ECE 321: Software Requirements Engineering Assignment 2

Arun Woosaree

1514457

October 10, 2018

1 Requirements and Psychology and The Syntactically Dangerous All and Plural in Specifications review

1.1 Main contributions

Both articles include opinions and suggestions from experienced authors on how to avoid ambiguous language to make your requirements documents more precise. Although their goals are the same, the authors' approaches are different. The first article outlines three common defects: "Deletion", "Generalization", and "Distortion". Deletion happens when a verb, or a so-called "process-word" is not defined. For example, the requirement statement might specify an action like making a report, but who makes the report, what is reported, when, and how is undefined. Generalization can happen when universal quatifiers are used, and certain exceptional cases arise, to which the general statement won't apply. Distortion occurs when the process is "reformulated into an event". In the second article, the author focuses on when and how "all", "each", and plural words should be used, in order to minimize ambiguity. Article [2] focuses a bit more on the syntactic nature of the problems providing examples and, analyzing some math and other languages, while article [1] focuses more on describing the application of techniques described.

1.2 Criticisms

In both articles [1] and [2], the authors make excellent recommendations overall on how to improve the preciseness of the language used in requirements documents, and reducing ambiguous language. However, they both focus very narrowly on different aspects which can make requirements documents ambiguous. Unfortunately, following these authors' tips alone will not singlehandedly make your requirements documents amazingly super precise. For example, the author from article [2] suggests specific cases where "all", "each", or plural words should be used, but while following these recommendations, a statement which has defects outlined in [1] can still be made. For example, the sentence: "All lights at any intersection must have a single turning signal" is precise according to article [2], since "all" is used to describe a property of the whole set (lights). However, according to article [1], this statement suffers from generalization, since there might be exceptional cases where a turning signal may not make sense, or is impossible. The statement can be made more precise by saying: "There should be only one turning signal per intersection." The suggestions from the authors, although excellent should be combined with multiple other techniques for making your requrements document more ideal, that is with more precice language, and less ambiguities.

1.3

1.3.1 Difference Between "Generalization" and "All and Plural"

A requirement statement that suffers from the "All and Plural" defect can have multiple meanings, depending on how the reader understands the statement. On the other hand, requrement statements suffering from the "Generalization" defect tend to have ambiguities resulting from exceptional cases where a certain action may not be required, or is otherwise impossible. An example of a statement with the "All and Plural" defect is: "All traffic lights at an intersection shall have a turning signal." In this sentence, one could interpret it as each traffic light having its own respective turning signal, or someone else could understand it as having only one turning signal at each intersection. An example requirement with the "Generalization" defect is: "Each traffic light shall have a left turning signal." In this case the ambiguity lies in whether special cases or exceptions can occur, where there could be situations such that a left turning signal is not required, or is otherwise impossible. For example, an intersection where left turns are not allowed.

1.3.2 When Should We Use All vs. Each?

According to the author in [2] "all" should be used when the intention is to talk about properties of the whole set, whereas "each" should be used when the intention is to talk about properties of each member of the set. For example, we could say: "All intersections must have a single turning signal." to mean that there should be only one turning signal per intersection, or we could say: "Each traffic light at any intersection shall have a turning signal." to convey that every single traffic light should have a turning signal.

1.3.3 Example Requirement Statement

The following requirement statement includes problems related to all three defects listed in [1], i.e. deletion, generalization, and distortion: "For all servers, when a server experiences downtime, a report to the system shall always occur." In this example, deletion is present, since the process of reporting is ambiguous unless the following are defined beforehand: Who is reporting to whom, what is being reported, when, and how is it being reported? This example also contains generalization, as indicated by the use of the universal quantifiers "all" and "always, which may include special cases where the action is not required, or is otherwise impossible. Lastly, the example above also contains distortion, since it describes an event, which in this case is when a server experiences downtime. It does not describe how the report is initiated, and how it ends as a result of this action.

1.3.4 Do I agree that distortion "often appears in domains with an extensive technical language"?

I agree with the author of [1], who suggests that distortion "often appears in domains with an extensive technical language". It is natural for a human to describe a process conditional on some event happening, and it is also human to forget to elaborate, or to assume that the reader would automatically understand what the writer is trying to say. An example requirement statement with distortion is: "In emergency mode, when the hardware is fixed, the traffic light system should enter normal mode." This example suffers from distortion because the process by which the system transitions from emergency to normal mode is unknown to the reader from this statement alone.

2 Understanding the Customer: What Do We Know about Requirements Elicitation? review

2.1 Main contributions

The purpose of this paper is to outline the most common software requrements elicitation techniques used in the industry. The authors analyzed the overall trends of multiple studies related to requirements elicitation. According to the authors, they looked at two papers before, which attempted to do the same thing as what the authors wanted to accomplish, but the authors of the previous papers included a wide range of various elecitation strategies. Instead, this paper focuses more specifically on one-on-one scenarios, and draws some general conclusions from them. The paper discusses the benefits and drawbacks of structured and unstructured interviews, and also explores other non interviewing techniques such as protocol analysis, "contrived techniques" such as card sorting, and textual laddering. The authors then express what they found to be the most effective combination of techniques for requirements elicitation, which is an interview with some stucture, i.e. some general questions to guide the customer, that can be combined with some non interviewing techniques if specific information is required.

2.2 Criticisms

The article does a fantastic job overall of summarizing data from multiple studies related to requirements elicitation, however, some criticisms can be made. Although the paper is focused on software requirements elicitation, the authors of the paper also looked at studies they deemed to be "related fields" such as marketing. This raises some questions about the conclusions the authors are drawing from the data, and how applicable it is to software requirements elicitation. The article also explicitly states that "when we try to abstract a general conclusion from multiple studies, we find that each study has measured completely different effects of the elicitation approaches. This makes it hard to summarize multiple studies in a straightforward way." Although the authors claim to be doing this to find some broad conclusions, a lot of detail is lost that may contain some interesting information, in an area they originally were focusing in on. The authors also mention that their measurements individually don't give a complete picture, and that by comparing such a large amount of variables, they've lost the ability to combine the data statistically, which is arguably one of the more rigorous approaches for extracting conclusions from the results.

2.3

2.3.1 Contrived Techniques

The "contrived techniques" mentioned by the authors are as follows:

1. mock-ups
2. scenarios
3. storyboarding
4. card sorting
5. textual laddering

The author defines "contrived techniques" as a process that asks users to "engage in some kind of artificial task to help elicit information". According to this definition, card sorting is a contrived technique, since the user is asked to arrange some concepts or entities from the problem domain into groups. This fits the description of an "artificial task" that helps with the elicitation of information.

2.3.2 Unstructured vs. Structured Interviews

According to the authors, "interviews with some stucture are preferable to completely unstructured ones." In their view, the main benefit of structured interviews is that more customer needs tend to be identified during structured interviews. They also mention that the experience of the developers doing the interviewing did not affect the amount of customer needs identified for the interviews with some structure.

2.3.3 Main Technique for Elicitation of Requirements

According to the authors, the main technique for requirements elicitation is interviewing. They suggest that non interviewing techniques, including "contrived Techniques" can be used to improve the efficiency of a structured interview, when specific types of information are required.