**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**](http://searchsoftwarequality.techtarget.com/feature/Writing-requirements-Common-sense-measures-for-success)**, wasted effort and rework.**

**Example#1:**

* **Negative requirements**  
  **Negative, or inverse, requirements state what the system will not do. Here's an example from an actual project: "All users with three or more accounts should not be migrated." Try to rephrase negative requirements into a positive sense: "The system shall migrate only users having fewer than three accounts." When changing a negative requirement into a positive one, you often need to insert the word "only" to clarify the conditions that permit the system action to take place. Double and triple negatives are especially confusing; avoid them in all situations**.

**Example#2:**

* **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.**

**Example#3:**

* **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.**

**THE END**