2021 IEEE 29th International Requirements Engineering Conference (RE) | 978-1-6654-2856-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/RE51729.2021.00064

Enhancing NL Requirements Formalisation Using a Quality Checking Model

1st Mohamed Osama *, 2nd Aya Zaki-Ismail *, 3rd Mohamed Abdelrazek *, 4th John Grundy †, and 5th Amani Ibrahim *
Information Technology
* Deakin University , † Monash University
Melbourne, Australia
* {amohamedzakiism, mdarweish, mohamed.abdelrazek, amani.ibrahim}@deakin.edu.au , †<john.grundy@monash.edu>

***Abstract*—The formalisation of natural language (NL) requirements is a challenging problem because NL is inherently vague and imprecise. Existing formalisation approaches only support requirements adhering to specific boilerplates or templates, and are affected by the requirements quality issues. Several quality models are developed to assess the quality of NL requirements. However, they do not focus on the quality issues affecting the formalisability of requirements. Such issues can greatly compromise the operation of complex systems and even lead to catastrophic consequences or loss of life (in case of critical systems). In this paper, we propose a requirements quality checking approach utilising natural language processing (NLP) analysis. The approach assesses the quality of the requirements against a quality model that we developed to enhance the formalisability of NL requirements. We evaluate the effectiveness of our approach by comparing the formalisation efficiency of a recent automatic formalisation technique before and after utilising our approach. The results show an increase of approximately 15% in the F-measure (from 83.8% to 98%).**

***Index Terms*—Requirements specification, Requirements analysis, Quality analysis**

I. Introduction

Requirements quality issues affect the time and cost of the development life-cycle and can lead to catastrophic consequences in critical systems [1], [2]. Hence, several quality standards (e.g., [3]) mandate the incorporation of formal methods to detect issues at the early stages of development. However, formal methods operate on requirements expressed in formal notations only [4]. Nevertheless, the majority of requirements are specified in natural language (NL) [5], [6].

Formalising NL requirements is affected by their quality issues. Thus, identifying such issues and detecting them automatically, enhances the reliability and applicability of formalisation approaches. This enables the application of formal methods on a wider range of systems.

Informal quality checking aims to improve the quality of requirements and detect the quality issues. Several quality models have been proposed to assess the quality of requirements (e.g., [5], [7]). However, such models are not designed to enhance the success rate of automated formalisation approaches and are only thought of as an independent mean of quality checking.

We propose a quality checking tool to address the challenges of detecting quality issues affecting the formalisability of requirements. The intended users of this tool are the engineers contributing to the specification of requirements and/or the formalisation of system requirements. Additionally, the tool can be used by researchers and/or developers working on requirements formalisation approaches. We also developed a quality model focusing on the quality issues that can affect requirements formalisation. This should encourage more research targeting the development of NLP-based formalisation approaches (e.g., [8]) to support NL requirements with different structures, and enable reliable verification of the requirements of critical systems through formalisation. Further details about the tool is available here [1].

II. Quality-Checking Tool

To determine the quality metrics (affecting requirements formalisation) to be included in the proposed model, we analysed: (1) the existing informal quality checking approaches, and (2) the failure-reasons of the automated formalisation approaches. Then we designed a multi-pipeline tool, presented in Fig. 1, to support checking the included quality metrics.

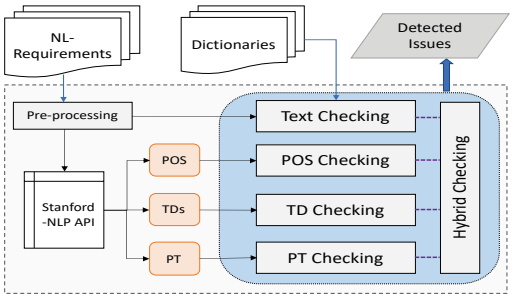

Fig. 1: NL-Requirements Quality Checking Tool

A. Tool Description

The tool consists of five main pipelines: (1) Text checking, (2) Part-of speech (POS) checking, (3) Typed-dependency (TDs) checking, (4) Parse-tree (PT) checking, and (5) Hybrid checking. The developed set of patterns [2] are specified

[1]Tool-Annex (Quality-Model & UI): https://github.com/ABC-7/QCModel/blob/main/QM_Annex.pdf

[2]Detection Patterns: https://github.com/ABC-7/QCModel/tree/main/Patterns

978-1-6654-2856-9/21/$31.00 ©2021 IEEE
DOI 10.1109/RE51729.2021.00064

according to the developed quality indicators in our model. The tool relies on the Stanford CoreNLP library for all the NLP-analysis performed.

1) Text Checking Pipeline: Detects quality issues whose indicators depend on the presence of a certain word or sequence of words. The issues are detected through a set of crafted regular expression patterns matching the requirements against dictionaries for the considered quality metrics. Each dictionary contains the indication words for only one quality metric. This facilitates the maintenance of the dictionaries for different domains.

2) Part-of-Speech Checking Pipeline: Identifies issues requiring the presence of a specific word-type attached with grammatical information. The POS tags for every word in a given requirement sentence are processed using the Stanford POS tagger. The obtained POS tags are then checked against the crafted regular expression patterns for the related quality metrics.

3) Typed-Dependency Checking Pipeline: This pipeline addresses the metrics relying on the syntactic relation between the words. The indicators of each quality metric are represented with regular expression patterns specified for TDs. Supplementary string matching operations may be applied after the patterns matching. For example, to properly identify multi-subjects, it is essential to verify that the extracted "Subj" mentions within the TDs, are related to the same verb (done through string comparison operations over the extracted TDs).

4) Parsing-Tree Checking Pipeline: This pipeline includes metrics that require certain syntactical structure information. For example, to detect syntactic ambiguity, we identify verb phrases comprising of multiple prepositional phrases. The regular expression patterns in this pipeline are specified on the PT.

5) Hybrid Checking: The pipelines are combined to support metrics that require checking multiple aspects of the syntactical information. The nature of the metric decides which pipelines are included in the checking. For example, in the "implicity" quality metric [9], the indication word "above" should be a modifier to a noun. This requires merging: 1) textual checking – ensuring the sentence contains the word "above"– and 2) TDs checking – ensuring the word "above" is a modifier to the main noun (e.g., the above threshold). Hence, false positives are avoided (e.g., "X is above 2" is avoided because it does not match the criteria).

III. Evaluation

We conducted two experiments recording the performance measures for the proposed tool. We also conducted a third impact evaluation experiment on the formalisation approach proposed in [8] for critical systems to show the gain from utilising the proposed tool. For the performance experiments, we used the requirements dataset used by the formalisation approach in [8]. The first experiment is applied on the requirements without correcting the incorrect grammar, while the grammar was corrected for the second experiment. Table I shows the performance measures (True Positive TP, False Positive FP, True Negative TN, and False Negative FN) for the quality model. our tool achieved very good accuracy (89% and 91%) in identifying the quality issues.

TABLE I: Quality Model Performance

Dataset	TP	FP	TN	FN	Recall	Precision	F-measure	Accuracy
Exp1	36	14	108	4	90%	72%	80%	89%
Exp2	36	14	112	0	100%	72%	83.7%	91%

Impact Evaluation We applied the RCM-Extractor approach on its dataset after manually fixing the issues identified by the tool. Table II shows that the extraction performance improved by 23%, 3%, 15% and 23% in recall, precision, F-measure and accuracy, respectively.

TABLE II: RCM-Extractor Performance

Dataset	TP	FP	FN	Recall	Precision	F-measure	Accuracy
Original-DS	122	7	40	75%	95%	83.8%	72%
Updated-DS	158	4	4	98%	98%	98%	95%

IV. Conclusion

In this paper, we proposed an automated tool to detect NL requirements quality-issues and support NLP-based formalisation approaches based on the proposed quality model. The evaluation shows that our approach achieved 89% accuracy in detecting requirements sentences with quality issues. In addition, it shows the effectiveness of the tool by improving the F-measure performance of the investigated formalisation approach from 83.8% to 98%.

References

[1] A. Zaki-Ismail., M. Osama., M. Abdelrazek., J. Grundy., and A. Ibrahim., "Rcm: Requirement capturing model for automated requirements formalisation," in Proceedings of the 9th MODELSWARD, INSTICC. SciTePress, 2021, pp. 110–121.

[2] M. Osama., A. Zaki-Ismail., M. Abdelrazek., J. Grundy., and A. Ibrahim., "Srcm: A semi formal requirements representation model enabling system visualisation and quality checking," in Proceedings of the 9th MODELSWARD, INSTICC. SciTePress, 2021, pp. 278–285.

[3] I. ISO, "26262: Road vehicles-functional safety," International Standard ISO/FDIS, vol. 26262, 2011.

[4] A. Zaki-Ismail., M. Osama., M. Abdelrazek., J. Grundy., and A. Ibrahim., "Requirements formality levels analysis and transformation of formal notations into semi-formal and informal notations," in Proceedings of the 33rd SEKE 2021, 2021.

[5] J. Kocerka, M. Krześlak, and A. Gałuszka, "Analysing quality of textual requirements using natural language processing: A literature review," in In 23rd MMAR. IEEE, 2018, pp. 876–880.

[6] M. Osama, A. Zaki-Ismail, M. Abdelrazek, J. Grundy, and A. Ibrahim, "Score-based automatic detection and resolution of syntactic ambiguity in natural language requirements," in In 36th ICSME. IEEE, 2020, pp. 651–661.

[7] R. Saavedra, L. C. Ballejos, and M. A. Ale, "Software requirements quality evaluation: State of the art and research challenges," in XIV Simposio Argentino de Ingeniería de Software (ASSE)-JAIIO 42 (2013), 2013.

[8] A. Zaki-Ismail., M. Osama., M. Abdelrazek., J. Grundy., and A. Ibrahim., "Rcm-extractor: Automated extraction of a semi formal representation model from natural language requirements," in Proceedings of the 9th MODELSWARD, INSTICC. SciTePress, 2021, pp. 270–277.

[9] G. Lami, S. Gnesi, F. Fabbrini, M. Fusani, and G. Trentanni, "An automatic tool for the analysis of natural language requirements," Informe técnico, CNR Information Science and Technology Institute, Pisa, Italia, Setiembre, 2004.