Ambiguity in Natural Language Software Requirements: A Case Study

de Bruijn, F.; Dekkers, H.L.

DOI
10.1007/978-3-642-14192-8_21
Publication date
2010
Document Version
Author accepted manuscript
Published in
Requirements Engineering: Foundation for Software Quality

Ambiguity in Natural Language Software Requirements: A Case Study

Fabian de Bruijn and Hans L. Dekkers

University of Amsterdam,The Netherlands
{f.debruijn,h.l.dekkers}@uva.nl

Abstract. [Context and motivation] Ambiguous requirements are often seen as a cause for project failure, however there is little empirical data to support this claim. **[Question/problem]** In this research we study the effect of a highly ambiguous requirements document on project success. **[Principal ideas/results]** The studied project was a complex data processing system that took about 21 man year to develop. First, we determined the level of ambiguity by three independent tests. Next, we did a root cause analysis on a selection of the main issues to establish if ambiguous requirements were a significant cause. Surprisingly, this case study shows that only one of the examined failures was caused by ambiguous requirements. Both the independent test team and the third party development team found ways to cope with the high level of ambiguity. For the development team this required a substantial investment to clarify requirements. **[Contribution]** The main contributions of this paper are the counterintuitive findings, the collected empirical data and the method used to collect these data.

Keywords: Requirements specification, Ambiguity, Natural language, Empirical.

1 Introduction

Requirement specifications serve as contract, a starting point for development, and a focal point for quality control. It is generally considered to be vital that requirements are unambiguous [2,4,9,12,15]. Formal languages serve this purpose, however, since formal languages are not well understood by most stakeholders, natural language requirements are the de facto standard. Berry *et al.*[4] make a strong case for writing unambiguous natural language specifications and provide a handbook which describes a taxonomy of different types of ambiguity and how to avoid them. Since natural language is inherently ambiguous [3], avoiding ambiguity is by no means a trivial task.

We wonder just how important it is to minimize ambiguity and how much effort should be invested. The agile movement puts the focus on communication and feedback. They stress that writing unambiguous requirements is an illusion[16] and that even if requirements are unambiguous, problems of validity, volatility, and correct interpretation remain. Still, the general practice of tenders and contracts for outsourced projects demands a requirement specification. In this research we study the effect of a highly ambiguous requirements document on project success.

R. Wieringa and A. Persson (Eds.): REFSQ 2010, LNCS 6182, pp. 233–247, 2010.

1.1 Research Question

Our main research question is:

> *What is the effect of ambiguity in the requirement specification on project success?*

The initial step we have taken is to analyze a real life project:

1. *How many requirement statements are ambiguous?*
2. *How many problems were caused by ambiguous requirements?*

2 The Importance of Unambiguous Requirements

2.1 Communication in Requirements Engineering

Requirements engineering is the process in which stakeholder needs are elicited, gathered, analyzed and in which decisions are made on the requirement set for the product to be built. Requirements are an abstraction and a perception of the true needs of the stakeholders. It is the result of a creative process where communication is crucial[17]. It is not evident that the requirement set as a whole is feasible, complete or correct.

In the context of this paper it is important to distinguish between the stakeholder need, the requirement statement and the way it is understood[20]. We study the effect of ambiguous requirement statements on the understanding of the developers. This understanding also depends on context information (goals, rationale, domain description), domain knowledge and personal factors.

Requirement specifications are not standardized and many different types of requirements exist [1,12,17,15]. It is good practice to write requirements in the problem domain, leaving the design space open for the development team. This poses the interesting problems of how to be specific and how to determine if a solution satisfies the requirements. To illustrate this consider usability requirements. Usability can be measured by the number of tick and clicks and user errors. But how to set a norm: what is an optimal solution, when is a suboptimal solution still acceptable?

2.2 Related Work

In software engineering literature there is no single definition of ambiguity. Several authors have given different interpretations and different causes for ambiguity. For instance, Davis[8] states that when a requirement can be interpreted in two or more ways then this requirement is ambiguous. Schneider *et al.*[19] mention that ambiguity is caused by an essential part in a software requirement that has been left undefined or defined in a way that causes confusion among humans. Berry *et al.*[4] focus on ambiguities which are caused by expression inadequacies.

There is general consensus that requirements should be unambiguous or at least that ambiguity should be recognized and intended. Much work has been done to achieve quality in requirements [2,12,15,18] and reduce the ambiguity in natural language software requirements. In [4,13,14] methods and rules are discussed to surface ambiguity in natural language text including techniques like checklist and scenario based reading.

Fabbrini et al. [10,9] present a tool to analyze the quality of natural language requirements written in English. The tool Alpino[5] can be used to automatically parse texts written in Dutch, to find the most probable interpretation.

3 Research Method

To answer the research question "What is the effect of ambiguity of software requirements on project success?" we studied a real life project that failed. First we established the level of ambiguity in the requirements specification. Next we established if ambiguity was a significant cause for the reported issues.

3.1 Case Study Project Information

The project started in December 2007 and was canceled in May 2009 after acceptance tests by an independent test team found over 100 blocking issues. The contractor was convinced that ambiguity was an important cause of many of the reported issues. The case study is referred to as Project X. For reasons of confidentiality we cannot fully disclose requirements. Project X was the development of an Oracle system that was composed of a GIS component, a data processing component and a rule engine connected by an enterprise service bus. Two of these components were existing systems that were adapted for this project. Project X took well over 40.000 man hours, roughly 21 man year; a significant budget overrun given the initial estimation of 10.000 hours. 80 persons worked on the project.

The requirement specification was in Dutch. From the start the requirements were considered to be unclear. To get more clarity workshops were organized and an elaborate design was made, however, when asked, the customer did not formally agree with the design. The independent test team based its findings on the initial requirements document.

3.2 Establishing Ambiguity

We consider a requirement to be ambiguous when it has at least two different valid interpretations. To establish ambiguity three independent tests were performed on a sample set of the requirements:

- three professionals searched for differences in interpretations;
- a systematic review by two software engineers based on the taxonomy of [4];
- an automatic analysis by natural language analysis tool Alpino [5] .

3.2.1 Sample Data

The sample size was set to 102[1] from a total of 279 requirements. A stratified random sample [6] was chosen from the requirement specification. All tests to detect ambiguity

[1] After the first day of reviewing a sample of 100 requirements was considered feasible. To get an even distribution over the requirement categories we took the % and rounded this. E.g. if a requirements category consisted of 5.6% of the complete number of requirements, we took 6 requirements from this set. This resulted in a complete sample of 102.

were performed on this sample. Table 1 lists the different requirements categories, % of total is the total number of requirements in the category / total no. of requirements.

3.2.2 Review Panel Interpretations

The requirement statements were reviewed by two requirement engineers and one software engineer. The reviewers were experienced professionals with no prior knowledge of Project X and with a similar domain knowledge as the development team. During $2\frac{1}{2}$ days they studied each requirement and wrote down the interpretation they thought was most fit. To determine if interpretations differed, a joint session was organized by the first author, taking half a day. Reviewers shared their interpretations and could judge if they had the same understanding. To make sure that interpretations were plausible and truly different the reviewers had to present some form of argument or example.

Table 1. Requirement categories

ID	Category	% of total	sample size
1	Functionality A	19,00%	19
2	Functionality B	13,98%	14
3	Functionality C	8,96%	9
4	Functionality D	8,60%	9
5	Functionality E	7,53%	8
6	Infrastructure	6,81%	7
7	Maintainability	5,02%	5
8	Software features	5,02%	5
9	Functionality F	4,66%	5
10	Usability	4,30%	4
11	Security	3,94%	4
12	Data model	3,94%	4
13	Reliability	3,23%	3
14	Performance	2,51%	3
15	Functionality G	2,51%	3

3.2.3 Systematic Review

The review was carried out by the first author and a software engineer with no prior knowledge of Project X. A checklist was used, based on the linguistic ambiguity taxonomy of Berry *et al.*[4]. The reviewers had no linguistic background. The review protocol was as follows.

1. Each reviewer individually determines the ambiguity types present in a requirement statement.
2. The results are merged, showing the identified ambiguity types.
3. Each reviewer can adjust his findings based on the merged results.
4. Finally consensus was reached in a face to face meeting.

3.2.4 Automatic Analysis

To get an objective measure for ambiguity each requirement was parsed by the tool Alpino[2][5]. Alpino recognizes lexical and structural (or syntactical) ambiguity. The output showed the possible parse trees. The number of possible parses was used as indicator for ambiguity. Not all of these parses have a clear semantics or are semantically different. The limit of parses was set to a maximum of 50.

[2] Alpino version 14888 for the x86 Linux platform was used in this experiment.

3.3 Relating Issues to Ambiguous Requirements

To establish if ambiguity was a significant cause for project failure, we did a systematic root cause analysis on a sample of issues found by an independent test team. The issue tracking system used by Project X was JIRA. Issues have fields to describe and reproduce the issue (e.g. description, version, component); to capture the status of the issue (e.g. type, status, priority) and a chronological log showing the comments on the issues and steps to clarify and solve these by the developers. Sometimes attachments were presented containing snapshots of the system, or a specification of desired behavior by the test team.

Table 2. Root cause conditions

Label	Root cause condition
RCC1	The implementation satisfies the requirement.
RCC2	The test team rejects this because of a different but also valid interpretation of the requirement.
RCC3	Test team respects the design space of the contractor[3].
RCC4	Disambiguation of the requirement would have prevented these differences in interpretation

In total 256 issues were reported, we focused on the 125 issues that were considered blocking or urgent. 16 of these were addressing problems like incomplete product documentation and were excluded from our research. The 109 remaining issues were related to the requirement categories. While reading the issues to relate them to requirement categories, 5 issues surfaced that were likely to have been caused by ambiguity. These issues were either labeled as such or there was a discussion about the interpretation. A root cause analysis was performed on these five issues and on a random selection of 35 issues. The issues were annotated by discussion threads of the contractor which were also studied. All issues were analyzed by each of the reviewers individually.

We consider ambiguity to be the cause of an issue if all conditions in table 2 are satisfied.

4 Results

4.1 Measured Ambiguity

4.1.1 Review Panel Interpretations

The review panel of three different persons qualified 36 requirements as unambiguous, 41 requirements had two different interpretations, and finally 25 requirements had three different interpretations. Figure 1 shows for each requirement category how many requirements were ambiguous and how many were unambiguous.

[3] Many requirements are formulated in the problem domain, explicitly leaving the choice of solution to the contractor. In some cases the test team prefers a solution different from the one that is implemented while it is apparent that the requirement is interpreted in the same way as the contractor. These issues are not considered to be caused by ambiguity.

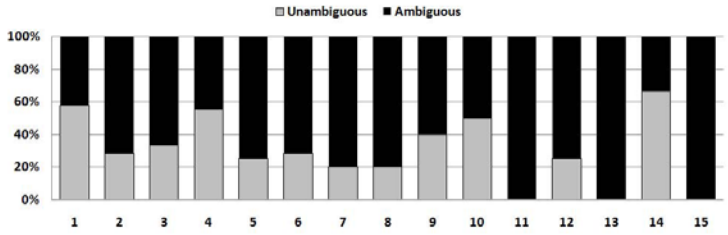

Fig. 1. x-axis: requirement categories, see table1. y-axis: % of requirements.

Table 3. Ambiguity types top 5 found in requirements sample

Ambiguity type	# Requirements
Vagueness	58
Language error	32
Coordination Ambiguity	11
Scope Ambiguity	9
Attachment Ambiguity	8

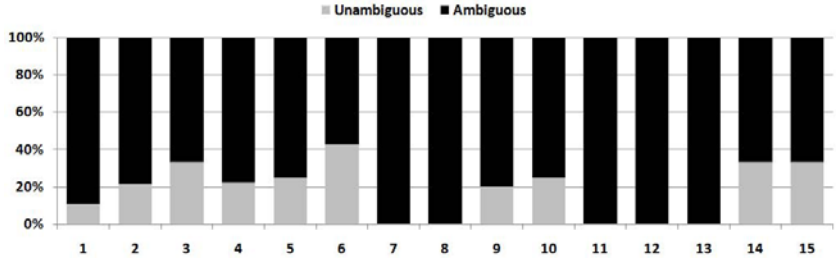

Fig. 2. x-axis: requirement categories, see table1. y-axis: % of requirements.

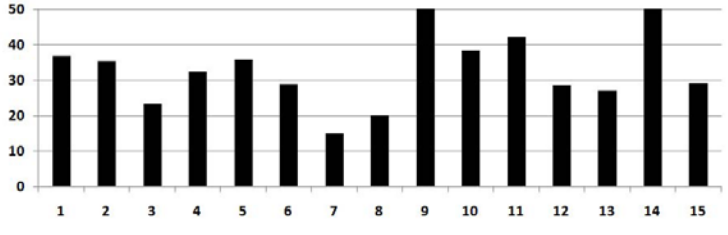

Fig. 3. x-axis: requirements categories, see table 1. y-axis: average number of parse trees per requirement.

4.1.2 Systematic Review

The systematic review found 20 of the requirements to be unambiguous. 82 requirements were found to be ambiguous. In 36 of the requirements multiple types of ambiguity were

discovered. Table 3 lists in how many requirement statements the most common ambiguity types were found. Note that the number of ambiguities within one single requirement statement is not counted. Figure 2 shows for each requirement category how many requirements were ambiguous and how many were unambiguous.

4.1.3 Alpino Tool Interpretations

For 94 of the requirements Alpino found more than one parse tree, meaning that these are syntactical ambiguous. Seven requirements could not be parsed. Alpino calculated an average of 33 different parse trees for each requirement[4]. The median was 47 and the standard deviation was 19. Only one requirement had one parse tree. This requirement was also understood in one way by the review panel members. Figure 3 shows the average number of interpretations for each requirement category.

4.2 Issue Causes

40 issues were analyzed. The root cause of these issues is presented in table 4. We found one issue to be caused by ambiguity. The issue was "Processing batch file takes too long". The corresponding ambiguous requirement was "The processing of the batch programs should be completed within a reasonable time frame without affecting the performance of the application and therefore without impeding the usual business." For the contractor it was and still is not clear when the processing time is acceptable. The performance was improved, the effort is discussed in section 5.

Table 4. Results issue root cause analysis

Root cause	No. of issues	Explanation
Ambiguity	1	Issue caused by ambiguity
Feature not found	3	Feature had been implemented, but the implementation was not known to the test team.
Missing requirement	5	For these issues no requirement could be identified.
New feature request	1	The reported issue is not required by the current requirements.
Incorrect implementation	27	Acknowledgment by contractor that the issue reports undesired behavior
Not reproducible	3	These issues could not be reproduced
Total no. issues	*40*	From 109 blocking or urgent issues

To make our reasoning process transparent we illustrate this in table 5. The presented issue was selected because it raised some discussion and gives good insight in our analysis. Determining if the implementation satisfied the requirements was sometimes straightforward, sometimes it was hard to ascertain. The issue and requirements were in Dutch and there is some risk that ambiguity is lost in translation. Also the issue description and annotations are too large to be presented.

[4] Calculations over the set of requirements that could be parsed.

Table 5. Reasoning process details: issue I23

(a) Context information

Issue caption	The application can not be controlled by the keyboard. Note: the test team provides some examples of expected key controls.
Requirement	The whole system has a consistent user interaction. The application can both be controlled by keyboard and by mouse, based on a completely WEB oriented graphical user interface.

(b) Reasoning process

RCC1	TRUE	A consistent keyboard interface has been implemented. To use the application the mouse is not required. The development team agrees that the keyboard control can be improved upon, however this is considered to be a new feature request.
RCC2	TRUE	The test team provides some examples of keyboard controls that they had expected to be implemented but were not.
RCC3	FALSE	The comments and examples of the test team show that they interpret the requirement identically to the development team. From the issue text it becomes clear that the test team is not knowledgeable of the keyboard controls that have been implemented. There is no indication if they find the current implementation acceptable.
RCC4	FALSE	The requirement is specific enough to judge the present implementation. The vagueness in the requirement leads to misunderstanding. However this vagueness is intended, so the contractor can choose the specific solution.
Conclusion		Feature not found. For the test team it was not apparent how this requirement was implemented. As they did not find this feature, they filed an issue report. The issue specification is not required by the current set of requirements.

5 Evaluation

Research question: How many requirement statements are ambiguous?

Table 6 shows that the studied requirements sample of 102 requirements revealed a lot of ambiguous requirements. Alpino considered all but 1 requirement to be ambiguous. The systematic review considered 83 out of 102 requirements to be ambiguous. The review panel considered 66 of the 102 requirements to be ambiguous. That the review panel has a single reading for an ambiguous statement corresponds with the notion of innocuous ambiguity[7].

Research question: How many problems were caused by ambiguous requirements?

Only one of the forty inspected issues was caused by ambiguity in the requirements. This issue was not a costly one. From our study we cannot conclude that ambiguous requirements caused the failure of this project.

How come there weren't more issues caused by ambiguous requirements?

Table 6. Ambiguity of requirements according to review panel, systematic review, and Alpino

Review panel	Systematic review	Alpino	Requirements (#)
Unambiguous	Ambiguous	Unambiguous	1
Unambiguous	Unambiguous	Ambiguous	9
Unambiguous	Ambiguous	Ambiguous	27
Ambiguous	Unambiguous	Ambiguous	10
Ambiguous	Ambiguous	Ambiguous	48
Ambiguous	Ambiguous	Parse Error	7

Although the requirements specification from project X was highly ambiguous most of the examined issues could not be attributed to ambiguity. Project X used workshops involving the customer to clarify the requirements. Yet, even workshops and discussion don't guarantee a good interpretation. Given the complexity of this project, the many issues, and the lack of contact between test team and development team we were surprised by this finding. This is in line with the observation of [11] that the biggest danger is unconscious disambiguation. The software engineer interprets an ambiguous requirement differently then the customers intention, but is unaware of this. In this project the contractor was from the start aware of the high level of ambiguity.

What is the cost of ambiguous requirements?

The issue that was caused by ambiguity is about performance, potentially a costly issue. However the architecture of the application was set up to process vast amounts of data in reasonable time. To get a reasonable performance took roughly 550 hours[5]. The issue was considered to be resolved however from the issue annotations it was apparent that still much was and is unclear about the real time scenarios (how much data in what time slots) and what performance is considered to be acceptable.

The project suffered a major budget overrun of 30.000 hours. The 550 hours for the issue caused by ambiguity is limited. The workshops to clarify ambiguity were included in the budget (180 hours). The project data does not show what part of the budget overrun is caused by ambiguity.

6 Threats to Validity

6.1 Validity of the Tests to Determine Ambiguous Requirements

Is the sample set of requirements representative?

Throughout the research project we have read and interpreted the complete set of requirements intensively. The requirement sample was characteristic for the whole set of requirements. For the different types of requirements the requirement statements follow a similar pattern and use similar words. We found no occurrences of requirements that were more specific than the ones studied in our sample. Since the different requirement types are in different requirement categories, we consider our sample to be representative for the complete set of requirements.

[5] According to the project manager one software engineer worked on the performance optimization for three months.

6.1.1 Threats to Validity of Ambiguity Tests

6.1.1.1 The Interpretations by the Review Panel. The good thing about this test is that ambiguity that does not lead to misinterpretations will not be reported. However, the reviewers might have an invalid interpretation or a different interpretation from the actual project team or customer. This test is not just an indicator of ambiguity, it also says something about the interpretation process of the reviewers. The final threat to validity is that the review panel is under the impression they have a different interpretation, while they actually share the same interpretation (false positive). This last threat could have been avoided by making the interpreters formalize their interpretation as described in [13].

6.1.1.2 The Systematic Review. Detecting ambiguities by humans is a hard task. The reviewers were no trained linguists and unconscious disambiguation makes it easy to miss ambiguity types. We expect that the number of false negatives is rather high. The ambiguities found complied with the taxonomy of [4] and the test protocol ensured that the found ambiguities were analyzed well. We expect that the number of false positives is rather low.

6.1.1.3 Alpino Scan. Alpino was used to get objective measures for ambiguity. When a requirement has at least two parse trees then the requirement has structural or lexical ambiguity. As described in [7] many of these ambiguities have a single reading by humans and are innocuous. Alpino features a maximum-entropy based disambiguation component to select the best parse for a given sentence. From our discussion with the Alpino research group it became clear that there is not a clear threshold that can be used to automatically determine which of the parse trees is a plausible interpretation. This would have enabled us to automatically distinguish between nocuous and innocuous ambiguity. Also to date Alpino has no feature to report the different types of ambiguity. The parse trees of Alpino are used to detect false negatives.

6.1.2 False Positives of the Ambiguity Tests

6.1.2.1 Unambiguous by Review Panel and Ambiguous in Systematic Review. Since the systematic review showed that 80% of the requirements were ambiguous it is especially interesting to learn about false positives. We did two checks. The first was to assess if the reviewers had a correct understanding of vagueness. This was the ambiguity type that was most discovered (in 58 requirement statements, see table 3). The analysis was done by the second author and a researcher with a linguistic background. Three requirements were analyzed and we could conclude that the classification vague was used according to the taxonomy presented in [4].

False positives are most likely to occur in the requirements where only one ambiguity type was revealed and which the review panel found to be unambiguous. The second author examined five of these requirements to see if the found ambiguity types were according to the taxonomy presented in [4]. Fourteen requirements, see figure 4, met this condition. Eleven of these had ambiguity type "vague", two were of type "language error", one had the type "attachment".

The requirements of type language error contained two plurals but both requirement statements contained no verb. The lack of a verb leaves it to the reader to guess about it. Depending on the verb it could be a language error or it could be a scope ambiguity.

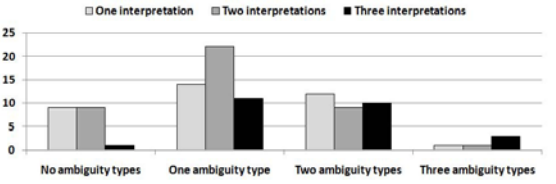

Fig. 4. x-axis: the number of different ambiguity types. y-axis: the number of requirements.

The requirement with ambiguity type attachment was of the form "overview of old and new values with difference percentages". The difference percentages could indeed be attached to new values alone, or to old and new values and to the difference between old and new values. It is also logical that the review panel interpreted this unambiguously as the difference between old and new presented in %. However we would expect that there is still some ambiguity not considered by the review panel. One of the formalizations could be (new value - old value) / old value, now it is easy to see what variations are allowed by this requirement: should we divide by old value or by new value. Should we subtract the old value from the new value or vice versa or take the absolute difference. Clearly an ambiguous statement. Also the two vague requirements were true positives. That the review panel considered these to be unambiguous can be explained that the vagueness was caused by IT terms which were understood identically by the review panel because of their identical IT background.

This analysis revealed no false positives.

6.1.2.2 Ambiguous by Review Panel and Unambiguous in Systematic Review. False positives are also likely to occur in the requirements that were considered to be ambiguous by review panel and unambiguous by the systematic review. Ten requirements met this condition; five of these requirements were analyzed. From the transcripts of the review panel it was apparent that four of these five requirements were understood in a different way and these can be considered to be true positives. In three of these cases the requirement contained an ambiguity type that was unnoticed during the systematic review (these were of types scope and coordination ambiguity). In the fourth case one of the interpreters made a simplification of the requirement that left out important details. This difference in interpretation was not caused by ambiguity.

The fifth case was a possible false positive. This concerned a requirement of the form "Possibility to change the start and end date of a specific process". During the analysis the reviewers were convinced that each of them had a different interpretation. The transcript with their interpretations showed some differences in the sense that the start date can be changed if and only if the end date is changed as well. However it is hard to conceive that the reviewers would have constructed a truly different solution for this requirement.

This analysis revealed one false positive.

6.1.3 False Negatives of the Ambiguity Tests

6.1.3.1 Unambiguous by Review Panel and Unambiguous in Systematic Review. To identify false negatives we analyzed requirements found to be unambiguous by both the review panel and by systematic review. The second author examined these nine occurrences using the checklist of the systematic review. This inspection found two requirements to be ambiguous, both of type attachment. It was easy to see why these were missed the first time; the semantics of the requirement only allowed one sensible interpretation. For four requirements for which the second author also could not identify an ambiguity type the Alpino parses were analyzed. This revealed no new ambiguous requirements. Alpino had multiple parse trees for compound words (finding the right grouping), for words that in Dutch are sometimes used as verb and sometimes as adjective and with the use of ":" as classifier. The different parse trees have the same semantics, and are examples of innocuous ambiguity[7].

This analysis revealed no false negatives as the ambiguous requirements only had one interpretation.

6.2 Validity of the Root Cause Analysis

Is the sample set of issues representative?

A first read of all blocking and urgent issues revealed five issues that were likely to have been caused by ambiguity. Indeed all of these five issues sparked a lot of discussion. Initially we extended this set of five with a random sample of 20 issues. When our first analysis revealed that only 1 issue was caused by ambiguity, another 15 issues were randomly selected and analyzed. The analysis showed that the new set of 15 issues had similar causes as the initial set of 20 randomly selected issues. This strengthens our belief that the sample was representative

6.2.1 Threats to Validity

Since few issues were found to be caused by ambiguity, there is limited danger of false positives. To reduce the number of false negatives the root cause analysis followed a formalized protocol and was carried out autonomously by both authors. The controversial issues have been discussed in depth and clear reasons were found to eliminate ambiguity as root cause.

6.2.2 False Negatives

The category least discussed is the category in which the contractor clearly acknowledges that the implementation is not conform the requirements. This was also the biggest category with 27 of the 40 researched issues. Thirteen of these issues were inspected further. To our surprise for 9 of the 13 cases no requirements were found. The contractor was given a lot of freedom in choosing the design. The issues could be seen as improvements or comments on the design choices made. Even though the

improvements and comments were not specified by the requirements, the contractor felt that the improvements and comments were reasonable and without much discussion qualified each issue as bug. Most issues had been resolved at the time of the research. Our analysis would classify these 9 issues as "new feature request". This analysis of 13 issues revealed no false negatives.

7 Other Observations

Is the number of words an indicator for ambiguity?

We were curious about the effect of length of a requirement statement on ambiguity. The longer the requirements the more prone it is to syntactical ambiguity. However the length could also indicate that extra information was presented that would help to interpret the requirement. For this analysis the requirements were partitioned by requirement size in 8 groups ranging from 6 to 45 words. Each group has a span of 5 words. Figure 5 shows that the more words were used the more interpretations were given by the review panel. The average word count for unambiguous requirements was 11, the average word count for ambiguous requirements with two interpretations was 20. It is tempting to say that requirements should be written as short as possible, but it could very well be that the requirements were so lengthy because the message being expressed was complicated. What we can learn is that lengthy requirements demand extra attention.

Will disambiguation decrease the different interpretations among stakeholders?

Five randomly selected requirements were rewritten with the help of the rules described in Berry *et al.*[4]. The selected requirements were found to be ambiguous by both the review panel and by the systematic review. Furthermore, the average word count of these requirements was 25. Rewriting these five requirements took about half a day, however this did not include a check that the new description expressed the intention of the customer. The rewritten requirements were reviewed again by the same protocol as specified in subsection 3.2.2.

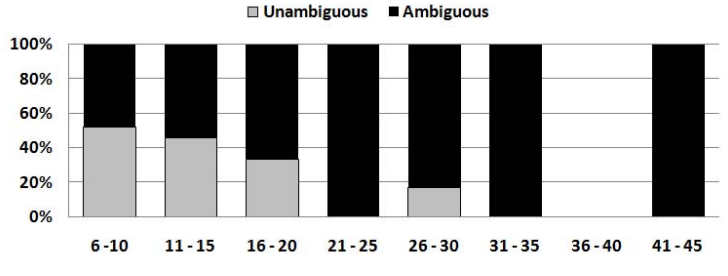

Fig. 5. x-axis: requirements with no. of words. y-axis: % ambiguous vs unambiguous.

This test found that four of the five requirements were unambiguous. The fifth requirement was still ambiguous, caused by a vague word. Disambiguating required mandating a specific solution, limiting the design space more than the customer required.

The rewritten requirements contained more words than the original requirements. In fact, the average word count was 46 for the rewritten requirements, the requirements contained more and shorter sentences. The review panel members mentioned they had no problems in comprehending the rewritten requirements. This shows that length is not the most important indicator for ambiguities.

8 Conclusion

In this research we studied the effect of a highly ambiguous requirements document on project success. The studied project was the development of a complex system that took about 21 man years to develop and was canceled after an independent test team found over 100 blocking issues. The perception of the contractor was that many of these issues were caused by ambiguity in the requirements. Independent tests by humans showed that 91% of the requirements were ambiguous. An automated test revealed that 92% of the requirements were ambiguous. A root cause analysis on 40 of the main issues showed that only one of the examined issues was caused by ambiguous requirements. This issue was resolved and explained 2% of the budget overrun.

In this project, ambiguous requirements were not the main cause of the issues found by the external test team, and cannot explain the failure of the project. Both the independent test team and the third party development team found ways to cope with the high level of ambiguity.

We can only speculate to the reason why the project was canceled. As a fixed price project it wasn't because of the budget overrun. Studying the bug reports we saw that the number of open defects and newly found defects remained high throughout the acceptance test. This resulted in a loss of confidence in the product by the customer. A possible explanation for most of the issues is that due to schedule pressure not enough care was given to implementation details. Also, as Brooks already knew, adding people to a project that is already late is usually not effective.

9 Future Work

We speculate that a requirements document that is perceived to be of low quality triggers a process of better understanding requirements. It would be interesting to see what the effect is of ambiguity in a requirements document that is perceived to be of high quality. This might result in false confidence that all requirements are correct. The development team will be less inclined to question these requirements and the problems have a bigger chance to go unnoticed. It would be interesting to repeat this research for such projects. It is also interesting to repeat this research in projects where communication is challenged by organizational, structural and political factors. In these cases we speculate that ambiguous requirements will have a bigger impact.

References

1. Abran, A., Moore, J.W., Bourque, P., Dupuis, R.: SWEBOK: Guide to the software engineering Body of Knowledge. IEEE Computer Society, Los Alamitos (2004)
2. Alexander, I.F., Stevens, R.: Writing better requirements. Addison-Wesley, Reading (2002)
3. Berry, D.M.: Ambiguity in Natural Language Requirements Documents. In: Paech, B., Martell, C. (eds.) Monterey Workshop 2007. LNCS, vol. 5320, pp. 1–7. Springer, Heidelberg (2008)
4. Berry, D.M., Kamsties, E., Krieger, M.M.: From contract drafting to software specification: Linguistic sources of ambiguity. Univ. of Waterloo Technical Report (2003)
5. Bouma, G., Van Noord, G., Malouf, R.: Alpino: Wide-coverage computational analysis of Dutch. In: Computational Linguistics in the Netherlands 2000. Selected Papers from the 11th CLIN Meeting (2001)
6. Campbell, M.J., TDV Swinscow: Statistics at square one. John Wiley & Sons, Chichester (2002)
7. Chantree, F., Nuseibeh, B., de Roeck, A., Willis, A.: Identifying nocuous ambiguities in natural language requirements. In: 14th IEEE International Conference Requirements Engineering, pp. 59–68 (2006)
8. Davis, A., et al.: Identifying and measuring quality in a software requirementsspecification. In: Proceedings of First International Software Metrics Symposium, pp. 141–152 (1993)
9. Fabbrini, F., Fusani, M., Gervasi, V., Gnesi, S., Ruggieri, S.: Achieving quality in natural language requirements. In: Proceedings of the 11 th International Software Quality Week (1998)
10. Fabbrini, F., Fusani, M., Gnesi, S., Lami, G.: An automatic quality evaluation for natural language requirements. In: Proceedings of the Seventh International Workshop on Requirements Engineering: Foundation for Software Quality REFSQ, vol. 1, pp. 4–5 (2001)
11. Gause, D.C.: User DRIVEN design - The luxury that has become a necessity. In: A Workshop in Full Life-Cycle Requirements Management. ICRE (2000)
12. Hull, E., Jackson, K., Dick, J.: Requirements engineering. Springer, Heidelberg (2005)
13. Kamsties, E.: Surfacing ambiguity in natural language requirements. PhD thesis, Fachbereich Informatik, Universitat Kaiserslautern, Kaiserslautern, Germany (2001)
14. Kamsties, E., Berry, D.M., Paech, B.: Detecting ambiguities in requirements documents using inspections. In: Proceedings of the first Workshop on Inspection in Software Engineering (WISE 2001), pp. 68–80 (2001)
15. Lauesen, S.: Software requirements: styles and techniques. Addison-Wesley, Reading (2002)
16. Mullery, G.: The perfect requirement myth. Requirements Engineering 1(2), 132–134 (1996)
17. Nuseibeh, B., Easterbrook, S.: Requirements engineering: a roadmap. In: ICSE 2000: Proceedings of the Conference on The Future of Software Engineering, pp. 35–46. ACM, New York (2000)
18. Robertson, S., Robertson, J.: Mastering the requirements process. Addison-Wesley Professional, Reading (2006)
19. Schneider, G.M., Martin, J., Tsai, W.T.: An experimental study of fault detection in user requirements documents. ACM Transactions on Software Engineering and Methodology (TOSEM) 1(2), 188–204 (1992)
20. Schramm, W.: How communication works, p. 51. Mass Media & Society (1997)