AIR university multan campus

**Software Engineering**

Muhammad Shahroz

#153163

Definition Of Ambiguity:

**Ambiguity** is when the meaning of a word, phrase, or sentence is uncertain. There could be more than one meaning.

Typically, it is best to avoid ambiguity in your writing. When you make statements that are ambiguous, you confuse the reader and hinder the meaning of the text. However, sometimes **ambiguity** is used deliberately to add humor to a text.

**Examples of Ambiguity:**

Sarah gave a bath to her dog wearing a pink t-shirt.

Ambiguity: Is the dog wearing the pink t-shirt?

I have never tasted a cake quite like that one before!

Ambiguity: Was the cake good or bad?

Did you see her dress?

Ambiguity: Is she getting dressed or are they talking about her clothes?

**Examples of Ambiguity in Literature and Speech:**

In "The Rose" by William Blake, there is enough ambiguity about what is meant by "rose," "sick," "bed of joy," and "worm," that the poem is open to many interpretations:

"O Rose thou art sick.

The invisible worm,

That flies in the night

In the howling storm:

Has found out thy bed

Of crimson joy;

And his dark secret love

Does thy life destroy"

From *Romeo and Juliet* by Shakespeare-Mercutio is dying from his wound, but he attempts to remain lighthearted. The word "grave" has an ambiguous meaning: Ask for me tomorrow and you shall find me a grave man.

**Ambiguous Requirements:**

Ambiguity is the property of being ambiguous, where a word, term, notation, sign, symbol, phrase, sentence, or any other form used for communication, is called ambiguous if it can be interpreted in more than one way. In either case, [ambiguous requirements lead to confusion](http://searchsoftwarequality.techtarget.com/feature/Writing-requirements-Common-sense-measures-for-success), wasted effort and rework.

Ambiguous functional requirements are any requirements that have any kind of ambiguity or have more than one type of interpretation. Any task in requirements that can have more than one correct output that is contingent on a different understanding of the task is ambiguous.

**Boundary conditions**

Boundaries between numerical ranges or date ranges are a common source of missing requirements. One requirement might describe what the system does if the amount of the sale is less than $100, while a second requirement describes the behavior if the amount is more than $100. But what happens if it's exactly $100? That's not defined. Similarly, if two requirements are written so that the endpoint of a range appears in both requirements, the expected behavior is ambiguous. Use the words "inclusive" or "exclusive" to make it clear whether the endpoints of the range are included or not.

**Synonyms and near synonyms**

Use specialized terms consistently throughout the project documentation. A requirements specification is not a place to creatively vary your language in an attempt to keep the reader's interest. If your project involves several terms that have similar but not identical meanings, put those into a glossary so everyone knows where to go for the exact definitions.

**Abbreviations *i.e.* and *e.g.***

These abbreviations are often misused. The abbreviation i.e. means "that is;" whatever follows is a complete list of items in the stated category. The abbreviation e.g. means "for example," so the items that follow are merely representatives of the complete set. These abbreviations are so often used incorrectly that I don't trust them in requirements. Use English words instead of Latin abbreviations to avoid any confusion.

**Adverbs**  
Adverbs are subjective and qualitative, and they inevitably result in diverse individual interpretations. Here's an illustration from an actual specification: "Generally incurs a 'per unit' cost…" But this requirement did not provide any indication regarding the conditions under which we would not incur a per-unit cost or what to do then. I don't know how to determine whether we've satisfied a requirement that says "Provide a reasonably predictable end user experience." I guess it doesn't have to be completely predictable.

**A/B**  
With rare exceptions, such as "input/output," avoid using the A/B writing construct, as in "feature/function." This can be interpreted in many ways. Is a feature the same as a function? Are we referring to features and functions? Features or functions? Features divided by functions? Sometimes this A/B notation means the author isn't really sure exactly what to say so he's leaving it up to all readers to interpret as they wish. This is an invitation to confusion.

It's important for requirements to be precise because our objective is clear and effective communication. Watch out for natural language expressions that can be read in a variety of ways.

**Example No. 1:**

**Math’s Problem:**

I will start with a simple example from a grade 1 class:

An ambiguous task can be as simple as dividing 8 in a half. What is half of 8? The correct answer us 4 but other answer like 0 and 3 may be the correct answer to this question as well. If depends on how you define “a half”.

**Example No. 2:**

**Online Brain Game for Software Testing:**

From any word related to software testing, create another word randomly by adding 3 letters. As a result, the following is one of the online questions created by the application:

Cross out 3 letters so that a familiar software testing vocabulary word will remain bcdecdeug.

To find the correct answer, note that you need to cross out 3 letters as many times as they can be found in the word above. Cross cde two times and you will be left with the word.

**Conclusion:**

If we want to decrease the ambiguous requirements we should write complete requirements and also use precise language that communicates information across domain to reader**.** We need to monitor the effectiveness of our communication, and balance that with the amount of time

**References:**

1. [**http://softschools.com/examples/examples/ambiguity\_examples/259/**](http://softschools.com/examples/examples/ambiguity_examples/259/)
2. [**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)
3. [**http://searchsoftwarequality.techtarget.com/tip/Ambiguous-software-requirements-lead-to-confusion-extra-work**](http://searchsoftwarequality.techtarget.com/tip/Ambiguous-software-requirements-lead-to-confusion-extra-work)
4. [**http://www.cigniti.com/blog/how-harmful-can-be-ambiguous-software-requirements/**](http://www.cigniti.com/blog/how-harmful-can-be-ambiguous-software-requirements/)
5. [**http://searchsoftwarequality.techtarget.com/tip/Ambiguous-software-requirements-lead-to-confusion-extra-work**](http://searchsoftwarequality.techtarget.com/tip/Ambiguous-software-requirements-lead-to-confusion-extra-work)
6. [**https://shane.io/2013/04/22/ambiguity-in-software-specs.html**](https://shane.io/2013/04/22/ambiguity-in-software-specs.html)