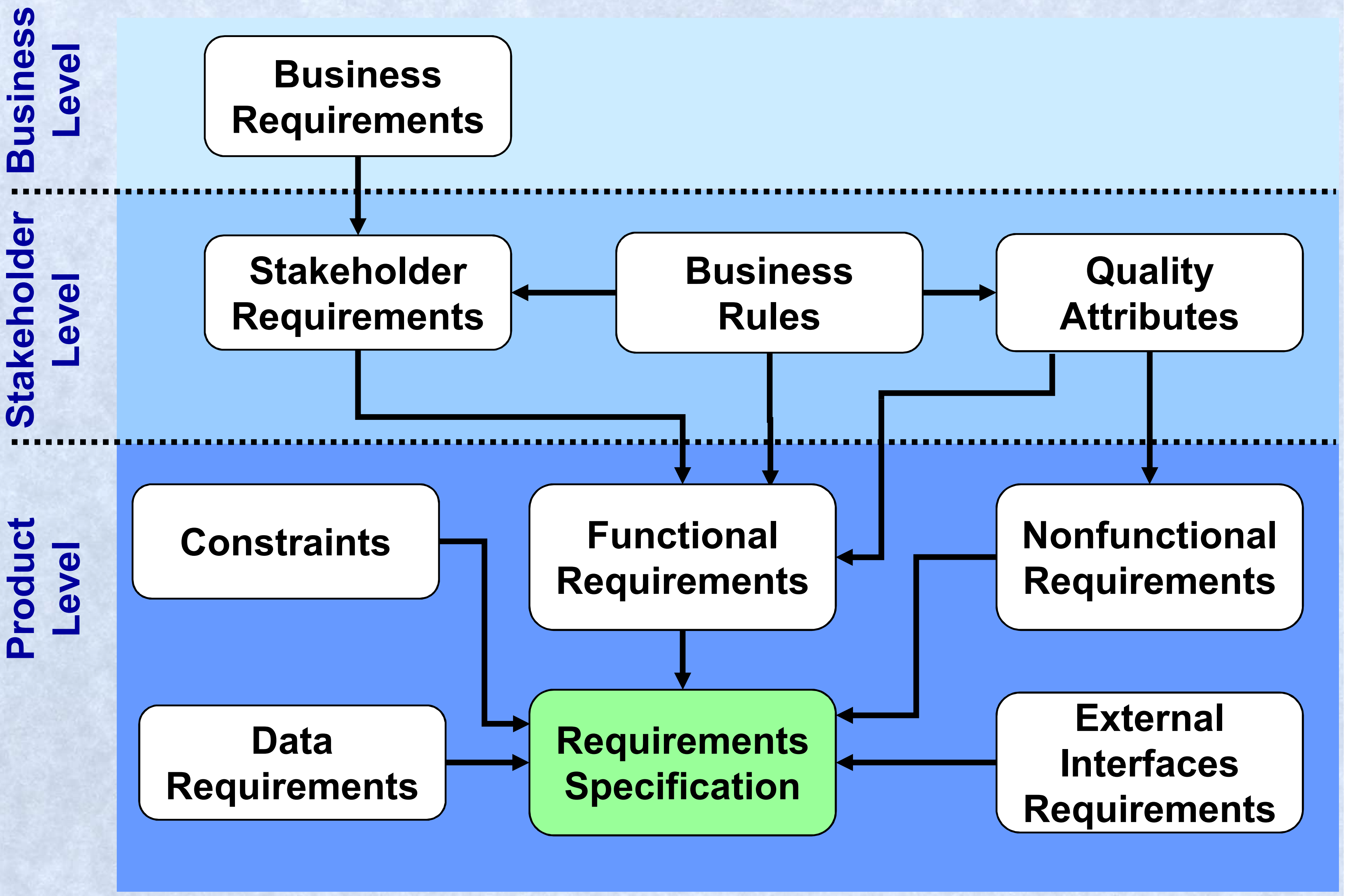
What Are Requirements?

A requirement can be defined as:

- A statement that identifies a capability, characteristic or quality factor of a system in order for it to have value & utility to a user. [Young-01]
- A user need or a necessary feature, function or attribute of a system that can be sensed from a position external to that system. [Allen Davis in Wiegers-04]
- A specification of what should be implemented:
  - A description of how the system should behave
  - A system property or attribute
  - A constraint on the development process of the system. [Sommerville-97]

Levels of Requirements Information



Why Are Requirements Important?

"The hardest part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements, including all of the interfaces to people, to machines & to other software systems. No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later." [Brooks-95]

Writing Testable Requirements

Tips for writing testable requirements include:

- Use short, direct, complete sentences
- Make requirements internally & externally consistent
- Remove ambiguity
- Make requirements measurable
- Make requirements finite
- Include testers in the requirements peer reviews
- Define a testing strategy for each requirements as soon as it is specified
- Trace requirements to test cases

Short, Direct, Complete

Use short, direct, complete sentences:

- A requirements specification is not the *Great American Novel*
- Use active, not passive voice
- Include only one requirement per requirement
- Remove adverbs
- Question adjectives – do they add value?
  - Are they making the requirement more specific?
  - Are they making the requirement more understandable?

Consistent Requirements

Externally consistent product-level requirements trace to & match their source:

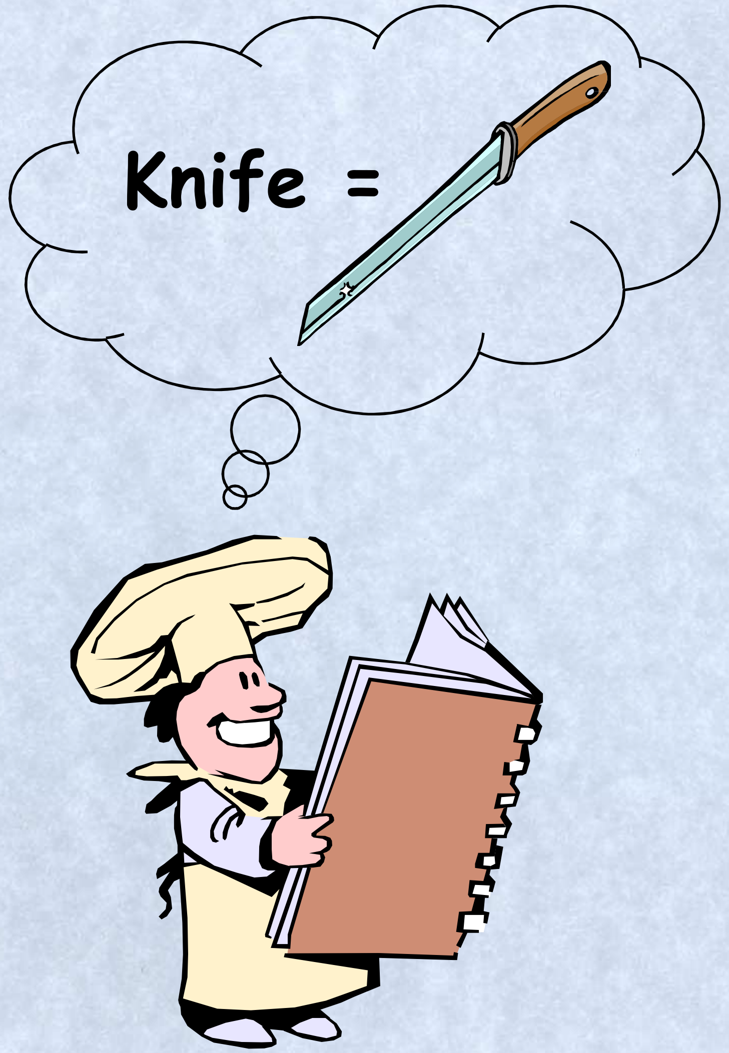
- Business needs
- Stakeholder functional requirements
- Business rules
- Quality attributes

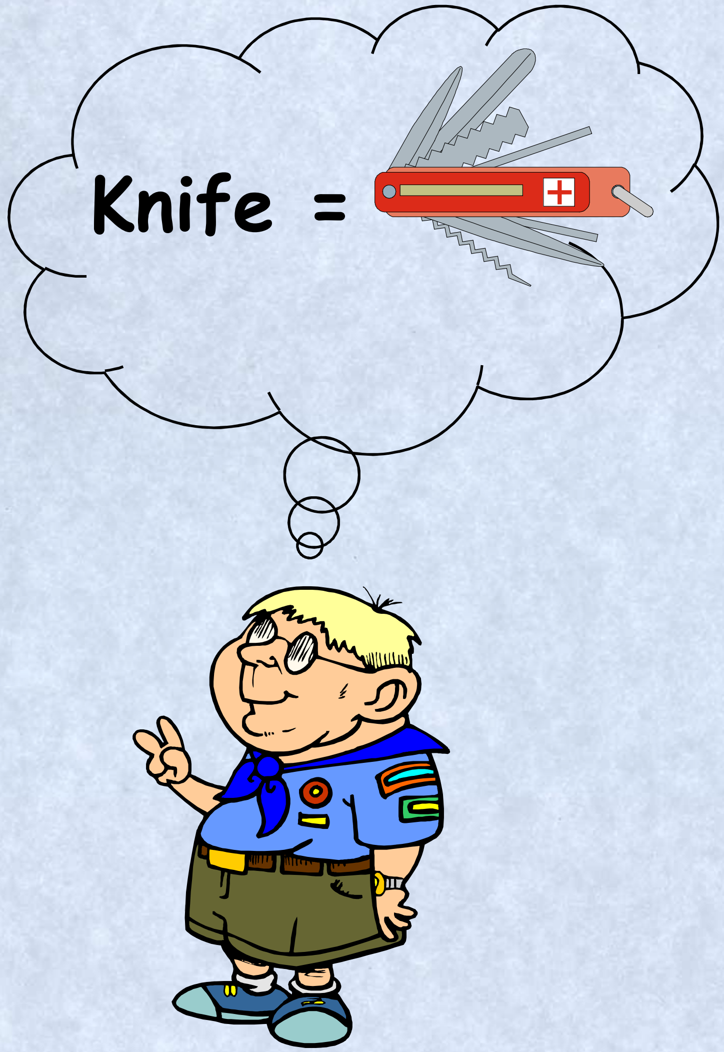
Make product-level requirements internally consistent by:

- Ensuring that they do not contradict one another
- Stating the requirements only once & referenced elsewhere
- Using consistent terminology & define it

Ambiguity

Think of a knife – are you picturing your knife?





Tips When Reviewing for Ambiguity

Thing to look for:

- Do a text search:
    - Words ending in “ly” (quickly, user friendly, clearly, easily)
    - Words containing “ize” (minimize, maximize, optimize)
    - Other key words (flexible, accommodate, safe, fast, easy, sufficient, small, large, improve, reduce, etc.)
    - Pronouns (he, she, it, they, them …)
- Turn on the grammar checker & look for passive voice wording

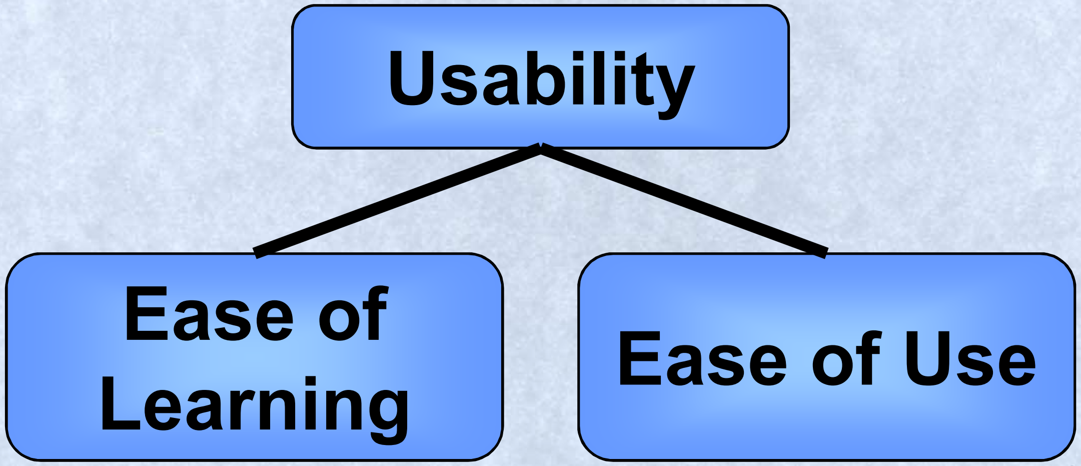
Measurable Quality Attributes

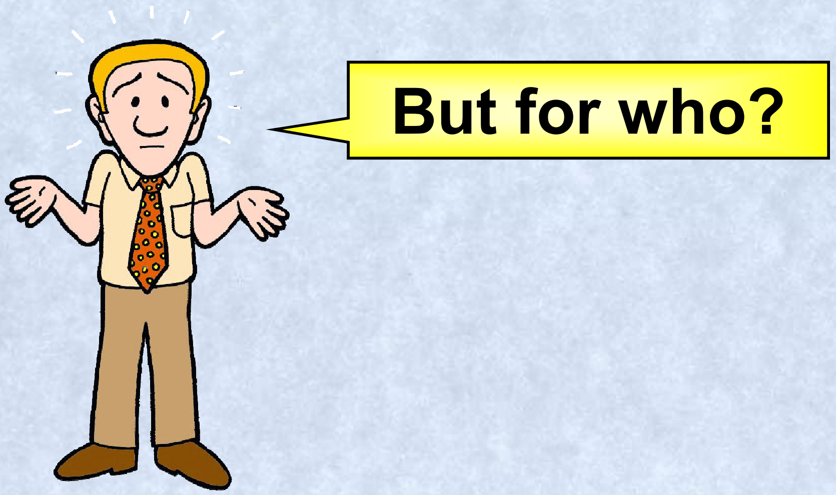
Quality attributes are the characteristics that define the software product's quality, for example:

- Usability
- Reliability
- Availability
- Performance
- Efficiency
- Security (Integrity)
- Safety
- Interoperability
- Accuracy
- Installability
- Flexibility
- Robustness
- Maintainability
- Reusability
- Testability
- Portability

Developers Need More Guidance

A quality attribute like "usability" doesn't provide the tester (or developer) with enough information.





Usability Requirements - Examples

Usability requirement for Pay-at-the-Pump:

- The system will prompt the customer for all inputs in a single session
- Less than 0.2% of customer’s with valid credit cards will require more than 3 attempts to be able to swipe their credit card without error
- >=99% of trained gas stations managers will be able to perform end-of-month reporting without referring to the user manual or online help
- Updates to the customizable gas cost & tax rate values will require the navigation of no more than 3 screens per value being updated

Measurable Nonfunctional Requirements

Guidance examples for specifying measurable nonfunctional requirements:

- Reliability: [mean time] for a defined [product or component] to experience defined [failure type] under defined [conditions]
- Availability: [percentage] of defined [time period] a defined [product] is [available] for its defined [tasks]
- Security: [probability] for a defined [product] to [handle (e.g., detect, prevent, recover from)] a defined [attacks] under defined [conditions]

Measurable Nonfunctional Requirements (cont.)

- Performance:
  - Throughput: [quantity] of defined [work units] which can be successfully handled per [time unit]
  - Capacity: [quantity] of defined [entities or components] that can be [task, action or state] under defined [conditions]
  - Response time: [mean time or maximum time] of a defined [response] to a defined [event]

Measurable Nonfunctional Requirements (cont.)

- Usability:
  - Ease of learning: [effort or time] required to become proficient in performing a [task or action] under defined [conditions]
  - Productivity: [quantity] of defined [tasks or actions] that can be performed using the product per [time unit] under defined [conditions]
  - Response time: [mean time or maximum time] of a defined [response] to a defined [user stimulus]
  - Error rate: [quantity] of user [error type] mistakes experienced per [time unit] under defined [conditions]
  - Likeability: [quantity or percentage] of users that report liking the product

Finite Requirements

The requirement should not be stated in an open-ended manner.

- Look for words like:
  - “all”
  - “most”
  - “every”
  - “throughout”
  - “sometimes”
- Look for phrases like: :including but not limited to”

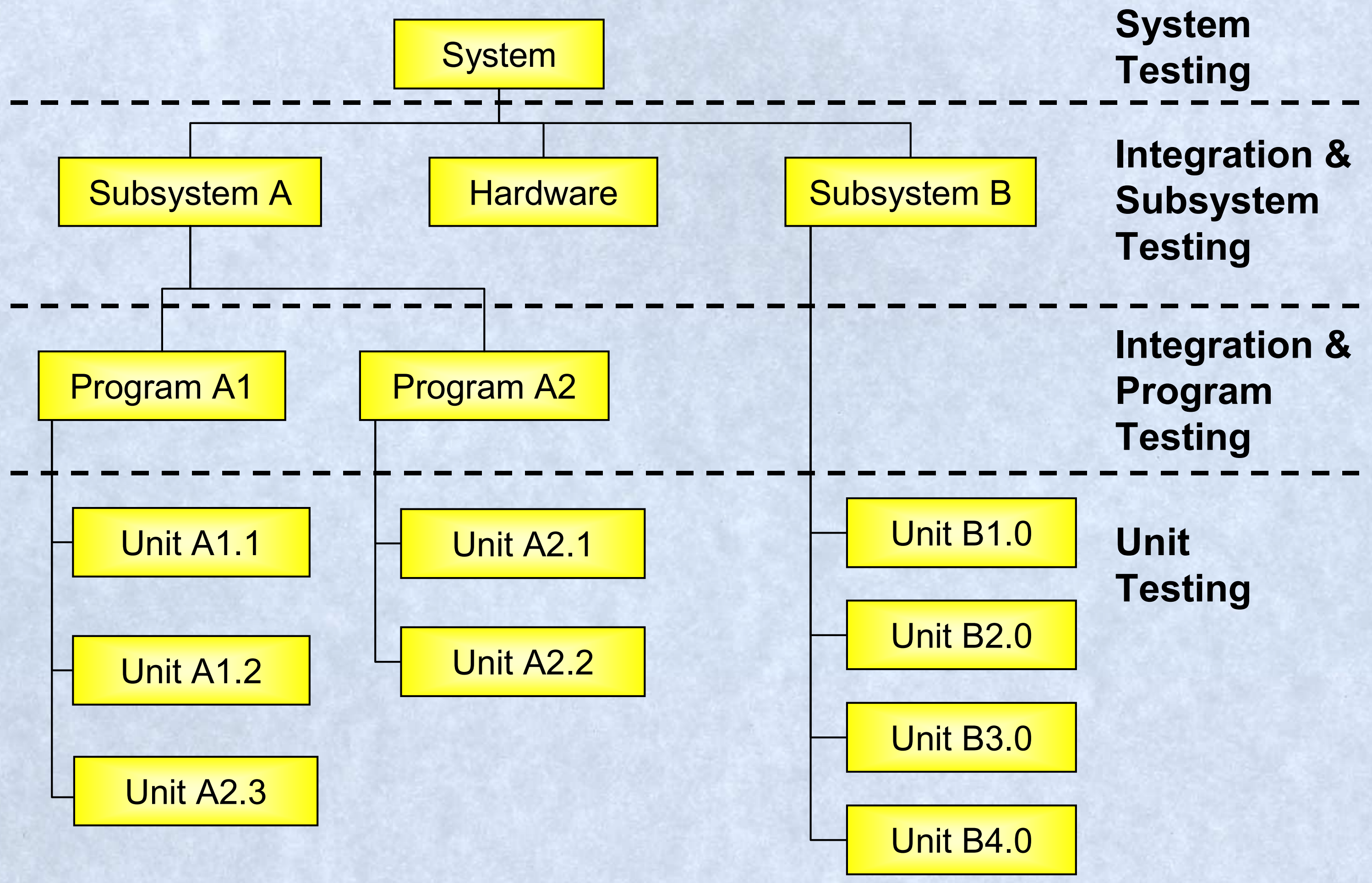
Tester Participation in Peer Reviews

Testers are excellent candidates for inclusion in requirements peer reviews:

- Vested interest in assuring the quality of the requirements -- the higher the quality, the easier their job is (WIIFM)
- Perspective of “how can I break it”
- Adds diversity to the review team resulting in more unique defects being identified – For example:
  - Excellent job of identifying what’s missing in the error handling sections of the requirements
  - Reviewing the testability of the requirements

Define a Testing Strategy

Define a testing strategy for each requirements as soon as it is specified.



Test Matrix - Example

Decide how to evaluate each requirement.

Requirement #	Requirement	Test Level	Test Strategy
R00103	The system shall respond to all user commands and data entries within 3 seconds	System	Black-Box - Demonstration
R00200	The software shall store all currently active transactions in memory to allow access if the storage media fails	Unit	White-Box
R00397	The source code shall be written in C++	Peer Reviews of Code	Inspection
R00560	The system shall calculate sales tax at current tax rates	Unit System	White-Box Black Box - Analysis
R00050	The system shall work with all credit cards issued by all banks	Not Testable	

Writing Testable Requirements

Testable requirements:

- Have only one requirement per requirement
- Are specified in short, direct, complete sentences
- Are internally & externally consistent
- Are unambiguous
- Are measurable
- Are finite
- Are peer reviews by testers
- Have their testing strategy defined early
- Trace to test cases