I. ANNEX

This section describes how this demo paper will be presented at the RE21 conference. For the Demos session, we will present the quality model behind the proposed checking tool and discuss how the checking tool works.

A. *Model*

The model is based on an in-depth analysis that we conducted on three perspectives. First, we examined the existing informal quality checking approaches to identify the suitable quality metrics within our scope (metrics that can be automated and enhance requirements formalisability). Second, we investigated the quality metrics proposed for requirements represented in other formats (e.g., Use Cases, User Stories), to identify any possible and/or generic quality metric that can be redefined for NL-requirements. Finally, we analysed the failed requirements in the extraction/formalisation conducted by a recent comprehensive NLP formalisation approach

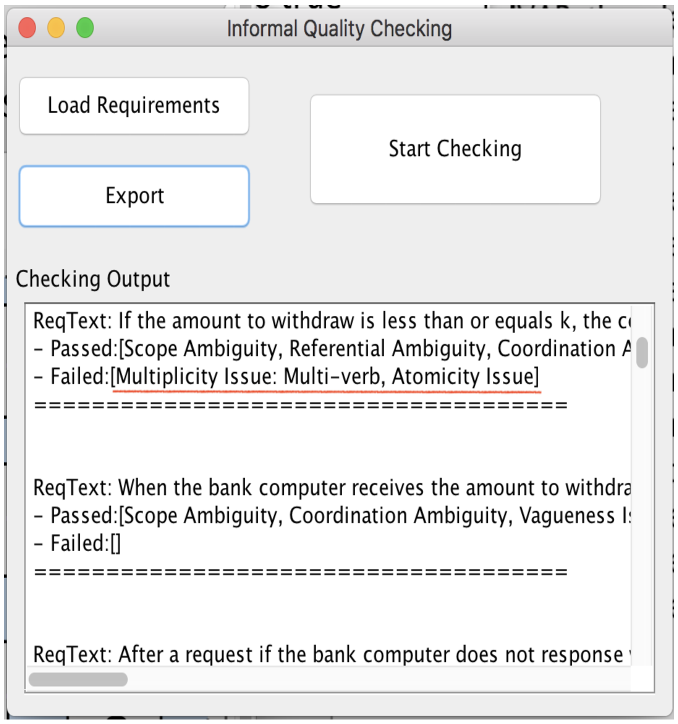
Table I outlines our developed indicators for each metric and the underlying checking to detect each of them. The first column lists the quality metrics. The second column provides the indicators of each metric. The third column states the status of these indicators (i.e., (1) "New": newly developed by us, (2) "Modified": the indicators themselves may be modified or the checking technique of the indicator(s) proposed by other researchers is modified, and (3) "Same": the indicators used without change). The table shows that (1) six metrics utilise completely new indicators, (2) six metrics use modified indicators (three extended (referential ambiguity, subjectivity and uncertainty), and three completely changed (Syntactic ambiguity, coordination ambiguity and atomicity)), and (3) the remaining eight metrics use the same indicators introduced in the literature. The final column highlights the checking pipeline assigned for each metric in compliance with our checking tool.

B. *Quality Checking Tool Interface*

The proposed quality checking tool is available online [1]. Figure 1 presents the user interface of the tool. The interface consists of three buttons for: loading requirements, applying the checking, and exporting the results. First, the user loads the requirements .txt file, where the requirements are separated with "%%%". Then, the "Start checking" button can be pressed to detect existing issues. After that, each requirement along side the passed metrics and the detected issues are listed in the checking output area as indicated in Figure 1. The user can export the output (each requirement attached with the passed metrics and detected issues) to an external file through the export button.

The dictionary folder of the tool contains a set of dictionaries serving the text checking pipeline of the checking approach. Figure 2.(a) shows the current supported dictionaries and Figure 2.(b) shows a sample dictionary, where each candidate word or sequence of words is written in a separate line. These files could be extended or updated by users. These dictionaries are loaded automatically into the tool.

[1]Quality Checking Tool: https://github.com/ABC-7/QCModel/tree/main/Tool

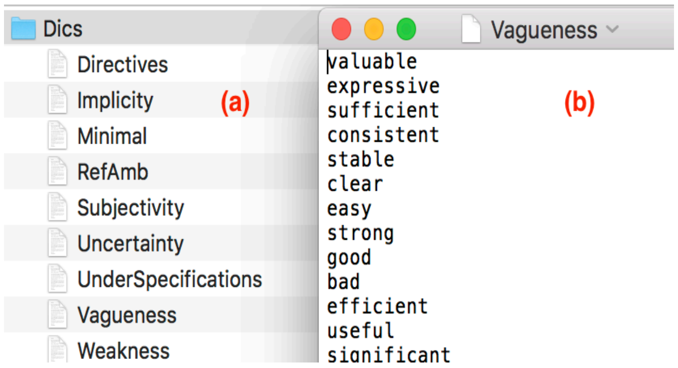


```
ReqText: If the amount to withdraw is less than or equals k, the c
- Passed:[Scope Ambiguity, Referential Ambiguity, Coordination A
- Failed:[Multiplicity Issue: Multi-verb, Atomicity Issue]
=====================================


ReqText: When the bank computer receives the amount to withdra
- Passed:[Scope Ambiguity, Coordination Ambiguity, Vagueness I
- Failed:[]
=====================================


ReqText: After a request if the bank computer does not response
```

Fig. 1: Quality Checking Tool Main View
Detected issues for the first requirements are underlined with red line



```
valuable
expressive
sufficient
consistent
stable
clear
easy
strong
good
bad
efficient
useful
significant
```

Fig. 2: Quality Checking Tool Main View
(a) Current Existing Dictionaries and (b) shows a sample dictionary

REFERENCES

[1] 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 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME). IEEE, 2020, pp. 651–661.

[2] B. Gleich, O. Creighton, and 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.

[3] G. Génova, J. M. Fuentes, J. Llorens, O. Hurtado, and V. Moreno, "A framework to measure and improve the quality of textual requirements," Requirements engineering, vol. 18, no. 1, pp. 25–41, 2013.

TABLE I: Proposed Quality Model Indicators Mapped to Metrics

Quality Metrics	**Quality Indicators**	**Status**	**Checking type**
Wrong interpretation	The underlying parsing tree is not marked as a sentence [1] or the used Typed dependencies contains unknown relation(s) (e.g., dep, X)	New	Hybrid of TDs and PT
Unintended Multi Object	Having direct and indirect objects without a preposition connector	New	TDs
Syntactic Ambiguity	-Identify if a given requirement sentence is vulnerable to have attachment by comprising prepositional phrase in the verb phrase.	Modified (in [2] apply Regex to detect revealing words from text)	PT
Scope Ambiguity	-Having a quantifier operator (e.g., all, each,every) in the given sentence followed by a reference (e.g., a, his, her, their, its)	Same in [2]	Text
Referential Ambiguity	-Having Pronoun/Administrative-Pronoun and having more than one possible refereed entity preceding the given pronoun	Modified ([2], [3], [4] check the existence of (administrative)pronoun only)	Hybrid of TD, POS and PT
Coordination Ambiguity	two types are considered: (1) Modifier distribution (e.g., project and election managers) and (2) brackets scope (e.g., Ali should come and Mona should come or John should come) to day) -Having at least one conjunction "and/or" between phrases of the same clause - Having Mixed conjunction "and" and "or" between clauses	Modified ([3] detect the existence of and/or in the sentence. [2] detect the existence of and/or together in the sentence)	PT
Vagueness	-Having one of the revealing words (e.g., close, clearly, sufficient, better, etc) with a confirmed type (e.g., the word close as verb is not ambiguous, but as adjective is ambiguous)	Same as in [4]. ([2] check either text or POS separately. [5] check against text only)	Hybrid of Text and POS
Implicitly	Having Pronoun/Administrative-Pronoun or Reveal-Words found in subject(previous, next, following,above, below, etc.	Same as [5]. [4] and [2] check against POS and text respectively.	Hybrid of Text and TDs
Multiplicity	Having Multiple Subject, object or verb	Same as [5]	TDs
Directives	Revealing words (e.g., figure, table, etc)	Same as [6]	Text
Weakness	revealing words (e.g., can, may, etc)	Same as [5]. [6] and [2] check against text only.	Text
Under-specification	Domain dependent words	Same as [5]	Hybrid of Text and TDs
Un-Explanation	Undefined domain dependent acronym/abbreviation	Same as [5], [3]	Text
Subjectivity	Revealing Words	Modified (in[5], [4] check against dictionary only	Hybrid of Text and POS
Uncertainty	Revealing Words	Modified (in[5], [4] check against dictionary only. [7] apply statistical approach succeeded by text checking against dictionary to catch the missed)	Hybrid of Text and POS
Atomicity	-Having a single "or" conjunction within or between a subordinating clauses of the same type (e.g., conditional clauses or time clauses) -Having non-finite clauses	Modified [3] check the existence of and/or in the sentence.	Hybrid of PT, TDs and Text
Minimal	Revealing words for consession and cause clauses (e.g., because, thus, etc) -Having appositive	New	Hybrid of Text and TDs
Too-Long	Percentage of dative construction	New	TDs
Too-Short	Ratio of entities to clauses	New	TDs

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

[5] 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.

[6] W. M. Wilson, L. H. Rosenberg, and L. E. Hyatt, “Automated analysis of requirement specifications,” in ICSE, vol. 97. Citeseer, 1997, pp. 161–171.

[7] H. Yang, A. D. Roeck, V. Gervasi, A. Willis, and B. Nuseibeh, “Speculative requirements: Automatic detection of uncertainty in natural language requirements.” IEEE, Sep. 05 2012, pp. 11–20.