Respecify: A requirements authoring tool harnessing CNL

Gavin Caird*, Stephen Boyd†, Michael Ledger‡, Didar Zowghi§ Muneera Bano¶
{∗gavin.caird, †stephen.boyd, ‡mike.ledger}@transportengineering.com.au
{§didar.zowghi, ¶muneera.bano}@uts.edu.au
Transport Engineering Pty Ltd
University of Technology Sydney

Abstract—This paper presents a novel framework for writing requirements specifications supported by a tool called Respecify which assists requirements engineers in identifying issues with ambiguity defects. The tool guides the requirements engineer to use a constrained natural language (CNL) which is structured to allow a requirements specification to be interpreted by the tool and rendered as several different diagrams which can expose a variety of defect types. The effectiveness of Respecify has been evaluated by conducting preliminary experiments in an industrial setting and the results have been promising.

I. INTRODUCTION

Ambiguity and incompleteness defects in requirements specifications are pervasive, difficult to prevent, and difficult to detect. If not detected during stakeholder review, these usually subtle defects can manifest themselves as costly errors in the final contract.

This paper proposes solutions to the problem of ambiguity and incompleteness in requirements specification, surveys existing solutions, and proposes a framework for a constrained natural language (CNL) based requirements authoring tool, called Respecify. The framework extends prior CNL study [16][12][25][20] by providing a interface for requirement generation, and alternative and elucidating perspectives of requirements specifications. We use a novel CNL-tailored parser generator that supports defining relationships, syntax that can be defined by the user, and syntactically correct auto-completions.

This paper outlines the perceived need that Respecify fulfills, its development process, its novel features, demonstrates on real world examples how the alternative views provided can aid in the detection of subtle ambiguity defects in sets of requirements, proposes an empirical study to quantify Respecify’s efficacy towards our hypothesis, and explores planned future work and possible future directions.

II. BACKGROUND

Transport Engineering Pty Ltd has over 15 years experience in developing requirements specifications for the defense and transportation industries.

A. Literature review

Requirements reviews have been in practice for more than four decades [3] and yet their application is still considered challenging today [4]. Improving the quality of requirements by making them complete and unambiguous is difficult at the review stage and therefore must be dealt with earlier [5].

Ambiguity is an inherent attribute of natural language (NL) and a necessary characteristic that makes NL adaptable in various contexts. It refers to the comprehension of text to hold multiple interpretations. As majority of the artefacts for requirements analysis are written in natural language [6], in requirements engineering (RE), ambiguity has thus long been recognized as a challenge [7]. In RE, ambiguity is defined as “having multiple interpretations despite the reader’s knowledge of the RE context” [8]. Ambiguity is considered to be more intractable than other requirements defects such as incompleteness and that it more frequently results in misinterpretation [9].

Ambiguity in natural language has been studied extensively in RE, especially in relation to its occurrence in written requirements. To avoid ambiguity, the use of formal languages, controlled natural languages, style guides, lexicons and other informal guidelines are often recommended. Some researchers used Natural Language Processing tools to address challenges of ambiguity in NL requirements (e.g. QuARS [10] and LOLITA [11]). Concerning the use of constrained natural languages, the EARS template [12] and the Rupp’s template [13] are well known constrained formats for editing requirements. Arora et al. [14] defined an approach to check the conformance of requirements to these templates. However, not much attention has been paid for empirical evaluation of these tools and techniques in industrial settings [15]. According to a systematic review on empirical studies on ambiguity in RE [15], out of 28 only 3 provided empirical evaluation of the proposed tools with controlled natural language [16], [17], [18].

B. Requirements quality defects; completeness and ambiguity

This paper focuses on six key types of ambiguity and incompleteness defects commonly found in transportation industry specifications and made by novice and experienced requirements engineers alike. The subtlety and nuance of these defects often leads to requirements engineers overlooking them, sometimes leading to costly variations at later stages of the project.

1) Unintended prescription: requirements engineers can unintentionally circumvent the design process by pre-
scribing subsystem solutions within system-level specifications. “The CCTV System shall enable the Guard to observe passengers boarding the Train” – if the intent is to allow the guard to observe boarding passengers, why must this be achieved by the CCTV system and not the train? Could the same outcome be achieved by providing clear sight lines from the guard’s compartment?

2) **Implicit object**: requirements often involve transitive verbs, which are those verbs that require an object so that the action is done to someone or something. That someone or something is often omitted by the analyst; perhaps it is implied. “The Supplier shall provide a System Verification Review certificate” – to whom?

3) **Ambiguous states and modes**: inconsistency in the use of states and modes in requirements, leading to ambiguity. Where state is omitted it is often assumed that this means the requirement is persistent. But this is often not the intention. “The Train shall provide Heating, Ventilation and Air Conditioning for all passenger areas” – is this continuous 24/7 or only when the train is in passenger service? What about when it is being cleaned or maintained?

4) **Do without undo**: we often see requirements that contain the action to do something (e.g. *open*, *raise*, *isolate*) without a requirement specifying the corresponding counter action to undo the original action (e.g. *close*, *lower*, *de-isolate*). “Each Train must enable the Crew to apply the Park Brakes from any Crew Cab at any time” – are the park brakes fixed once applied, or is there an additional need to release the park brakes? Under what conditions?

5) **Indefinite subject**: determiners in English precede a noun and include articles, demonstratives, quantifiers and possessives. We often see ambiguity created by incorrect use of determiners. “The Train shall not draw more than 3200 A from the overhead wire” – which Train? Suppose it is possible to amalgamate and divide trains into longer and shorter consist. Does this requirement apply equally to each possible train consist?

6) **Ambiguous nouns and verbs**: it is common to define special terms unique to the domain or project. It is uncommon to define terms that exist in standard dictionaries, yet they can have many meanings, leading to ambiguity.

**C. Motivation**

In each of the above examples 2-6, an obvious remedy is to add more information to the requirement (or its context). But in applying a philosophy of systematically removing defects, by directly adding contextual/qualifying information to each requirement, a significant burden is placed upon the requirements engineer that can decrease their subjective clarity and intuition over the specification. We do not want to annotate nearly every requirement with “when in Passenger Service”, despite that being strictly more correct. The added verbosity of doing so, ostensibly to increase its clarity, ironically makes it more likely for defects to occur unnoticed.

**III. Hypothesis**

Ambiguity and incompleteness defects may be more readily detected by requirements engineers, and accepted as genuine defects, when specifications are presented in novel alternative perspectives which accentuate the effect of the defect whilst masking details that are not relevant. These alternative perspectives can be generated through the use of a CNL.

**IV. Objective**

Our claim is that defects such as the examples given may be more readily detected by requirements engineers when specifications are constrained by a CNL and presented in a transformative way that accentuates the effect of the defect, without reducing the information content of a specification. The use of an authoring tool to automatically validate and evaluate the requirements specification as it is written allows ambiguity to be resolved at the time of writing instead of being remedied after stakeholder review. The purpose of this paper is then to build evidence toward the efficacy of Respecify and the associated techniques used to build it.

**V. Method**

The process of writing of requirements in the CNL we define, parsing of requirements using the CNL, and generation of alternative perspectives, is through the use of our requirements authoring tool Respecify. Using a bespoke diagram notation (based on dataflow diagrams, but with UML-like nested classes – see figure 4-5) we demonstrate real-world examples where ambiguity and incompleteness defects are made obvious through a visualization.

**A. Developing the supporting tool Respecify**

The method we implement is tool-oriented. We use the Agile[2] software development approach to rapidly prototype. There is a feedback loop between the development of the tool and CNL, and *Transport Engineering* requirements engineers; their feedback guides development, and development guides their feedback.

A custom parser generator[2] interprets this grammar to create a parser which is then used to parse requirements, give the user feedback (see Figure 1) and suggestions (see Figure 2) on how to make requirements compliant with the CNL, and generate relationship diagrams.

The current CNL in use in production has been stable since around November 2016, but recent work to support the generation of component and interface relationships from requirements specifications has lead to further extensions on the “stable” CNL.

In comparison to EARS[12] and Rupp’s template[13], we consider that requirements specification is facilitated by a tool that ensures CNL compliance (and provides other benefits) to

---

1n. “A set of railway vehicles forming a complete train”[21]

---

2A parser generator generates a parser from a file defining a grammar. Examples include Yacc[19], Happy[20]
be a key selling-point of Respecify. We believe that using a tool that guides its users can significantly lower the barrier to entry to use, and benefit from, CNL. Case studies demonstrating the success of using EARS in industry[12] illustrate the potential gain from using CNL alone. We want to build a “best of both worlds”, where a software tool eliminates the mental overhead of CNL compliance, so the requirements engineer benefits “for free”.

B. The Respecify CNL

The Respecify CNL is defined by a context-free grammar based on parts of speech. Its structure is similar to ISO/IEC guidelines [1]. We attempt to minimize linguistic sources of ambiguity. Notably, adjectives and adverbs are not defined within the CNL, as they are often sources of ambiguity[25].

In contrast to prior work[24][26][27], there is no attempt made to restrict ambiguity, as a separate phase apart from the CNL. Instead, our approach is for the CNL to limit possible ambiguities by construction.

A possible future direction is to analyze the requirement parse trees to identify ambiguities[24], based on heuristics for what constitutes ambiguity[22] within the CNL.

The novel features of Respecify are:-

1) Defined terms: Some key parts of the English lexicon are defined statically in the CNL determiners, prepositions, and auxiliary verbs are static lists included in the CNL grammar definition. Nouns, verbs, names of standards, and names of reference documents are not static; they belong to a user-defined glossary (see Figure 3) and must be added by the requirements engineer.

A limitation of the CNL’s support for user-defined terms as they are currently implemented is that there can only be one definition per defined term. This can be very inconvenient. However, for pronouns, this constraint eliminates the possibility of their causing anamorphic ambiguities[23].

In order for a requirement to be syntactically correct within the Respecify CNL, all nouns and verbs used within it must be defined within the specification. The tool assists in this process by offering definition candidates that would, if added, allow a requirement to parse. Figure 2 shows the suggestions generated for the word carriage, detected as a noun due to its position after the determiner “the”. Figure 3 shows a part of a definition glossary used in a real-world specification authored in Respecify.

In order for a requirement to be syntactically correct within the Respecify CNL, all nouns and verbs used within it must be defined within the specification. The tool assists in this process by offering definition candidates that would, if added, allow a requirement to parse. Figure 2 shows the suggestions generated for the word carriage, detected as a noun due to its position after the determiner “the”. Figure 3 shows a part of a definition glossary used in a real-world specification authored in Respecify.

In order for a requirement to be syntactically correct within the Respecify CNL, all nouns and verbs used within it must be defined within the specification. The tool assists in this process by offering definition candidates that would, if added, allow a requirement to parse. Figure 2 shows the suggestions generated for the word carriage, detected as a noun due to its position after the determiner “the”. Figure 3 shows a part of a definition glossary used in a real-world specification authored in Respecify.

We use a WordNet [28] search to find candidates.
The CNL defines 4 basic types of relationship – interface, component, mode, and compliance, and provides descriptions for how to transform sets of literals of each relationship type into diagrams.

Using literals brings interesting possibilities for consistency and correctness checking – but outside the scope of this paper. See the Future work section. We also do not yet attempt for relationships to deal with universal and existential quantification as in From contract drafting to software specification: Linguistic sources of ambiguity[25], where full predicate calculus interpretations of requirements, are given.

**Interface** A transactional relationship between two entities.

Interface literals come in the form \( \text{interface}(S, O, T) \) where \( O \) is thing being transacted (e.g. “Heating”), \( S \) is its source (e.g. “Train”), and \( T \) (e.g. “all passenger areas”) is its target. We use a subscript notation \( S_x \) if the transaction only occurs in some state/mode \( x \). Interfaces are detected based on the use of verbs provide, submit, transacts, gives etc. Interface literals \( \text{interface}(S, O, T) \) are visualized by drawing an edge with an arrow at its end, between nodes \( S \) and \( T \).

**Component** A component literal \( \text{component}(P, C) \) defines hierarchical composition between parent entity \( P \) and child entity \( C \). Component literals \( \text{component}(P, C) \) are visualized by drawing an edge with a line with a diamond on its end (à la UML) between nodes \( P \) and \( C \).

**Mode** A mode literal \( \text{hasMode}(O, M) \) defines that an entity \( O \) has the mode \( M \). Mode literals \( \text{hasMode}(O, M) \) are visualized by drawing an internal node labeled \( M \) inside the object with that mode: \( O \).

**Compliance** A compliance literal \( \text{compliance}(O, S) \) defines that an entity \( O \) must be in compliance with the standard \( S \). Compliance literals are drawn with a dashed line labeled "complies with" between nodes for \( O \) and \( S \).

Examples of train requirements and corresponding literals generated:-

- **Each Train shall have a Normal mode.** generates \( \text{hasMode}(\text{Train}, \text{Normal}) \)
- **Each Train shall provide Heating, Ventilation and Cooling for all passenger areas.** generates \( \text{interface}(\text{Train}, \text{Heating}, \text{all passenger areas}), \text{interface}(\text{Train}, \text{Ventilation}, \text{all passenger areas}), \text{and interface}(\text{Train}, \text{Cooling}, \text{all passenger areas}) \)
- **Each Train shall provide lighting for all passenger areas in Normal mode.** generates \( \text{interface}(\text{Train}_\text{Normal}, \text{lighting}, \text{all passenger areas}), \text{and hasMode}(\text{Train}, \text{Normal}) \) implicitly.

**C. Tool-assisted spotting of requirements quality defects**

Respecify aims to use literals generated from relationship descriptions to generate easy-to-understand graphs. The tool can analyze a requirements specification to generate diagrams, using the CNL and its ability to build relationships after parsing is completed. This will ensure that any defects in the visualization are reflected in defects in the specification – and are not simply mistakes made while drawing the diagram by hand.

**D. Validation steps**

1) **Ambiguity/incompleteness quality defect occurrence analysis:** We propose to conduct an empirical study that demonstrates how fewer requirements quality defects occur in a short (few than 100 requirements), but non-trivial specification, that is drafted with traditional methods, and with Respecify, by multiple requirements engineers.
We will use the requirements defects defined in the background section of this paper to classify types of requirements defects that occur, and use simple statistical methods to compare how the change in requirement authoring method affected the occurrence of the individual defect types.

Since the aggregation of requirements defects will be done by hand, the individual arguments that identify the found defects will need to be presented as part of the results.

2) Qualitative survey: We plan to conduct a survey of the opinions of the requirements engineers after having finalized their specifications. The survey would query for how the engineers felt about their work, about the process behind it, and their own judgment for the occurrences of defects in their work. Importantly we need to gauge from requirements engineers whether or not the alternative views provided by Respecify were rated, subjectively, as useful. We will assess the differences (if any) in specifications by engineers that said they found the graphs useful in comparison to engineers that did not.

Their subjective rating of quality may also be useful. If requirements engineers end up with a false sense of confidence having “Respecified” their specifications without producing a high quality result, this would be a negative outcome. With any tool there is a risk that the user will attempt to satisfy the tool rather than remain focused on the output.

VI. CURRENT RESULTS

A. Development progress

We have made substantial progress towards our research objective. The tool includes a CNL editor that enables us to iteratively refine the CNL. The parser checks each requirement for CNL compliance as the requirements engineer types, and syntax highlighting instantly flags any syntactic compliance issues, and makes suggestions to resolve them.

Once a requirement set exists that complies with the CNL, the user can view diagrams representing the formed relationships (compositional, interface, states etc.) and definitions perspectives.

B. Use of Respecify in industry so far

Though it is not yet available for public use, the tool has been used by over 12 users to generate some thousands of requirements in the transportation industry. As of March 2nd, 2017, there are over 20 non-trivial requirements specifications that have been (or partially) “Respecified”, spanning thousands of individual requirements across two organizations (Transport Engineering remains essentially the sole user, but there has been some initial exposure to outside organizations as well). The feedback from initial users has been positive with most being surprised that useful views can be automatically generated from text.

At this point, all feedback has been anecdotal, and as such cannot be the evidence upon which we base our claim. Specifications authored to date have been limited to public transportation conveyances (trains and ferries) and the requirements engineers using the tool are from the same company that is conducting this research – we are likely to be unabashedly biased when assessing our own work.

VII. FUTURE WORK

A. Tool development

The tool is still under active development, and the feature of generating relationship graphs is still immature. Greater expressiveness in the grammar language itself, to express more kinds of relationships in the CNL, is the immediate objective, as well as to develop the CNL using the existing features of the grammar to enable more kinds of relationships to be identified and generated as diagrams.

An exciting avenue of possible development is to leverage logic rules defining how literals should behave (for instance, that no entity can be a component of itself – \(\neg \exists x. \text{component}(x, x)\), or all entities in compliance with IFOOD101 must be charitable with sandwiches; \(\forall x. \text{complies}(x, \text{IFOOD101}) \rightarrow \forall p. (\text{person}(p) \rightarrow \text{interface}(x, \text{sandwich}, p))\) could allow for specifications to be checked for semantic correctness.

B. Industry validation

Preliminary qualitative surveys have been conducted. The next steps are to broaden the user base to experienced requirements engineers who are not employees of Transport Engineering, but who are within the transportation industry. We then intend to broaden the application to other industries such as defence and infrastructure. Once we are comfortable with the effectiveness of the tool we intend to make [https://respecify.com] publicly and freely available. In parallel with
this, we intend to survey the trains and ferries projects we have respecified to measure the impact of requirements defects in relation to comparable projects.

REFERENCES


