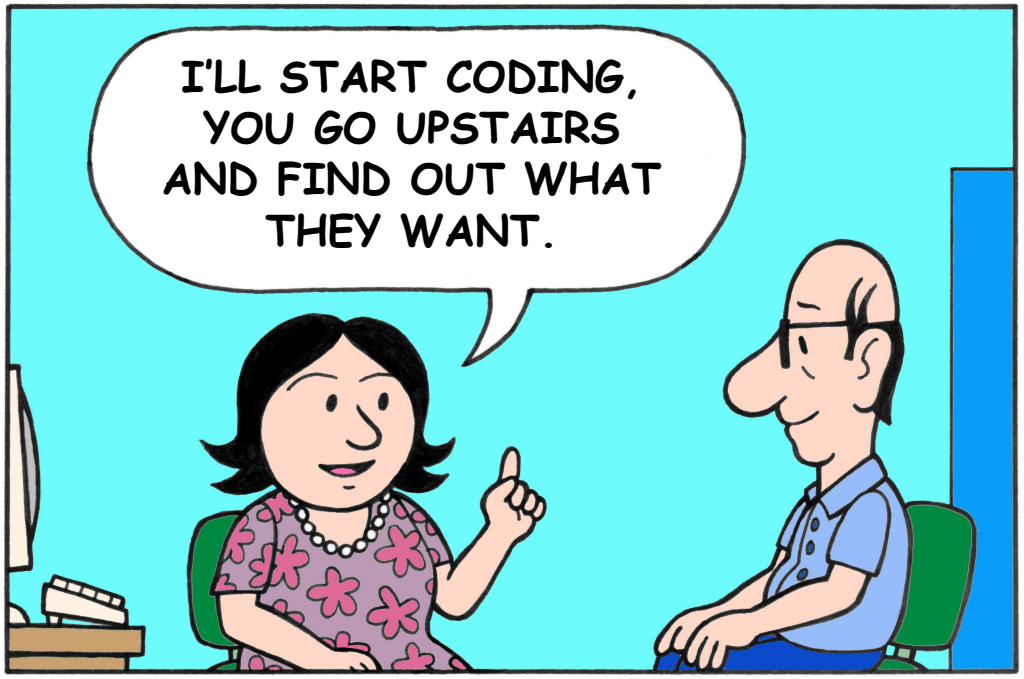
Writing Testable Requirements



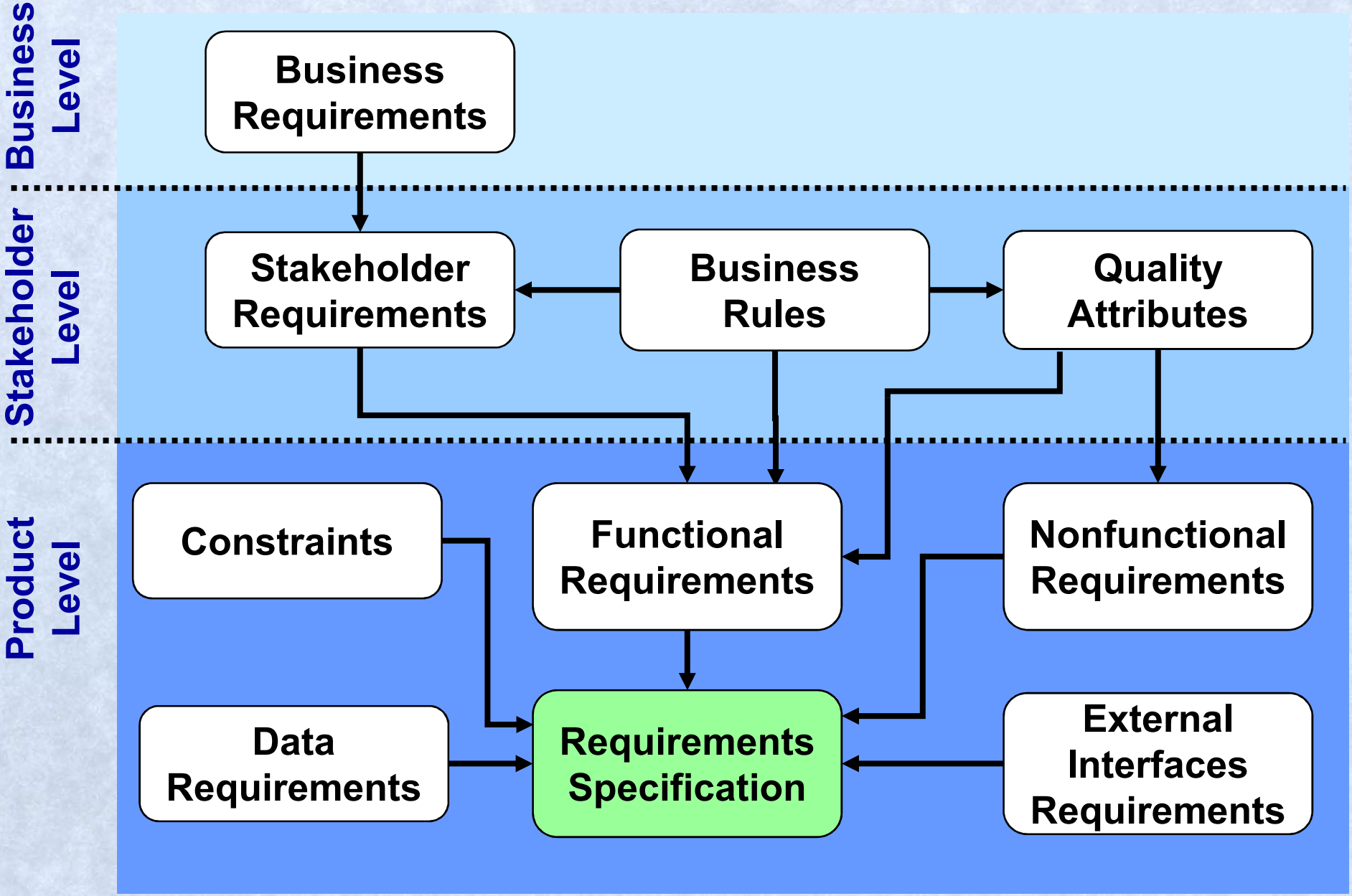
the WestfallTeam

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

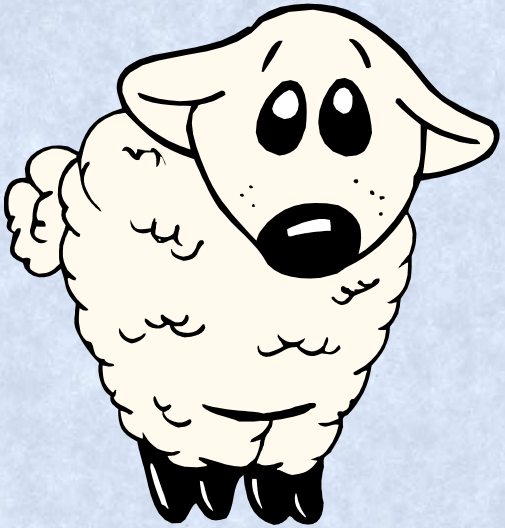
Make product-level requirements internally consistent by:

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

Natural Language is Ambiguous

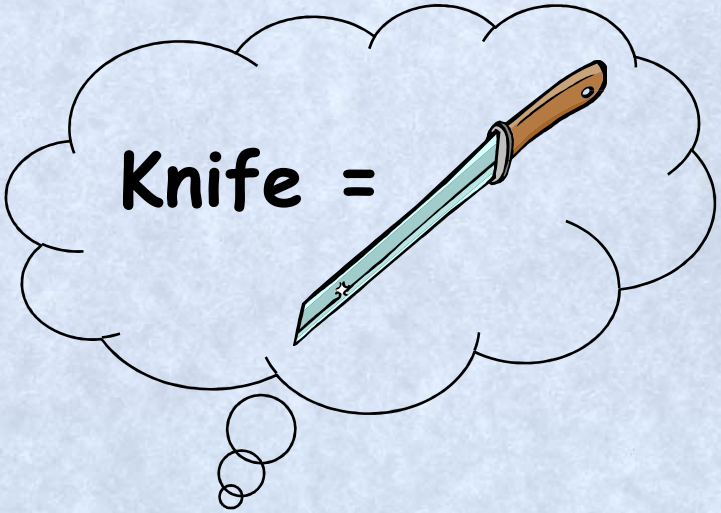
A simple nursery rhyme:

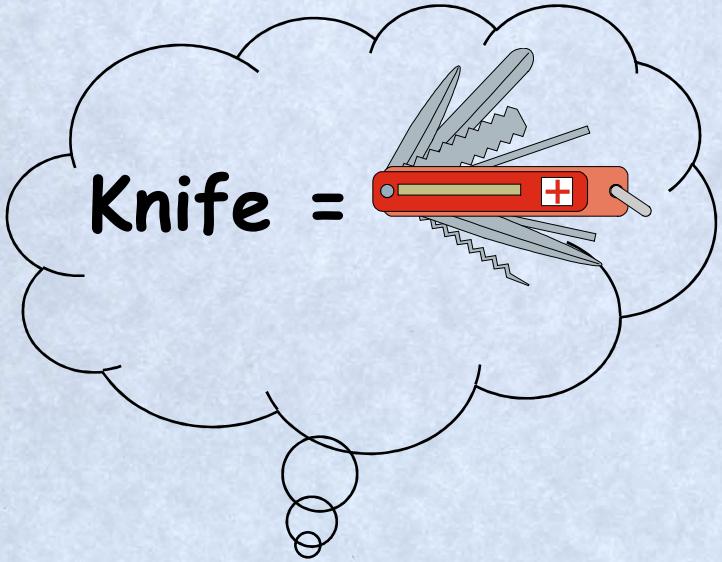
- **Mary** had a little lamb
- Mary **had** a little lamb
- Mary had **a** little lamb
- Mary had a **little** lamb
- Mary had a little **lamb**



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

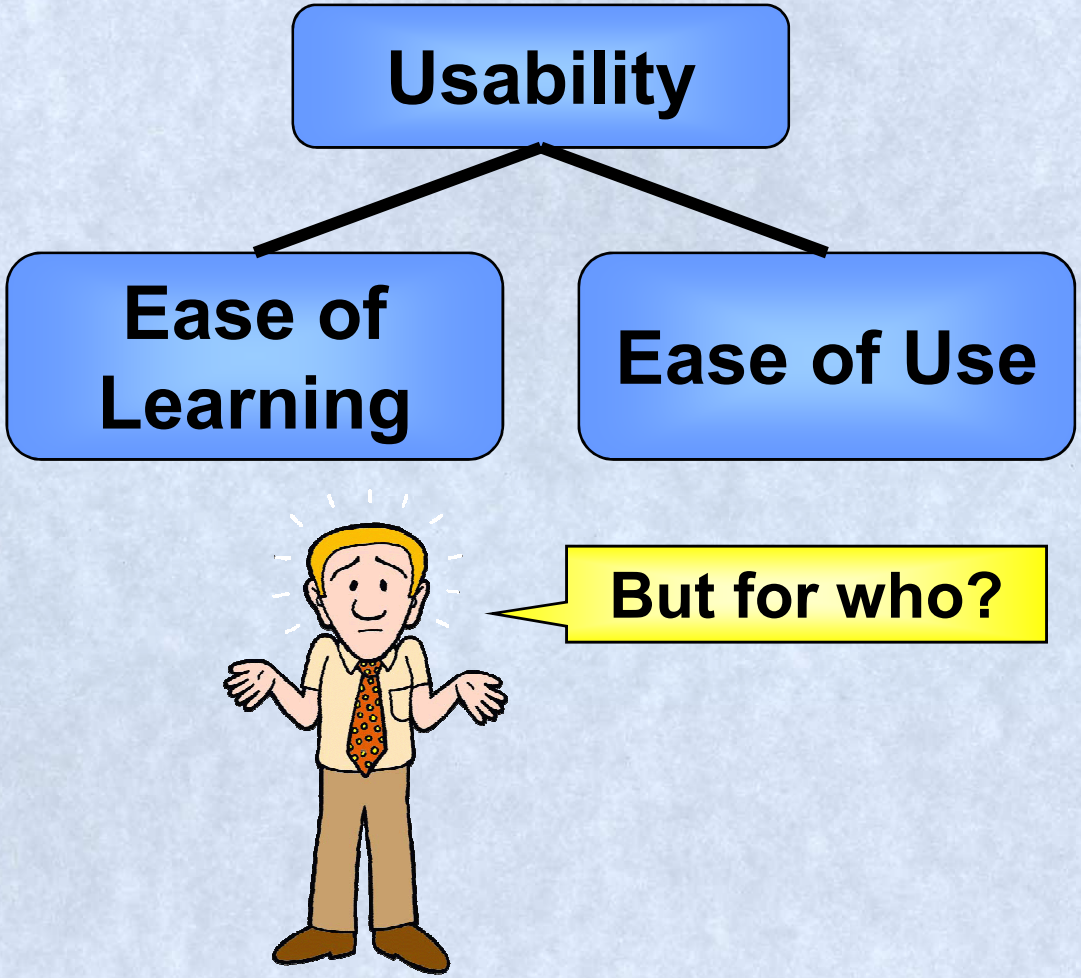
Measurable Quality Attributes

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]

[based on Gilb-02]

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]

[based on Gilb-02]

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

[based on Gilb-02]

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”

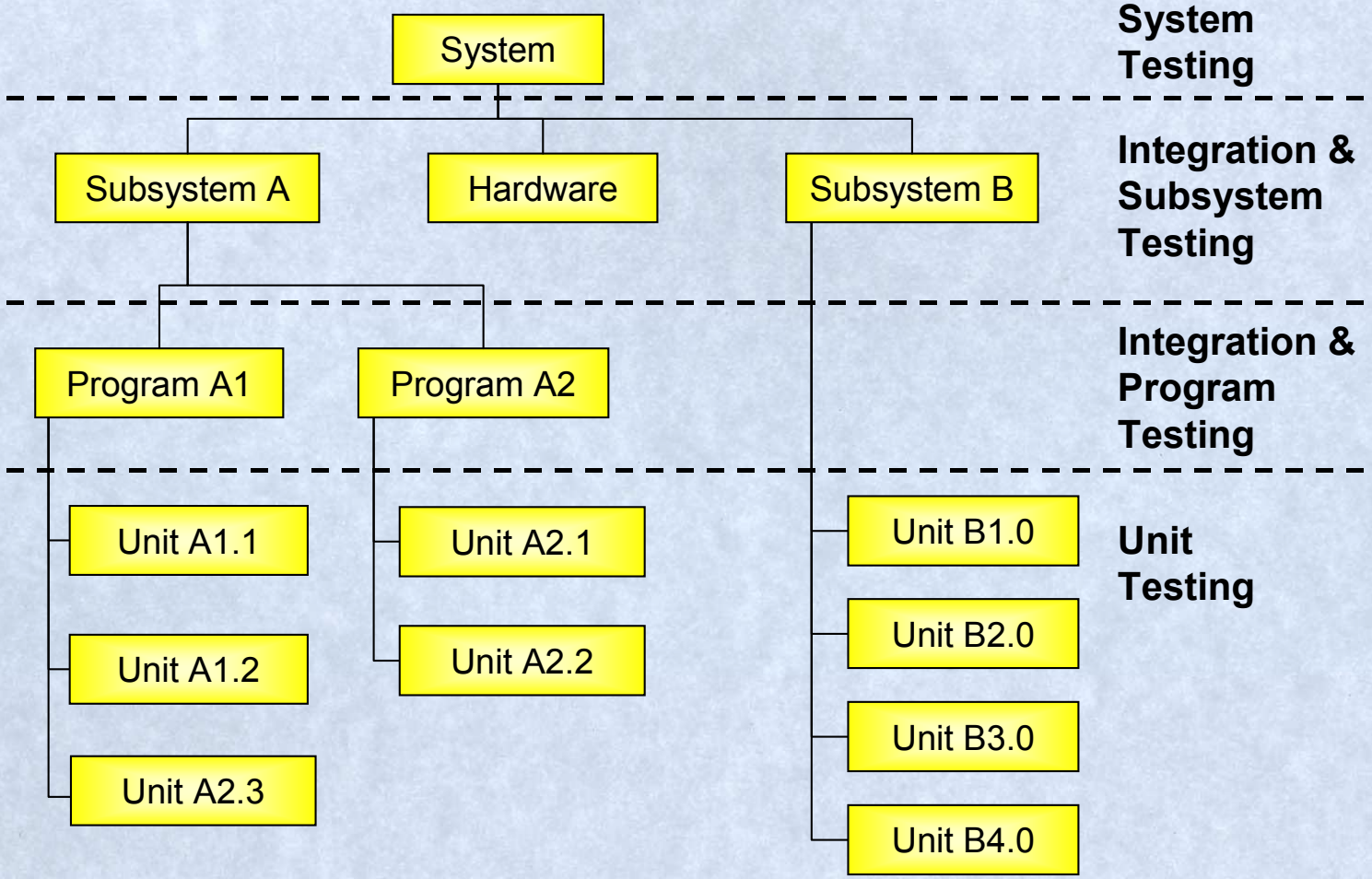
Tester Participation in Peer Reviews

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	

Trace Requirements to Test Cases

Requirement Source	Product Requirements	HLD Section #	LLD Section #	Code Unit	UTS Case #	STS Case #	User Manual
Business Rule #1	R00120 Credit Card Types	4.1 Parse Mag Strip	4.1.1 Read Card Type	Read_Card _Type.c Read_Card _Type.h	UT 4.1.032 UT 4.1.033 UT 4.1.038 UT 4.1.043	ST 120.020 ST 120.021 ST 120.022	Section 12
			4.1.2 Verify Card Type	Ver_Card _Type.c Ver_Card _Type.h Ver_Card _Types. dat	UT 4.2.012 UT 4.2.013 UT 4.2.016 UT 4.2.031 UT 4.2.045	ST 120.035 ST 120.036 ST 120.037 ST 120.037	Section 12
Use Case #132 step 6	R00230 Read Gas Flow	7.2.2 Gas Flow Meter Interface	7.2.2 Read Gas Flow Indicator	Read_Gas _Flow.c	UT 7.2.043 UT 7.2.044	ST 230.002 ST 230.003	Section 21.1.2
	R00231 Calculate Gas Price	7.3 Calculate Gas price	7.3 Calculate Gas price	Cal_Gas_ Price.c	UT 7.3.005 UT 7.3.006 UT 7.3.007	ST 231.001 ST 231.002 ST 231.003	Section 21.1.3

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

References

- [Brooks-95] Fredrick Brooks, *Mythical Man-Month: Essays on Software Engineering, Anniversary Edition (2nd Edition),* Addison-Wesley Professional, Boston, 1995.
- [Gilb-02] Tom Gilb, *Requirements: Practical Advanced Requirements Engineering* presentation, May 30, 2002, available at www.gilb.com.
- [Sommerville-97] Ian Sommerville and Pete Sawyer, *Requirements Engineering: A Good Practice Guide,* John Wiley & Sons, Chichecter, 1997, ISBN 0-471-97444-7
- [Wiegers-04] Karl E. Wiegers, *In Search of Excellent Requirements,* Process Impact, 2004.
- [Young-01] Ralph Young, *Effective Requirements Practices,* Addison-Wesley, Boston, 2001, ISBN 0-201-70912-0.

Contact Information

the WestfallTeam

Partnering for Software Excellence

Linda Westfall
3000 Custer Road
Suite 270, PMB 101
Plano, TX 75075-4499

phone: (972) 867-1172
fax: (972) 943-1484

lwestfall@westfallteam.com
www.westfallteam.com