fourth tool selected for this study is RCM, a tool developed by Alstom, primarily to measure requirements complexity in Alstom's internal projects, but it can be used to detect ambiguities. The focus of these tools is to improve the quality, reduce complexity, and identify ambiguity in requirements. They automatically detect ambiguous requirements by comparing requirements with their patterns or rules.

3.1. Automated Requirements Measurement (ARM)

NASA developed the ARM tool in the 1990s. In 2011, it was discontinued, and for our study, we are using a reconstructed ARM tool developed by Carlson and Laplante [8]. The ARM tool has several quality indicators that are used for requirement analysis. Those quality indicators are imperatives, directives, continuances, size, specification depth, text structure, options, and weak phrases. Imperatives indicate absolute necessity using command words such as “shall” or

“must”. Continuances are words or phrases that follow imperative words, and their extensive use could increase requirement complexity. Examples of continuances are “following”, “as follows”, “and”, etc. Size includes counts of three indicators: total lines of text, the total number of imperative words, and the total number of subjects. Specification depth indicates how concise the document specifies the requirement by calculating the number of imperative statements at each level of the requirement document’s text structure. The text structure measures the number of indicators identified at each hierarchical level of the requirement document. Based on NASA’s requirements quality model, indicators of ambiguity are optional and weak phrases. The terms for options are “can”, “may”, and “optionally”. These terms loosen the specification, and they could allow the developer to judge what should be implemented in more than one way. Weak phrases terms are a set of words that could leave requirements with multiple or subjective interpretations. There are 12 pre-defined weak phrases in the ARM tool, and some of those terms are “adequate”, “be able to”, “timely”, “as appropriate”, etc.

3.2. Quality Analyzer for Requirement Specifications (QuARS)

QuARS is a requirement analysis tool that focuses on ambiguity detection developed by Lami et al. [9]. QuARS performs lexical and syntactic analysis to find ambiguities in natural language requirements. To perform lexical analysis, QuARS uses a set of patterns or keywords to identify optionality, subjectivity, vagueness, and weakness in the requirements. Optionality indicates that the requirement contains an optional part. Examples of terms used to identify optionality are “possibly”, “eventually”, “optionally” etc. Subjectivity represents the occurrence of terms that indicates personal opinion. For example, “having in mind” or “take into account”. Vagueness indicates non-quantifiable terms such as “significant” or “adequate”. Weakness indicates that the requirement does not have an imperative, and it is identified with terms such as “can”, “could”, “may” etc. The goal of syntactic analysis is to find patterns that could identify implicitness, multiplicity, and under-specification among requirements. Implicitness occurs when pronouns or other indirect references represent a subject or object in a sentence. Multiplicity occurs when the sentence has more than one main verb, subject, or object. Under-specification represents the occurrence of words that should be instantiated, for example, testing instead of functional testing or unit testing.

3.3. Requirements Template Analyzer (RETA)

RETA is a requirement analysis tool for automatically checking natural language requirements against templates for conformance [5]. The tool is implemented as a plugin for an open-source framework called GATE workbench³. The RETA tool supports analysis of the requirements against two templates, Rupp’s [18] and the EARS templates [19]. Templates represent a requirement structure, and when followed, it should reduce ambiguity in the requirement. Also, RETA tool checks for potentially dangerous terms and phrases that could cause ambiguity. These terms are looking for constructions such as passive voice, pronouns, quantifiers (terms used for quantification, such as “all”, “any”, “every”), and/or, vague terms, plural terms, etc.

³<https://gate.ac.uk/>

3.4. Requirements Complexity Measurement (RCM)

RCM is a tool developed in Alstom used for analyzing requirements complexity [10]. The tool's metrics for measuring complexity are the following: number of words, number of vague phrases, number of conjunctions, number of reference documents, optionality, subjectivity, weakness, automated readability index, imperatives, and continuances. The number of words is a metric used to measure the length of the requirement. The number of vague phrases represents the total number of terms that could cause problems in understanding the requirement, such as "adequate", "appropriate", "normal", etc. The number of conjunction counts the total number of phrases such as "and", "after", "although", etc. The number of reference documents indicates a need for additional reading to understand the requirement that contains references to other documents. Optionality metrics is implemented in the same way as in the ARM tool, and it uses keywords such as "can", "may", and "optionally" to identify it. Subjectivity indicates a personal feeling or opinion in requirement. To identify subjectivity, the tool uses keywords such as "similar", "better", "worse", etc. Weakness indicates phrases that could introduce uncertainty into requirements by using keywords such as "timely", "be able to", "be capable of", etc. The automated readability index is dependent on the total number of words in the requirement and the average number of letters per word. Imperatives indicate command words, and it is identified using keywords such as "shall", "must", "will", etc. Continuances are implemented in the same way as in the ARM tool. Measures selected to identify ambiguity among requirements are optionality, subjectivity, vague phrases, and weak phrases. We chose these indicators since Alstom's engineers identified them during tool development as rules that should be followed to avoid ambiguity in requirements.

4. Methodology

To evaluate selected tools, we used a dataset provided to us by our industrial partner Alstom. Alstom is one of the leaders in the railway industry and the largest railway company in Sweden. The dataset contains 180 requirements, written in the English language, for the train propulsion system development used in one of their previous projects. The average size of the requirement is 32.46 words. The shortest requirement contains 8 words, and the longest one has 205 words. Most of the requirements, or some parts of them, are written in passive form. Requirements were developed from the customer specifications, and they describe both hardware and software components of the system.

We used selected tools described in Section 3 to analyze the requirements and detect potential ambiguities in them. We compared results from tools' analysis with the analysis from Alstom's engineer, a person responsible for requirements engineering with years of experience in the railway domain. We asked the engineer to "Identify requirements in the dataset that have defects and could cause ambiguity among different stakeholders". If the requirement is potentially ambiguous, the engineer was supposed to mark it with "yes" and with "no" if it is not. We presented several examples of ambiguous sentences to the engineer. The ambiguous sentences were developed using definitions and examples from the Ambiguity Handbook [20].

- "I saw a man at the bank." - the bank can be a financial institution or ground bordering a

river

- “I saw Peter and Paul and Mary saw me.” - can be understood as I saw (Peter and Paul) and Mary saw me, or I saw Peter and (Paul and Mary) saw me
- “All linguists prefer a theory.” - can be understood as all linguists love the same one theory, or each linguist loves a, perhaps different, theory
- “The trucks shall treat the roads before they freeze.” - pronoun “they” can be either trucks or roads
- “Avoid long C functions.” - we do not know the value of “long”

Results from the engineer’s analysis are taken as a ground truth. If the requirement was marked as ambiguous by the engineer and it was identified by the tool, it was counted as a true positive result (TP). A false-positive result (FP) is if the requirement was marked as ambiguous by the tool but not by the engineer. A false-negative result (FN) is if the requirement was marked as ambiguous by the engineer but not identified as ambiguous by the tool. We used precision and recall as performance metrics to evaluate tools since it’s one of the most common metrics for tools’ evaluation. Precision and Recall for tools are calculated as $Precision = TP/(TP+FP)$, and $Recall = TP/(TP+FN)$.

5. Results

Results from the tools’ analysis are presented in Table 1. We group the results by tools’ quality indicators. If the tool does not support a specific category, the dash symbol “-“ is used in the table. The ARM tool identified 7 potential issues that fit the option quality indicator category and 10 for the weak phrase quality indicator. The same results for these two categories came from the RCM tool. This is expected since both tools use the same terms for these two categories. The RCM tool also identified 2 issues for subjectivity and 38 issues because of vague terms. Terms “can“ and “may“ are used by ARM and RCM to identify issues for option quality indicator, QuARS uses them to identify weakness in requirements, along with additional terms. QuARS identified 18 issues that could cause weakness in requirements, 5 issues for subjectivity, and 42 vagueness issues. QuARS identified 21 implicitness, 116 multiplicity, and 32 underspecification issues in syntactic analysis. RETA tool identified 56 issues with vague terms in requirements, 42 issues with quantifier terms, 52 pronouns, 66 complex requirements, and 51 adverbs. The RETA also uses patterns to identify plural terms and passive sentences. Because of it, almost all requirements were marked as potentially ambiguous. It is important to mention that multiple issues can be identified in one requirement.

In the Table 2 results of tool evaluation is presented. We compared tools’ analysis with the analysis performed by the engineer from Alstom, who identified 74 requirements as ambiguous. RCM achieved the highest precision in our analysis, 0.68. The rest of the tools achieved similar precision between each other. ARM and QuARS precision are 0.43, while RETA’s precision is 0.41. RETA also achieved the highest recall, 0.98, by identifying 177 requirements as potentially ambiguous. At first, this looks like significant results, since in literature recall is considered more important [21]. But it is essential to mention that this result was expected because RETA identifies plural words and passive sentences as an issue that could cause ambiguity. The second

Table 1

Ambiguity issues identified by each tool (“-” stands for not supported by the tool)

Category	ARM	QuARS	RETA	RCM
Option/Optionality	7	0	-	7
Weak Phrase/Weakness	10	18	-	10
Subjectivity	-	5	-	2
Vagueness/Vague Term	-	42	56	38
Quantifier	-	-	42	-
Pronoun	-	-	52	-
Plural	-	-	301	-
Passive	-	-	159	-
Complex	-	-	66	-
Adverb	-	-	51	-
And/OR	-	-	154	-
Implicitity	-	21	-	-
Multiplicity	-	116	-	-
Underspecificaiton	-	32	-	-

highest recall was achieved by QuARS 0.85, RCM follows it with 0.38, and ARM tool with only 0.08.

From the results, we can conclude that it's hard to achieve high precision in ambiguity detection using solutions based on pattern detection. Even though it is possible to achieve a very high recall with an extensive set of patterns, it will also result in many false-positive results. Similar results were also achieved in [7]. RCM tool achieved the highest precision, and this could be expected because it was developed inside Alstom.

Table 2

Summary of tool's analysis

	True Positive	False Positive	False Negative	Precision	Recall
ARM	6	8	68	0.43	0.08
QuARS	63	85	11	0.43	0.85
RETA	73	103	1	0.41	0.98
RCM	28	13	46	0.68	0.38

6. Threats to validity

In this section, we present the threats that could affect the validity of our results. According to [22], the threats to validity can be categorized as construct validity, internal validity, external validity, and reliability.

Construct validity refers to the extent to which studied operational measures represent what the researchers intended to study. The construct validity of our results is the number of tools selected for this paper. As mentioned in Section 3 many tools are not publicly available. To tackle potential threats, we've included all relevant tools that, to the best of our knowledge, were publicly available.

External validity refers to what extent results from the study can be generalized. The presented findings in this study are obtained from the analysis of system requirements used to develop an electric train propulsion system. The dataset we used for evaluation contains 180 requirements, which might not be enough to evaluate the general usage of selected tools. We do not claim

that these results can be generalized in other contexts. However, we believe that these findings could be helpful for the further evaluation or usage of these tools or to gain insights for future tool development.

Internal validity refers to what extent relations between investigated factors can affect the credibility of results. Assessing whether something is ambiguous or not is a subjective activity, and it depends on the analyst's experience and interpretation. Also, to determine the level of ambiguity, it is more common to analyze and compare answers from multiple analysts. To mitigate internal validity, we asked an expert in requirements engineering with years of experience working in the railway industry to analyze requirements and establish ground truth.

Reliability refers to what extent the data and the analysis depend on the researchers who performed the study. We asked an expert to analyze the system requirements to mitigate this threat. The expert was not included in the design of the study, or in performing the calculations of the results. Also, the tools were used after the expert finished the analysis.

7. Conclusions and Future work

We analyzed four different requirement analysis tools using 180 system requirements from a larger project in train propulsion design. The tools in the study all use a pattern-based approach to identify ambiguities in the requirements. Our analysis showed that it is possible to achieve high recall by following this approach, but increased precision is challenging. The RCM tool, developed inside Alstom, had the highest precision in our analysis. In comparison, the RETA tool achieved the highest recall.

A possible explanation is that domain knowledge plays an important role in identifying ambiguities. To the best of our knowledge, no papers have investigated how domain knowledge affects ambiguity identification. Therefore, how domain knowledge and experience affect identifying ambiguous requirements is the question that we would like to examine in the future. We plan to survey industry practitioners and academics to explore these questions. This kind of study would help us better understand the problem of ambiguity and what kind of ambiguity we should focus our research on in the future. We also aim to analyze other approaches and tools and identify what techniques can achieve the highest precision and a low number of false-positive cases in our context.

8. Acknowledgments

This research work has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 101007350. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Sweden, Austria, Czech Republic, Finland, France, Italy, Spain.

References

- [1] Infectra, Principles of requirements engineering or requirements management 101, 2022. URL: <https://www.infectra.com/ideas/whitepaper/>

- principles-of-requirements-engineering.aspx.
- [2] B. Rosadini, A. Ferrari, G. Gori, A. Fantechi, S. Gnesi, I. Trotta, S. Bacherini, Using NLP to detect requirements defects: An industrial experience in the railway domain, in: International Working Conference on Requirements Engineering: Foundation for Software Quality, Springer, 2017, pp. 344–360.
 - [3] M. Kassab, C. Neill, P. Laplante, State of practice in requirements engineering: contemporary data, *Innovations in Systems and Software Engineering* 10 (2014) 235–241.
 - [4] D. M. Fernández, S. Wagner, M. Kalinowski, M. Felderer, P. Mafra, A. Vetrò, T. Conte, M.-T. Christiansson, D. Greer, C. Lassenius, et al., Naming the pain in requirements engineering, *Empirical software engineering* 22 (2017) 2298–2338.
 - [5] C. Arora, M. Sabetzadeh, L. Briand, F. Zimmer, Automated checking of conformance to requirements templates using natural language processing, *IEEE transactions on Software Engineering* 41 (2015) 944–968.
 - [6] B. Gleich, O. Creighton, L. Kof, Ambiguity detection: Towards a tool explaining ambiguity sources, in: International Working Conference on Requirements Engineering: Foundation for Software Quality, Springer, 2010, pp. 218–232.
 - [7] S. F. Tjong, 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, pp. 80–95.
 - [8] N. Carlson, P. Laplante, The NASA automated requirements measurement tool: a reconstruction, *Innovations in Systems and Software Engineering* 10 (2014) 77–91.
 - [9] G. Lami, S. Gnesi, F. Fabbrini, M. Fusani, G. Trentanni, An automatic tool for the analysis of natural language requirements, *Informe técnico, CNR Information Science and Technology Institute, Pisa, Italia, Setiembre* (2004).
 - [10] K. Rajković, Measuring the complexity of natural language requirements in industrial control systems, 2019.
 - [11] L. Zhao, W. Alhoshan, A. Ferrari, K. J. Letsholo, M. Ajagbe, E.-V. Chioasca, R. T. Batista-Navarro, Natural language processing (NLP) for requirements engineering (RE): A systematic mapping study, *ACM Computing Surveys* (2020).
 - [12] A. Brooks, L. Krebs, B. Paulsen, Beta-testing a requirements analysis tool, *ACM SIGSOFT Software Engineering Notes* 39 (2014) 1–6.
 - [13] M. Arrabito, A. Fantechi, S. Gnesi, L. Semini, An experience with the application of three NLP tools for the analysis of natural language requirements, in: International Conference on the Quality of Information and Communications Technology, Springer, 2020, pp. 488–498.
 - [14] M. Q. Riaz, W. H. Butt, S. Rehman, Automatic detection of ambiguous software requirements: An insight, in: 2019 5th International Conference on Information Management (ICIM), IEEE, 2019, pp. 1–6.
 - [15] K. H. Oo, A. Nordin, A. R. Ismail, S. Sulaiman, An analysis of ambiguity detection techniques for software requirements specification (SRS), *International Journal of Engineering & Technology* 7 (2018) 501–505.
 - [16] A. Yadav, A. Patel, M. Shah, A comprehensive review on resolving ambiguities in natural language processing, *AI Open* 2 (2021) 85–92.
 - [17] U. S. Shah, D. C. Jinwala, Resolving ambiguities in natural language software requirements:

- a comprehensive survey, ACM SIGSOFT Software Engineering Notes 40 (2015) 1–7.
- [18] C. Rupp, K. Pohl, Requirements engineering fundamentals, Rocky Nook (2011).
 - [19] A. Mavin, P. Wilkinson, A. Harwood, M. Novak, Easy approach to requirements syntax (EARS), in: 2009 17th IEEE International Requirements Engineering Conference, IEEE, 2009, pp. 317–322.
 - [20] D. M. Berry, E. Kamsties, M. M. Krieger, From contract drafting to software specification: Linguistic sources of ambiguity, 2003. URL: <http://se.uwaterloo.ca/~dberry/handbook/ambiguityHandbook.pdf>.
 - [21] D. M. Berry, Empirical evaluation of tools for hairy requirements engineering tasks, Empirical Software Engineering 26 (2021) 1–77.
 - [22] P. Runeson, M. Höst, Guidelines for conducting and reporting case study research in software engineering, Empirical software engineering 14 (2009) 131–164.