Extended Disambiguation Rules for Requirements Specifications

Sri Fatimah Tjong[1], Michael Hartley[1], Daniel M. Berry[2]

[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

{kcx4sfj,Michael.Hartley}@nottingham.edu.my, dberry@uwaterloo.ca

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. As described in Section 2, several researchers have proposed several methods for reducing the ambiguity and imprecision of NL RSs.

This work is an extension of work by the first author [20, 21] describing rules that guide a RS writer in writing less ambiguous and more precise RSs. A thorough discussion of these rules can be found in the first author's technical report on the subject [19]. This paper describes additional guiding rules. The old rules and the new rules are derived from an analysis of different RSs [16, 6, 11, 7]. Some of the rules are derived from Denger's rules [4]. 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 [1], 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 [10] 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 problem, some have defined rules to limit the level of freedom in writing NL RSs. Rupp and Götz [17] develop a set of rules that detect defects, ambiguities, and weak phrases in RSs. They distinguish three main language transformations, deletion, generalisation, and distortion, that help expose defects in RSs.

Juristo *et al.* [13] classify each requirement as static or dynamic. They show how to recast any static requirements into the structure of the Static Utility Language (SUL) and how to recast any dynamic requirements into the structure of Dynamic Utility Language (DUL). Each of SUL and DUL is specified by a formal grammar and is composed of several natural language structures each of which can be translated into predicate logic. Therefore every utterance in either language is not ambiguous.

Macias and Pulman [15] apply domain-independent Natural Language Processing (NLP) techniques to control the production of NL requirements. Their study shows how NLP techniques can help in the design of subgrammar of an English grammar to limit the generation of ambiguous NL Rstats.

Fuchs, Schwitter, and Schwertel [9, 18], describe a restrictive approach in their definition of a restricted NL called Attempto Controlled English (ACE). ACE uses a sublanguage of English that is simple enough to avoid ambiguities, yet allows a domain specialist to define requirements in NL with the rigour possible with a formal specification language.

Works summarised by Denger, Jörg, and Kamsties [5] 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.

Another approach is to provide guidelines for writing good RSs. Hooks [12] describes common problems found RSs and suggests guidelines that help avoid them. Similarly, Firesmith [8] describes both characteristics of a good RS and potential problems that occur in writing a RS.

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 [4, 20, 21] 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. The reader should pay attention to the typeface of any punctuation at the end of any such sansserif snippet to determine if the punctuation is part of the snippet or part of the surrounding explanatory text. When there are two adjacent punctuation symbols, usually, the first belongs to the snippet ending with the symbols and the second belongs to the explanatory text surrounding the snippet.

We try to obey our own rules but cannot completely, because these rules cannot be blindly enforced. They *should* be followed, but there exists many a circumstance in which a rule cannot and should not be followed. Describing these exceptional circumstances systematically is extremely difficult, if not impossible, and is a research topic all its own.

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 [20, 21]. They are numbered as before even though the ordering has been changed, and one rule has been split into three to bring out its special cases. 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, etc. 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.[1] In-

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

stead, 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.

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.

[2]This use of both is not excluded by Rule 7, which suggests avoiding both when it is combined with a following and.

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.

Note that Rule 10 should be applied before Rule 9.

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 she intended. For example, even though "/" means or, an occasional writer or reader believes that "/" means and, and she would be surprised when presented with a "/" replaced by an or.

3.2 New Rules

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

Rule 16: Avoid writing S containing all, any, or both modifying a direct object when the intent of S is to describe what happens to each instance of the set that is described by the modified 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 or both is confusing because it is hard to distinguish whether the action happens to the whole set or to each element of the set. See Rule 25 for a similar difficulty. The use of any is confusing, because any can be interpreted as an existential quantifier instead of the desired universal quantifier. Instead, write each[3] in place of all, any, or both e.g.:

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

However, not every instance of all or both should be replaced by each. When the intent of S is to describe something that happens to the entire set which is the direct object, use all or both, e.g.:

[3]Some have wondered whether every should or could be used instead of each. When either is used as an adjective of a noun, each and every appear to interchangeable, as each is singular. However, there is evidence that each and every differ in meaning [14]. Resolution of this issue is among our future work.

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.

Rule 17: Avoid writing S containing some, many, few, for example, e.g., velc. to describe a set of objects by example rather than by describing the set itself, e.g.:[4]

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.

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.

E5 violates also some other rules, i.e., Rules 7 and 12b. Therefore, the modification of E5 to get E6 took into account also the recommendations of these other rules.

Note that neither that is nor i.e. is among the items to be avoided; i.e. means that is, and each implies that what follows it is a *complete* list and not just a list of examples, as implied by for example or e.g.. We even suggest replacing e.g. by for example and replacing i.e. by that is, because we have discovered that some readers and writers—certainly not those reading *this* paper (:-))—confuse the two. The replacement serves to remind the writer what he wrote.

Rule 18: Avoid writing S containing 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.

[4]The authors realise the irony of a rule specifying by example that one should not specify by example. Indeed, when we go to implement this Rule in the proposed tool, we shall have the difficulty that the rule helps to avoid, namely that the implementer will have to complete the specification of the requirements before she can complete the implementation.

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.

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 a vague adjective such as prompt, fast, routine, etc. 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: Restrictions between different types of data access, either logical or physical, made at LVL2 must be valid if the data passed on to the online environment are stored and retrieved in the offline environment.

E11 is a long and complicated RStat that contains violations of Rules 1, 2, 4, and 16. The reference to logical or physical access is not clear, because there is insufficient information to tell whether logical access and physical access are the only two kinds of accesses. There is no explanation of who passes the data, of the kinds of data that can be passed, and of when the data are to be passed. The author of the Rstat would have to be asked to supply this additional information. Since we do not have access to the author, we took some guesses and rewrote E11 as E12 according to the recommendations of the violated rules.

> **E12:** Restrictions between logical or physical data access made at LVL2 must be valid when observational data that the server passes to the online environment are stored and retrieved in the offline environment.

Rewriting E11 as E12 may have obscured the relationships that exist between type of data access other than logical and physical. Rule 20 prevents this possibly obscurring rewriting.

Another example is E13.

> **E13:** The system must prohibit direct public access between external networks and any system component that stores cardholder information.

E13 violates Rule 16. The use of between indicates that there is a relationship between external networks and cardholder information. Rule 16 suggests rewriting E13 as E14.

> **E14:** The system must prohibit direct public access between each external network and each system component that stores cardholder information.

Rewriting E13 as E14 may have obscured the relationships between heretofore unidentified external networks and system components and thus may have modified the meaning of the original Rstat. Rule 20 prevents these obscuring changes that upset relationships that exist between the arguments of the between.

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

> **E15:** 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.:

> **E16:** 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, observe 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 the ambiguous identifier common, generic, customary, velc. Each of E17 and E18 has an instance of common, but the scopes of the two commons differ.

> **E17:** The simulation shall use instrument geometry that is defined and is common to all analysis modules.
>
> **E18:** All output messages shall be categorised (e.g., error, warning, debug) and reported via a common mechanism.

In E17, 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 containing RS. Furthermore, E17 contains a violation of Rule 16. The suggested rewriting of E17 is E19.

> **E19:** The simulation shall use instrument geometry that is defined and conforms to each analysis module.

Elimination of the vagueness of analysis module requires asking the RStat's author what he means. In E18, common is ambiguous, because it can mean same or everyday. In the absence of access to E18's author, we assume that the intended meaning is same. E18 violates also Rule 12b or 17. Hence, E18 can be rewritten as E20.

> **E20:** Each error, warning, and debug message shall be categorised and reported via the same mechanism.

In general, each of common, generic, customary, velc. has more than one scope. To remedy the ambiguity, it is necessary to describe the scope of the commonality, genericity, customariness, velc.

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

> **E21:** 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.

We understand that each Rstat shall be uniquely identifiable. Rule 23 suggests grouping together several dependent RStats into one, contradicting the idea that each Rstat be separately identifiable. However, the idea behind Rule 23 is to make sure that Rstats that have some relationship or dependency, temporal or not, with each other are grouped together. Otherwise, it may be difficult for the reader to detect the relationship or dependency. Blindly rewriting E21 into 3 separately identifiable Rstats:

E22: The SDP shall provide the Level 1 data to the P1 sites.

E23: The Level 1 data shall arrive at the sites no later than 24 hours after completion of processing in the SDP.

E24: Then, the SDP shall provide the Level 0 data to the P1 sites.

without inserting P1 before sites in E23 will cause the sites to become vague. Moreover, the temporal order bond between E22, E23, and E24 has been broken. It is not possible to know that the processes described by E22, E23, and E24 must be done in the order written. On the other hand, if separation of individual Rstats is required, then each separated Rstat must have additional text to describe its context. For example, in E23, the sites must be changed to the P1 sites and the fact that E23 temporally follows E22 and precedes E24 must be described.

Rule 24: Avoid writing S containing should and similar words, except as an expression of a preference that is not a requirement. If S is supposed to be a requirement, then rewrite S using shall.

Rule 25: Avoid writing S containing a plural subject [18], e.g.:

E25: 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 E25, 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:

E26: Each person in the room lifts his or her own table.

and

E27: 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.:

E28: All persons in the room together lift one table.

Rule 26: Avoid writing S containing *A* unless *B*. Instead, use if not(*B*), then *A*, where not(*B*) means the logical negation of *B*. which usually contains somewhere the word not. Doing the replacement makes it clear what *A* unless *B* means. The suggested rewriting of E29 is E30 and of E31 is E32.

E29: Unless the user has the administrator's authorisation, the user will not be able to access the database.

E30: If the user does not have the administrator's authorisation, the user will not be able to access the database.

E31: The system will display registration alert unless the user has registered.

E32: The system will display registration alert if the user has not registered.

Strictly speaking, *A* unless *B* says nothing about what happens if *B* is true. Therefore, if it is desired to specify that *C* happens if *B* is true, an explicit Rstat, if *B*, then *C*, must be given. For E29 and E31, the corresponding additional specifications are E33 and E34, respectively.

E33: If the user has the administrator's authorisation, then the user will be able to access the database.

and

E34: The system will not display registration alert if the user has registered.

The reason that Rule 26 is necessary is that we have evidence that an occasional person uses *A* unless *B* as not(*B*) if and only if *A*. Perhaps such a person is worried about what happens during the unspecified situation of *A* unless *B* when *B* is true. Rewriting *A* unless *B* as if not(*B*), then *A* drives home the point that

that *A* unless *B* is not the same as not (*B*) if and only if *A*.

Compounding this ambiguity is the fact that for some, *A* unless *B* has a temporal interpretation, in which *A* is true initially and it continues to be true until such time as *B* happens to be true [3].

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

> **E35:** An associate can only view the report which contains the payment details entered by the associate himself.

probably means

> **E36:** An associate can view only the report which contains the payment details entered by the associate himself..

It seems common in English to place only or also always immediately before the main verb of the containing sentence no matter which word is limited by the only or also. This common practice leaves the reader uncertain about what word is really limited by the only or also.

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

4 Validation: Analysis and Rewriting of RSs

We validated the guiding rules by rewriting existing and ambiguous RSs [16, 6, 11, 7] 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 E37, 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.

> **E37:** 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 E37 into two Rstats, E38 and E39.

> **E38:** If the payment is with the payee's detail, then the system will treat each payment separately.

> **E39:** If the user chooses "No", all the payment records will be grouped together to become one cheque.

However, just splitting E37 into E38 and E39 does not make E38 and E39 into independent Rstats, because they nevertheless have a temporal relationship; E39 follows E38. Rule 23 suggests grouping together requirements that show any such temporal dependency. Once joined into one RStat, E38 and E39 regain the lost temporal context.

E40 and E41 show the ambiguity resulting from the use of all in writing a plural subject, in violation of Rule 25. Note that E40 has also (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.

> **E40:** All login attempts shall be done so in a secure manner (e.g., encrypted passwords).

> **E41:** All pipeline products shall contain keywords, which describe the pipeline modules used to create them.

The violated rules suggest rewriting E40 and E41 into E42 and E43, respectively.

> **E42:** Every login attempt shall be done with an encrypted password.

> **E43:** Every pipeline product shall contain keywords that describe the pipeline modules used to create the pipeline product.

The change embodied in E43 assumes that each pipeline product is built from several pipeline modules. If each pipeline product is built from exactly one pipeline module, then E41 should be changed to E44.

> **E44:** Every pipeline product shall contain the keyword that describes the pipeline module used to create the pipeline product.

E45 contains a violation of each of Rule 16, Rule 25, Rule 12a, and Rule 17.

> **E45:** 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 E45 is E46.

> **E46:** 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 E45 is E47.

> **E47:** Each mission element shall withstand EMI, shock, and thermal environments.

E48 contains violations of Rule 12b and Rule 17 or a violation of Rule12a.

> **E48:** 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 E49.

> **E49:** 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 E50.

> **E50:** Each user of the system shall login by using some form of unique identification.

E51 violates 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.

> **E51:** 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 E52.

> **E52:** 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 E53.

> **E53:** The SAS is responsible for the Level 2 processing of the LAT data that computes the maximum, minimum, and average values.

E54 gives example Rstats that together fall under the province of Rule 23 and should be grouped together.

> **E54:** 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.

E55 contains violations of Rules 18 and 24 by its use of should and of whereas.

> **E55:** The user manual should document the expected 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 E55 as E56 and E57.

> **E56:** The user manual shall document the expected results.

> **E57:** The user interface shall provide information or a warning indicating what changes will occur when a user changes the regional setting.

These examples of identifying which rules apply to a given Rstat and rewriting the Rstat according to the suggestions of the rules serve as a partial validation of the usefulness of the rules.

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. This lesson is particularly apparent in the deliberations over E55, in which ambiguity arises from a lack of context.

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 to 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.

An anonymous reviewer of the previous version of this paper suggested evaluating the rules presented in this paper by comparing their ambiguity detection power with that of other sets of rules found in the literature [e.g., 12, 22, 17, 5, 8, 2].

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.

Recall that almost every rule is described by example, using such terms as e.g., etc., velc., etc. These rule descriptions are thus highly ambiguous. Therefore the rules will have to be refined with as full a list of examples as is possible during the development of SREE. Even then, it is expected that more examples for each rule will be found and SREE will have to be updated.

Perhaps the most valuable use of the rules and of SREE to inspect a RS is that they identify questions that should be asked of the client of the RS. Any time a Rstat is determined to be ambiguous, the requirements analyst must ask the client what she means by the Rstat. The kind of ambiguity found in Rstat shows the question that should be asked. A suggested resolution given with any rule is only a suggestion. Only the client can say what the resolution of an ambiguous Rstat should be.

Acknowledgements

The authors thank the anonymous reviewers of the previous version of this paper for their comments and suggestions.

Daniel Berry's work is sponsored in part by Grant (Canada) NSERC-RGPIN227055-00.

References

[1] K. Bach. Ambiguity. In E. Craig and L. Floridi, editors, *Routledge Encyclopedia of Philosophy*, London, UK, 1998. Routledge.

[2] A. Bucchiarone, S. Gnesi, and P. Pierini. Quality analysis of NL requirements: An industrial case study. In *Proceedings of the Thirteenth IEEE International Conference on Requirements Engineering (RE'05)*, pp. 390–394, 29 August–2 September 2005.

[3] M. Chandler. The logic of 'unless'. *Philosophical Studies*, 41(3):383–405, May 1982.

[4] C. Denger. High quality requirements specifications for embedded systems through authoring rules and language patterns. Technical Report M. Sc. Thesis, Fachbereich Informatik, Universität Kaiserslautern, 2002.

[5] C. Denger, D. Jörg, and E. Kamsties. *QUASAR: A Survey on Approaches for Writing Precise Natural Language Requirements*. IESE Fraunhofer, Kaiserslautern, DE, 2001.

[6] R. Dubois. 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`.

[7] C. S. Eng. Batch poster system, detailed business requirements. Technical report, EDS MySC, Malaysia, 2005.

[8] D. Firesmith. Specifying good requirements. *Journal of Object Technology*, 2(4):53–64, July-August 2003.

[9] N. E. Fuchs and R. Schwitter. Specifying logic programs in controlled natural language. In *CLNLP'95, Workshop on Computational Logic for Natural language Processing*, 1995.

[10] D. C. Gause. *User DRIVEN Design—The Luxury that has Become a Necessity, A Workshop in Full Life-Cycle Requirements Management*. ICRE 2000 Tutorial T7, Schaumberg, IL, USA, 23 June 2000.

[11] S. George. 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/`.

[12] I. Hooks. Writing good requirements. In *Proceedings of the Fourth International Symposium of the NCOSE*, volume 2, pp. 197–203, 1994.

[13] N. Juristo, A. M. Moreno, and M. Lopez. How to use linguistic instruments for object-oriented analysis. *IEEE Software*, 17(3):80–89, May/Jun 1997.

[14] J. Lawler. Post subject: Re: difference between "each" and "every", Sun October 23 2005. `http://www.vocaboly.com/forums/ftopic9080.html`.

[15] B. Macias and S. Pulman. Natural language processing for requirements specifications. In F. Redmill and T. Anderson, editors, *Safety-critical systems: Current issues, techniques and standards*, pp. 67–89, London, UK, 1993. Chapman & Hall.

[16] R. Moeser and P. Perley. 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`.

[17] C. Rupp and R. Goetz. Linguistic methods of requirements-engineering (NLP). In *Proceedings of the European Software Process Improvement Conference (EuroSPI)*, November 2000. `http://www.iscn.com/publications/#eurospi2000`.

[18] U. Schwertel. Controlling plural ambiguities in Attempto Controlled English. In *Proceedings of the Third Internaltional Workshop on Controlled Language Applications (CLAW)*, Seattle, WA, USA, 2000.

[19] S. F. Tjong. Elaborated natural language patterns for requirements specifications. Technical report, Faculty of Engineering and Computer Sciences, University of Nottingham, August 2006. `http://sepang.nottingham.edu.my/~kcx4sfj/`.

[20] S. F. Tjong. Improving the quality of natural language requirements specifications through natural language requirements patterns. Technical report, Faculty of Engineering and Computer Sciences, University of Nottingham, March 2006. `http://sepang.nottingham.edu.my/~kcx4sfj/`.

[21] S. F. Tjong. Improving the quality of natural language requirements specifications through natural language requirements patterns. In *IEEE International Conference on Computer and Information Technology*, September 2006.

[22] W. M. Wilson, L. H. Rosenberg, and L. E. Hyatt. Automated analysis of requirement specifications. In *Proceedings of the Nineteenth International Conferences on Software Engineering (ICSE-97)*, pp. 161–171, New York, NY, USA, 17–23 May 1997. ACM Press.