

**AMBIGIOUS REQUIREMENTS**



**May 10, 2017**

**Muhammad Waseem**

**153182**

**AMBIGIOUS REQUIREMENTS**

Software requirements specifications (SRSs) need to be precise and accurate, to be self-consistent, and to anticipate all possible contingencies. The overwhelming majority of requirements specifications are written in natural languages, albeit often amplified by information in other notations, such as formulae and diagrams. Only occasionally, one finds a completely formalized SRS using very little natural language, except as commentary. Virtually every conception for a system is written in a natural language. Virtually every request for proposal is written in natural language. A recent on-line survey of businesses requiring software, conducted at University di Trento in Italy and available at eprints.biblio.unitn.it [, shows that a majority of documents available for requirements analysis are provided by the user or are obtained by interviews. Moreover, 71.8% of these documents are written in common natural language, 15.9% of these documents are written in structured natural language, and only 5.3% of these documents are written in formalized language. However, we all know that natural language is so imprecise, so ambiguous, and so inherently so! Thus, even if the authors of an SRS are trying to avoid ambiguities, unintended ambiguities sneak up behind us in the SRS.

**Detecting Ambiguity**

When ambiguity exists in writing, there are different possible underlying causes of the ambiguity. These kinds of ambiguity are outlined below. Note that in the academic literature there are a few more than are mentioned here, but they start to blur the distinction between the different forms, and so they have been omitted in the interest of keeping the conceptual differences clear.

## **Lexical Ambiguity**

[Lexical ambiguity](http://en.wikipedia.org/wiki/Polysemy) occurs when a word has several possible meanings, resulting in a sentence having multiple possible interpretations. In the sentence:

*I like writing.*

…it’s unclear whether or not the author is referring to the act of writing (the verb) or the result of writing (the noun). The best way to deal with lexical ambiguity is to use a word that does not have multiple meanings, or to rephrase the sentence such that the word now has only one possible meaning, for example:

*I like writing stories.*

In this case, however, we’ve narrowed the statement to referring to the act of writing stories, so it’s not a perfect match (though it may be what the author intended). If we wanted to remain general, we would have to say:

*An activity I like is writing.*

## **Synonyms**

In English class, we were often taught to use synonyms to describe the same word; this helped make our writing more interesting and varied.

Unfortunately, when it comes to technical specifications, using a synonym Y*Y* to describe a word X*X* can be a source of confusion. For example:

*The background task constructs a list of words for use by the game engine. The process then uses the list of words when creating anagrams for the user.*

In this situation, it’s unclear whether “process” refers to the previously mentioned background task, or a distinct process running the game. This form of lexical ambiguity can be avoided by always referring to an object in the system the same way each time, so that there is no possible misinterpretation.

**References**

1. [**https://link.springer.com/chapter/10.1007/978-1-4615-0465-8\_2**](https://link.springer.com/chapter/10.1007/978-1-4615-0465-8_2)
2. **https://shane.io/2013/04/22/ambiguity-in-software-specs.html**