Applying Ambiguity Reviews
To
Already Developed Components

© Richard Bender
Bender RBT Inc.
17 Cardinale Lane
Queensbury, NY 12804
518-743-8755
rbender@BenderRBT.com

The usual process is to perform ambiguity reviews as the specifications are being written. Ideally, a given requirement should be reviewed within 24 hours of being written in order to provide timely feedback. Unfortunately, the reviews are sometimes done late in the project. The issue then is to tailor the process to be added value even though code is nearing completion and testing is well underway.

When the reviews are done as the requirements are being written the analyst goes back and clarifies the specifications to resolve all of the issues identified. When the reviews are done late in the project the focus shifts a bit depending on the issues uncovered. The overall goals are two fold. The first is to ensure that the implemented system is close enough to the intent the analyst had when they wrote the specification. The second is to enhance the existing test cases. The tactic to addressing the issues identified has to be tailored to the type of issue.

Explicit Missing Case

Sometimes it is obvious what case is missing. The most common example is the "dangling else". The "go right" path is defined but not the "go wrong" path. Another example is where there is a list of choices which need to be accounted for and only some of them are explicitly defined. For example, you know a variable may be set to A, B, or C. The specifications only define what to do for A and B. If you find a specific missing case then you do two things. First, design a test to cover that case and see what the behavior of the system is. Second, review the observed results with the analyst to verify that that behavior is acceptable.

Unclear Alternatives

Sometimes a rule in the specification can have more than one alternative interpretation. A common example is an ambiguity of reference. For example "add A to B, this number must be positive". Which number: A, B, or the result? In this situation you might have to design a number of tests to determine the behavior, doing one for each interpretation. Again you go back to the analyst with the results for validation.

Insufficient Detail On The Expected Results

Sometimes the case is clear from the input side but not sufficiently detailed on the outputs. For example, "an attempt at unauthorized access will cause the system to generate an audit record and lock down the terminal". What is in the audit record? In this you create the test and check the contents of the audit record. You then review those with the analysts for validation.

Just Too Unclear To Do Anything

Sometimes the rules are so unclear that nothing can be done unless the analyst explains it more detail. For example, “the appropriate fees and service charges will be applied depending on the customer type, account balance, etc”. You have no chance at guessing enough to even build exploratory tests to deduce the application’s behavior. Your only recourse is getting clarification from the analyst or developer. You then design tests based on the clarified rules and review the results with the analyst.