Q&A about imprecise requirements and rich definitions

1. If you permit imprecision in the requirements, how does the QA group distinguish between issues, which are, not know vs. poorly written requirements? **Answer:** That is the purpose of using some form of marker for intentional imprecision. The marker invokes the social contract I spoke of so the imprecision will be resolved when more information is available and a choice is made.

2. At what point should imprecise information be specified? **Answer:** If you mean, "be resolved" (i.e. detailed alternatives identified), then this should be done as early as possible. Often you will find acceptable definitions in hours, though some may take longer than you expect. Occasionally, you will be unable to create a definition that satisfies all major stakeholders because there are deep disagreements about the product or project that are not obvious in the imprecise descriptions. If such conflicts exist, you need time to resolve them or should stop wasting resources and cancel the project. [The devil may be in the details.]

3. When do you start to talk about quality in solution? **Answer:** Required quality attributes drive architectural decisions. If the architecture is being developed, then qualities should be precisely defined early. If the architecture is a given, then check to see if the architecture supports them, but they don't need to be precisely defined because there are few choices.

4. If the supplier knows considerably more than the specifier, it seems there is need for the specifier to get more detail to avoid developing a solution that fails to meet expectations. Please comment**. Answer:** I agree. Since the specifier does not know much, the supplier should not expect to find many written details. The supplier needs to help the specifier discover details by exposure to similar systems, prototypes, and/or early increments.

5. Could you explain better the relation between plain definitions and risk? **Answer:** Most plain definitions (those found in dictionaries) are imprecise, but fine for most human communication. However, imprecision does not work in the technical communication that specifies a system. The risk is that if nothing is done before coding to make the meaning precise, then the programming or data definition will create the precision, and it will very likely be wrong. If it is obviously wrong, you will find it in test. Otherwise, you will find it in production – perhaps with a large price tag.

6. When do you begin risk analysis? **Answer:** During early project planning. For safety critical products, the first question to ask is, "How many ways can we kill or injure our users and how do we convince ourselves (and others) that this either can't happen or is very unlikely when our "simple" directions are followed?" I have found this to be a good question to ask on most projects along with "What should I be afraid of on this project and who can help me answer this question?"

7. So you try to get to a "formula" that specifies what the marketing dept really wants? **Answer:** For some phrases (“year-end bonus” as a derived value) Yes. For most others No. Just as natural language combines nouns, verbs, adjectives, and adverbs to create meaning. Rich definitions combine seven precise definitional patterns to create precise mean.

8. Action contracts come from what "methodology"? **Answer:** Neither Action Contracts nor any other rich definitional pattern is associated with a particular methodology. Since vague terms appear in every development methodology, rich definitions can and should be used everywhere. The use of Action Contracts at the system level was inspired by Dr. Bertrand Meyer’s “Design by Contract™” which uses similar ideas at the component level. Meyer is the developer of the Eiffel Method.

9. What is a good way to know how to "chop up" the precise versus less precise when it is interconnected? **Answer:** The short answer is – make precise those phrases that need to be precise so someone can do their job. All others can be less precise. The “consumers” of the phrases can tell you their needs. The long answer is – consider “Accurately identify potential customers …” Developers need to understand “identify … customers” precisely to create a program that works. They also need to understand precisely what “potential customers” means, as this will become a conditional expression in their code. Finally, both testers and developers need to understand precisely the meaning of “accurately identify” to decide if the program is working correctly.

10. Is there a strong relationship between the difficulty rating and the length of discussion to arrive at the definition? **Answer:** Yes, unless your team likes to chat. Precise definitions leading to long and heated discussion are well worth doing because everyone was not “on the same page” at the beginning. Occasionally, it may turn out to be impossible to agree. In such cases, you cancel the project happy in the knowledge that few resources were wasted before uncovering this conflict.

11. Agile allows more general specs up front, however a test perspective requires more detailed specs to test early on. How should this be handled early on when creating test cases based on those specs? **Answer:** When the details are discovered, both tests and code can be completed. If there is unresolved imprecision, then neither code nor tests can be completed. Both complete code and complete tests need precise understanding. There is no way to complete effective test design with imprecise specs.

If you have other questions or I did not answer these clearly, send an email to david@clearspecs.com