**Ambiguous Requirements:**

Many software requirements suffer from ambiguity. Ambiguity means that a single reader can interpret the requirement in more than one way or that multiple readers come to different interpretations. In either case, ambiguous requirements lead to confusion, wasted effort and rework.

**Examples**

**We are precise**

Avoid using general descriptors for characteristics or amounts. We don´t like to use various, some, any, a lot, every or up to. We also avoid using vague words for behaviors like possibly, probably, and usually. We also like to use the industry standard amount for things when we can or the exact pixel amounts for front-end elements.

**Engage Developer in Requirements:**

It is important to involve both the developers and testers in the requirements definition process as they are involved in the actual development & testing of the software. They can understand and analyze any ambiguous requirements if they are involved early in the requirements definition phase.

**How to avoid ambiguous requirements:**

**Get the context:**

Don’t depend on external context. The people reading the requirements may not have been in the same meeting that you were and the developers don’t generally have the kinds of access to the clients that the writers of requirements do. A missing context can create a vital flaw in the requirements. Is that large screen specification for a region in China that the majority of screens are still only up to 720px? That would make all the difference for this requirement.

**We command line**

When writing the requirements that are clear, always use the present with indicative mood. In other words, it happens now because I command you to do it. Good requirements use must, shall, and can. They do not use could, should or may.