Deeper

- Why must any definition of quality be measurable?
  - *If a product has quality requirements, it must be possible to express those requirements as testable statements.*
  - *If we want to improve the quality of our products, we must be able to measure the quality of our products.*

You can't optimize what you can't measure.

Deeper

- What bottom lines do customer facing quality dimensions translate into?
  - *User productivity and Total Cost of Ownership.*
- What bottom lines do producer facing quality dimensions translate into?
  - *Developer productivity, and total cost of development and support.*

Deeper

- Why are we talking about definitions of quality in the context of "requirements"?
  - *If we need a product to be of a particular quality, we must be able to define (and measure) the product characteristics that will give it that level of quality.*

Deeper

- Are developer-facing characteristics legitimately sources of requirements?
  - *It is a requirement that the software be built, supported and maintained. Factors that bear on the ease of development and maintenance can make or break a product.*
  - *It is common to require internal features that will enable future features or cost savings.*

Deeper

- Are there conflicts between some of the elements of quality?
  - *Conflicts, while not inevitable, are common:*
    - *efficiency can conflict with modularity*
    - *robustness can conflict with simplicity*
    - *McConnell table 20-1 shows such trade-offs*
- How to optimize multi dimensionl quality?
  - *Treat it like a linear programming problem:*

$$Qual_{prod} = \sum_{i \in \text{qual}} weight_i \times qual_i$$

Deeper

- With so many dimensions of quality, how can we define the quality goals for a particular project?
  - "*Once you start identifying your users and what they find valuable, you can determine how to quantify that into release criteria, so you meet a minimum standard of project quality for your specific project.*"

Deeper C_1

- Why are whole-system acceptance tests best performed by a distinct group?
  - *Socially, development groups are far from impartial. They are psychologically inclined to believe their S/W is already perfect.*
  - *Pragmatically, development groups often lack system view, and the tests to be run at the system level are very different than those to be run on individual components.*

Deeper C_2

- Why is unit testing best performed by development?
  - *Pragmatically, they have the knowledge and tools to most effectively exercise their own code. Moreover, bug isolation and fixing are done much more efficiently within development than when divided between independent groups.*
  - *Even though developers may be biased against the believe that their software has bugs, it makes sense to make them responsible for delivering code of measurable quality.*

Deeper C_3

- Testing does not improve the quality of a product?
  - *True … in that it only attempts to measure quality, and cannot directly improve it.*
  - *False … measuring quality directs our attention to sources of non-quality, and addressing these is key to improving the quality of a product.*
  - *So we should say that "testing <u>alone</u>" does not improve the quality of a product.*

Deeper C_5

- Why should the process assurance function be performed by an independent organization?
  - *Politically, having development do their own process assurance is like having the foxes guard the hen-house.*
  - *Pragmatically, process assessment is a very different activity than development, requiring both different skills and attitudes.*

Deeper C_6

- Why should product and process metrics be collected and reported by an independent organization?
  - *Politically, to ensure that the measurement and reporting is unbiased.*
  - *Pragmatically, because collection, analysis, and reporting of process data is very complex. Making sure that we are collecting the right data, and reporting it meaningful ways requires unique skills and training.*

Deeper? B_1

- Why is it important to find early-stage mistakes as quickly as possible?
  - *Errors in requirements or architecture can result in large amounts of code that does the wrong thing. This is called "defect amplification".*
  - *Errors are much less expensive to correct before they have had an opportunity to affect down-stream work.*
  - *Errors found sooner have less opportunity to cause damage.*

Deeper

- Does it make sense to make a distinct group responsible for customer advocacy?
  - *There should be customer focused people in every part of the organization (development, Q/A, support, sales, etc). If there aren't, that is a problem to be fixed.*

Deeper

- Why should engineering be responsible for Requirements Analysis?
  - *The primary input gathering is not particularly technical and can be gathered by anyone who is familiar with the problem domain.*
  - *The analysis process, however, should be done by people who can fully understand the implications of each requirement.*
  - *Engineering will be tasked with turning the requirements into code, so it makes sense to give engineering the opportunity to review and question the requirements before they are finalized.*

Deeper

- Why should engineering be responsible for Architectural and Design Reviews?
  - *The architectures and designs that are to be reviewed are highly technical artifacts. They (and any potential flaws) are unlikely to be fully understood by anyone who has not been trained in the language and problem domain.*
  - *However, documentation, support and testing personnel often obtain great value from attending such reviews.*

Deeper

- Why should engineering be responsible for Configuration Management?
  - *Engineering designs and constructs all of the components, and so is the obvious source for build procedures.*
  - *Engineering is the primary consumer for a standard set of build tools, so it makes sense for engineering to select, maintain, and distribute them.*
  - *As has already been discussed, the build scripts really are a part of the code, and must be maintained by the people who maintain the code.*
  - *Engineering makes all of the changes to code, so it makes sense for them to have the primary responsibility for version and change management.*

Deeper

- Why should engineering be responsible for bug management?
  - *Engineers are better prepared to assess the likely consequences of a bug than others who do not understand how the code works.*
  - *Engineering should obviously be responsible for tracking down and fixing the bugs.*
  - *The people who best understand the bugs are the best able to develop regression tests to ensure the fixes work.*
  - *Root cause analysis can only be done by people who understand both the problems and the process that gave rise to them.*

Deeper

- Why might statistical Q/A analysis lead to quicker, more dramatic improvements?
  - *Boehm's article told us that:*
    - *80% of rework comes from 20% of defects.*
    - *90% of down-time comes from 10% of defects*
    - *80% of defects come from 20% of modules.*
  - *If we can use real data to guide us to the few areas that are causing the most problems, our investments in improvement will yield the greatest results.*