Extended Disambiguation Rules for Requirements Specifications

Sri Fatimah Tjong[1] (kcx4sfj@nottingham.edu.my),
Michael Hartley[1] (Michael.Hartley@nottingham.edu.my), and
Daniel M. Berry[2] (dberry@uwaterloo.ca)

[1] Faculty of Engineering and Computer Science,
University of Nottingham Malaysia Campus,
Jalan Broga, 43500 Semenyih, Selangor Darul Ehsan, Malaysia
[2] Cheriton School of Computer Science, University of Waterloo,
200 University Ave. West, Waterloo, Ontario, Canada N2L 3G1

Abstract. This paper extends earlier work by the authors in identifying guiding rules for natural language (NL) requirements specifications (RSs) by analysing a few sets of requirements documents from different domains. It presents guiding rules that help reduce ambiguities and imprecision in NL RSs. It validates these rules by applying them to sentences in several industrial strength NL RSs.

1 Introduction

In industrial requirements engineering (RE), regardless of the availability of various notations such as diagrams, formal notations or even pseudo-code, natural language (NL) is still the most frequently used representation in which to state requirements that are to be met by information technology (IT) products or services.

It is widely known that NL is inherently ambiguous and imprecise and so are requirements specifications (RSs) written in NL. To reduce the ambiguity and imprecision of NL RSs, several researchers have proposed the use of different modeling techniques and methods, as summarised by Denger, Jörg, and Kamsties in the QUASAR Report [1]. Others, e.g., Fuchs, Schwitter, and Schwertel [2, 3], concentrate on developing a controlled language for specifying requirements in an almost NL.

This work is an extension of work by the first author [4, 5] describing rules that guide a RS writer in writing less ambiguous and more precise RSs. A thorough discussion of the rules can be found in the first author's technical report on the subject [6]. This paper describes additional guiding rules. The old rules and the new rules are derived from an analysis of different RSs [7–10]. Some of the rules are derived from Denger's rules [11]. Still others are derived from confusions about RS statements that the authors have observed. It is important to provide rules for the kinds of ambiguities and imprecisions that exist in the real world.

Even though ambiguity and imprecision are different phenomena [12], this paper collapses both phenomena into one term, "ambiguity", since the distinction between the two do not effect the nature of the guiding rules.

Guiding rules aim to reduce ambiguity in writing a RS. Therefore, it is suggested that a RS writer consider these rules when writing any NL RS. Other, perhaps more beneficial, uses of the rules are in inspecting

- client-supplied pre-analysis requirements documents and
- post-analysis SRSs (software RSs)

for ambiguities. In the first case, a detected ambiguity should trigger a question to be asked of the client, particularly to avoid the natural subconscious disambiguation that occurs when one reads a document and thinks that she understands it because there appears to be nothing ambiguous about the document.

Section 2 reviews the past work. Section 3 describes the full set of guiding rules and suggested possible rewritings of potentially ambiguous requirements statements (RStats). Section 4 describes the validation of the guiding rules on Rstats in several industrial strength RSs. Finally, Section 5 summarises the paper and suggests future work.

2 Past Work

RE, being the core of software development, is concerned with identifying the purpose of a software system and the contexts in which it will be used. It facilitates effective communication of the requirements among different developers, users, and clients. However, there are times when these requirements are not properly communicated and documented, which results in incorrectness, inconsistency, incompleteness, and even misinterpretation.

To overcome this, there have been research works that define rules to limit the level of freedom in writing NL RSs. Macias and Pulman [13] apply domain-independent Natural Language Processing (NLP) techniques to control the production of NL requirements. Their study discusses how NLP techniques can help in the design of subsets of English grammar to limit the generation of ambiguous NL requirements.

Fuchs and Schwitter [2] describe a restrictive approach in their definition of a restricted NL called Attempt Controlled English (ACE). ACE uses a subset of English that is simple enough to avoid ambiguities, yet allows domain specialists to define requirements in NL with the rigour of formal specification languages.

Other works summarised by Denger, Jörg, and Kamsties [1] identify language indicators, sentence structures, and rules that assist the RS reader in detecting ambiguous RStats in NL RSs. The focus of their rules is RSs in the automotive domain.

3 Guiding Rules

The guiding rules aim to avoid the introduction of ambiguities when writing any RStat. These rules are intended to be used along with language patterns [11, 4, 5] in order to reduce ambiguities in the writing of any RStat. In addition, the guiding rules can also be used to help find ambiguities in an existing RS.

In the rules and in examples, text from a RStat is typeset in a sansserif typeface. A constant is such text is typeset in an upright sansserif typeface, and a variable in such

text to be replaced by constant text of the variable's type is typeset in an oblique (a.k.a., slanted) sansserif typeface.

Note that we try to obey our own rules but cannot completely, because these rules cannot be blindly enforced requirements. They *should* be followed, but there exists many a circumstance in which a rule cannot and should not be followed.

The unit of application of each of most rules is a single Rstat S. Each rule that says to avoid a construction offers an alternative construction for saying the same thing less ambiguously; the alternative construction is signalled by "Instead,".

3.1 Old Rules

The first fifteen rules are from the first author's early work [4, 5]. Since these rules are motivated and described in detail elsewhere, they are mostly only listed here. An example is given only if the rule would be incomprehensible without an example.

Rule 1: S should be written as a simple affirmative declarative sentence that has only one main verb.

Rule 2: Avoid writing S in passive voice, especially in which no doer of the action is specified. Instead, write S in active voice, with the doer of the action as the subject of S.

Rule 3: Avoid writing S of the form There is X in Y. or X exists in Y.. Instead, write Y has X..

Rule 4: Avoid writing S containing a subjective option introduced by a keyword such as either, whether, otherwise, velc[1]. An example is The user shall either be trusted or not trusted. Instead, specify under what condition each option happens.

Rule 5: Avoid writing S containing an indefinite timing introduced by the keyword eventually, at last, velc. Instead, specify strict sequencing of events with no timing or specify timing with tolerances, both in measurable units.

Rule 6a: Avoid writing S containing a noun phrase containing maximum or minimum as an adjective modifying the main noun, e.g. The system shall return maximum results. Instead, replace the adjective with a more detailed, complete characterisation of the noun.

Rule 6b: Avoid writing S containing the phrase as much as possible or as little as possible. Instead, replace the phrase with a more detailed, complete characterisation of what is as much or as little as possible.

Rule 7: Avoid writing S containing both X and Y. Instead, write only X and Y.

Rule 8: Avoid writing S containing X but Y. Instead, write X and Y.

Rule 10: Avoid writing S containing X and/or Y. Instead, write X, Y, or both.[2]

Rule 9: Avoid writing S containing X/Y. Instead, write X or Y.

Rule 11: Avoid writing S containing any and equivalent, e.g., not only, but also, as well as, etc., that provides additional commentary. Instead write simply and.

[1] "velc." ("vel cetera") is to "or" as "etc." ("et cetera") is to "and".

[2] This use of both is not excluded by Rule 10, which suggests avoiding both when it is combined with a following and.

Rule 12a: Avoid writing S containing any pair of parentheses, braces, or brackets, i.e., ( ), { }, or [ ], that encloses unnecessary text. Instead, remove the unnecessary text and the enclosing pair of parentheses, braces, or brackets.

Rule 12b: Avoid writing S containing any pair of parentheses, braces, or brackets, i.e., ( ), { }, or [ ], that encloses necessary text. Instead, move the necessary text to its own Rstat, and remove pair of parentheses, braces, or brackets.

Rule 12c: Avoid writing S containing any pair of parentheses, braces, or brackets, i.e., ( ), { }, or [ ], in which the purpose of the pair of parentheses, braces, or brackets is to cause S to mean two or more Rstats, e.g., Turning the switch down (up) turns the light on (off). Instead, rewrite S as a sequence of as many Rstats that S means.

Rule 13: Provide a glossary to explain each domain-specific term or nominalisation, velc. that appears in the RS.

Rule 14: Provide an acronym list to explain each acronym that appears in the RS.

Rule 15: Provide an abbreviation list to explain each abbreviation that appears in the RS.

Because in some cases, what results after rewriting a Rstat is not what the writer intended, the stakeholder who owns a rewritten Rstat must be asked if the new Rstat is what he or she intended. For example, even though / means or, an occasional writer or reader believes that / means and, and he or she would be surprised when presented with a / replaced by an or.

3.2 New Rules

The following rules are from the first author's latest work.

Rule 16: Avoid writing S containing all or any modifying a direct object, e.g.:

> **E1:** The operator log will record all warning messages prompted by the system.

or

> **E2:** The operator log will record any warning messages prompted by the system.

The use of all forces the use of plural which is ambiguous in its own right, as suggested by Rule 25, and is to be avoided. The use of any is confusing, because any can be interpreted as an existential quantifier instead of the desired universal quantifier. Instead, write each in place of all or any e.g.:

> **E3:** The operator log will record each warning message prompted by the system.

However, not every instance of all should be replaced by each, e.g.:

> **E4:** The system must put all displayed text into one file, in order to facilitate software maintenance for developers and to ease future translations to local languages.

The difference between the use of all in E4 and the use of all in E1 is that in E4, the intention is to specify something that happens to the entire set of displayed texts, while in E1, the intention is to specify something that happens to each element of the set of warning messages. It is hard to describe this difference in a rule.

Rule 17: Avoid writing S containing some, many, few, e.g., velc. to describe a set of objects by example rather than by describing the set itself, e.g.:

> **E5:** Some of the software packages (e.g. each HLT algorithm, the selection control, the data access) shall be documented for both the user, developer, and maintainer.

Note that E5 violates also some other rules, i.e., Rules 7 and 12b. Instead, specify the specific instances that are supposed to be in the set, e.g.:

> **E6:** Each HLT algorithm, the selection control, and the data access shall be documented for the user, developer, and maintainer.

Rule 18: Avoid writing S containing any of meanwhile, whereas, on the other hand, velc. Each such phrase is usually used to combine two or more related Rstats. Each should be avoided as unnecessarily complicating or lengthening the containing RS without providing any essential information.

> **E7:** Each officer can print the report by selecting an associate. Meanwhile, an associate can only view the report which contains the payment details entered by the associate himself.

Instead, rewrite S without the meanwhile, whereas, on the other hand, velc., e.g.:

> **E8:** Each officer can print the report by selecting an associate.
> An associate can view only the report that contains the payment details entered by the associate himself.

Note that E7 has also a misplaced only that is moved to the correct place in E8 according to Rule 27. Moreover, the which is changed to that in accordance with English rules.

Rule 19: Avoid writing S containing any vague adjective such as prompt, fast, routine, velc. to describe the timing of a process, e.g.:

> **E9:** The Science Analysis Software performs prompt processing of Level 0 data to produce Level 1 event data.

Instead, replace the vague adjective with an actual amount of time in a measurable time unit, e.g.:

> **E10:** The Science Analysis Software performs within 0.1 seconds the processing of Level 0 data to produce Level 1 event data.

Rule 20: When between or among is used in S to differentiate one action or process from another action or process described in the same RS, then S should not be changed by any of the other rules. For example,

> **E11:** Relationships between objects made at LVL2 must still be valid if the objects are passed on to the EF or are stored and retrieved in the offline environment.

should not be changed by any other rule. This rule prevents upsetting any relationships that exist between the pair of actions or processes.

Rule 21: Avoid writing S containing any vague adjectives such as ancillary, relevant, routine, velc. that requires the reader to do his or her own requirements analysis to make S a complete Rstat, e.g.:

E12: In support of high-level processing, the SAS extracts from the LAT and SC Level 0 data ancillary information relevant to event reconstruction and classification.

Instead, the adjective should be replaced by a complete description of whatever is ancillary, relevant, routine, velc., e.g.:

E13: In support of high-level processing, the SAS extracts the Ground Observational Data from the LAT and SC Level 0.

Determination of the complete description may require consulting the client or other stakeholders. Finally, notice that the word routine is a signal for two different rules, Rules 19 and 21; vagueness has multiple uses.

Rule 22: Avoid writing S containing common, generic, customary, velc., e.g.:

E14: The simulation shall use instrument geometry that is defined and is common to all analysis modules.

Each of these words has more than one scope. For example, in E14, it is difficult to know if the instrument geometry is that which is known world wide in any analysis module or is that which is assumed in the specific analysis modules appearing in the system being specified by the RS. Instead, it is necessary to describe the scope of the commonality, genericity, customariness, velc.

Rule 23: Dependent Rstats should be grouped together, e.g.:

E15: The SDP shall provide the Level 1 data to the P1 sites. The Level 1 data shall arrive at the sites no later than 24 hours after completion of processing in the SDP. Then, the SDP shall provide the Level 0 data to the P1 sites.

Rule 24: Avoid writing S containing should and similar words, except as an expression of a non-functional requirement. If S is supposed to be a functional requirement, then rewrite S using shall.

Rule 25: Avoid writing S containing a plural subject, e.g.:

E16: All persons in the room lift a table.

With such a sentence, it is difficult to determine how many predicate or object instance is related to each subject instance. In E16, it cannot be determined if each person in the room lifts his or her own table or if the all the people in the room as a group lift one table. Instead, try to use only a singular subject, e.g:

E17: Each person in the room lifts his or her own table.

and

E18: The set of all person in the room lifts one table.

If you must use a plural subject, then reserve it for describing properties of the entire set of subject instances, e.g.:

E19: All persons in the room together lift one table.

Rule 26: Avoid writing S containing *A* unless *B*. Instead, use If (*B*), then *A*. We have seen evidence that an occasional person uses *A* unless *B*. as (*B*) if and only if *A*. Writing an "unless" Rstat as its logical equivalent will force the person who misinterprets the "unless" Rstat to see what the Rstat really means.

Rule 27: Move any only, also, or any other limiting word to before the phrase the only, also, or other limiting word is intended to limit. For example

E20: An associate can only view the report which contains the payment details entered by the associate himself.

E21: An associate can view only the report which contains the payment details entered by the associate himself.

We expect the guiding rules to assist a requirements engineer in inspecting and in writing a NL RS.

4 Validation: Analysis and Rewriting the Original Requirements

We validated the guiding rules by rewriting existing and ambiguous RSs [7–10] using recommendations from the guiding rules. Before we rewrote any RS, each RStat in the RS was examined with the help of the guiding rules in order to identify possible ambiguities in the RStat. Whenever a RStat violated one or more rules, we looked carefully at the Rstat in order to determine if it was indeed ambiguous. If a Rstat was judged to be ambiguous, the suggestions of the violated rules were followed to guide the rewriting of Rstat.

Space permits showing only a few of the ambiguous Rstats that we found. However, note that most of the examples cited in the explanations of the guiding rules are from the examined RSs.

In E22, meanwhile combines two Rstats into one long Rstat, in violation of Rule 18. Even though the second Rstat has a plural subject and uses all in apparent violation of Rule 25, the Rstat is describing a property of the entire set of payments, that they are grouped together into one payment.

E22: If the payment is with payee's details, then the system will treat each payment separately meanwhile if users choose "No", all the payment records will be grouped together to become one cheque.

Therefore, the suggested change of only Rule 18 is applied to split E22 into two Rstats, E23 and E24.

E23: If the payment is with the payee's detail, then the system will treat each payment separately.

E24: If the user chooses "No", all the payment records will be grouped together to become one cheque.

E25 and E26 show the ambiguity resulting from the use of all in writing a plural subject, in violation of Rule 25. Note that E25 has (1) a violation of Rule 12b, against the use of a pair of parentheses to enclose essential information and (2) a violation of Rule 17, against the use of e.g. to describe example elements of a set of objects instead of describing the set.

E25: All login attempts shall be done so in a secure manner (e.g. encrypted passwords).

E26: All pipeline products shall contain keywords, which describe the pipeline modules used to create them.

The violated rules suggest rewriting E25 and E26 into E27 and E28, respectively.

E27: Every login attempt shall be done with an encrypted password.

E28: Every pipeline product shall contain keywords that describe the pipeline modules used to create the pipeline product.

The change embodied in E28 assumes that each pipeline product is built from several pipeline modules. If each pipeline product is built from exactly one pipeline module, then E26 should be changed to E29.

E29: Every pipeline product shall contain the keyword that describes the pipeline module used to create the pipeline product.

E30 contains a violation of each of Rule 16, Rule 25, and Rule 12a or Rule 17.

E30: All mission elements shall withstand all environments (e.g. EMI, shock, and thermal) to be encountered from component fabrication.

If EMI, shock, and thermal are only some of the possible environments that can be encountered during component fabrication, then a suggested rewriting of E30 is E31.

E31: Each mission element shall withstand each environment that can be encountered during component fabrication.

If, on the other hand, EMI, shock, and thermal are all of the possible environments that can be encountered during component fabrication, then an alternative suggested rewriting of E30 is E32.

E32: Each mission element shall withstand EMI, shock, and thermal environments.

E33 contains violations of Rule 12b and Rule 17 or a violation of Rule12a.

E33: All users of the system shall login using some form of unique identification (e.g. username and password).

If the purpose of the information in the pair of parentheses is to give the only form of unique identification possible, then a suggested rewriting is E34.

E34: Each user of the system shall login by using his username and his password.

If the purpose of the information in the pair of parentheses is to give one possible form of unique identification, and it is truly the case that *any* form of unique identification is to be used for login, then a suggested rewriting is E35.

E35: Each user of the system shall login by using some form of unique identification.

E36 violates Rule 17 or Rule 21 because the word routine requires the reader to do requirements analysis to determine what sort of processing is really intended. Moreover, it is not clear if the missing information is timing or functional information.

> **E36:** The SAS is responsible for routine Level 2 processing of the LAT data.

If the missing information is about the timing of the processing, then a suggested rewriting is E37.

> **E37:** The SAS is responsible for daily Level 2 processing of the LAT data.

If the missing information is about the function of the processing, then a suggested rewriting is E38.

> **E38:** The SAS is responsible for the Level 2 processing of the LAT data that computes the maximum, minimum, and average values.

E39 gives example Rstats that together fall under the province of Rule 23 and should be grouped together.

> **E39:** The system shall be designed to accommodate the addition of a propulsion subsystem. The propulsion subsystem shall be capable of transferring the system from the circular parking orbit to the operational orbit.

E40 contains violations of Rules 18 and 24 by its use of should and of whereas.

> **E40:** The user manual should document the expect results whereas the user interface should provide information or warning indicating what changes will occur when a user changes the regional setting.

The violated rules suggest rewriting E40 as E41 and E42.

> **E41:** The user manual shall document the expect results.

> **E42:** The user interface shall provide information or a warning indicating what changes will occur when a user changes the regional setting.

The main lesson to learn from these examples is that while guiding rules help identify which Rstats are potentially ambiguous, only a human being can determine *if* any Rstat is really ambiguous, and only a *stakeholder human being* can explain the intended meaning of an ambiguous Rstat so that the Rstat can be rewritten correctly.

5 Conclusion and Future Work

This paper describes the latest guiding rules for avoiding ambiguities in NL RSs that we have found based on examination of several industrial strength RSs. As a partial

validation of the new rules, the paper gives some examples of ambiguous sentences from the RSs and their rewritten, less ambiguous forms.

We expect to continue to examine industrial strength RSs to find additional rules. In addition, the first author is developing a Systemised Requirements Engineering Environment (SREE) that searches for potentially ambiguous Rstats and offers suggestions for rewriting each potentially ambiguous Rstat it finds. The effectiveness of SREE will be validated by applying it industrial strength RSs.

The lack of uniformity and the hit-and-miss nature of the guiding rules are a bit disconcerting. However, these guiding rules cover the kinds of ambiguities we have found in actual industrial RSs. Of course, the method by which the guiding rules are found makes it difficult to assess when enough rules have been found. Probably, there is no limit on the number of rules. However, we expect that at some point, the rate of addition of new rules will drop off considerably, just because we will eventually begin not to find new kinds of ambiguities. Thus, the work described in this paper is complementary to all the other work cited in Section 2 that attempts to find systematic ways of detecting or avoiding ambiguities.

Another disconcerting property of these rules is the difficulty of finding a pattern for each of these ambiguities. For any rule, there is no guarantee that every Rstat meeting the pattern of the rule is an instance of the kind of ambiguity that is intended to be described by the rule; and conversely, there is no guarantee that the rule describes every instance of the kind of ambiguity that is intended to be described by the rule. As SREE is developed, and we see its recall and precision in identifying potentially ambiguous RStats, we will be able to refine the patterns.

References

1. Denger, C., Jörg, D., Kamsties, E.: QUASAR: A Survey on Approaches for Writing Precise Natural Language Requirements. IESE Fraunhofer, Kaiserslautern, DE (2001)
2. Fuchs, N.E., Schwitter, R.: Specifying logic programs in controlled natural language. In: CLNLP'95, Workshop on Computational Logic for Natural language Processing. (1995)
3. Schwertel, U.: Controlling plural ambiguities in Attempto Controlled English. In: Proceedings of the Third Internaltional Workshop on Controlled Language Applications (CLAW), Seattle, WA, USA (2000)
4. Tjong, S.F.: Improving the quality of natural language requirements specifications through natural language requirements patterns. Technical report, Faculty of Engineering and Computer Sciences, University of Nottingham (2006) `http://sepang.nottingham.edu.my/~kcx4sfj/`.
5. Tjong, S.F.: Improving the quality of natural language requirements specifications through natural language requirements patterns. In: IEEE International Conference on Computer and Information Technology. (2006)
6. Tjong, S.F.: Elaborated natural language patterns for requirements specifications. Technical report, Faculty of Engineering and Computer Sciences, University of Nottingham (2006) `http://sepang.nottingham.edu.my/~kcx4sfj/`.
7. Moeser, R., Perley, P.: EVLA operations interface, software requirements. Technical report, EVLA-SW-003 Revision: 2.5 (2003) `http://www.aoc.nrao.edu/evla/techdocs/computer/workdocs/array-sw-rqmts.%pdf`.

8. Dubois, R.: Large area telescope (lat) science analysis software specification. Technical report, GE-0000X-DO (2000) http://www-glast.slac.stanford.edu/IntegrationTest/DataHandling/docs/LA%T-SS-00020-06.pdf.
9. George, S.: PESA high-level trigger selection software requirements. Technical report, Centre for Particle Physics at Royal Holloway University (2001) http://www.pp.rhul.ac.uk/atlas/newsw/requirements/1.0.2/.
10. Eng, C.S.: Batch poster system, detailed business requirements. Technical report, EDS MySC, Malaysia (2005)
11. Denger, C.: High quality requirements specifications for embedded systems through authoring rules and language patterns. Technical Report M. Sc. Thesis, Fachbereich Informatik, Universität Kaiserslautern (2002)
12. Bach, K.: Ambiguity. In Craig, E., Floridi, L., eds.: Routledge Encyclopedia of Philosophy, London, UK, Routledge (1998)
13. Macias, B., Pulman, S.: Natural language processing for requirements specifications. In Redmill, F., Anderson, T., eds.: Safety-critical systems: Current issues, techniques and standards, London, UK, Chapman & Hall (1993) 67–89