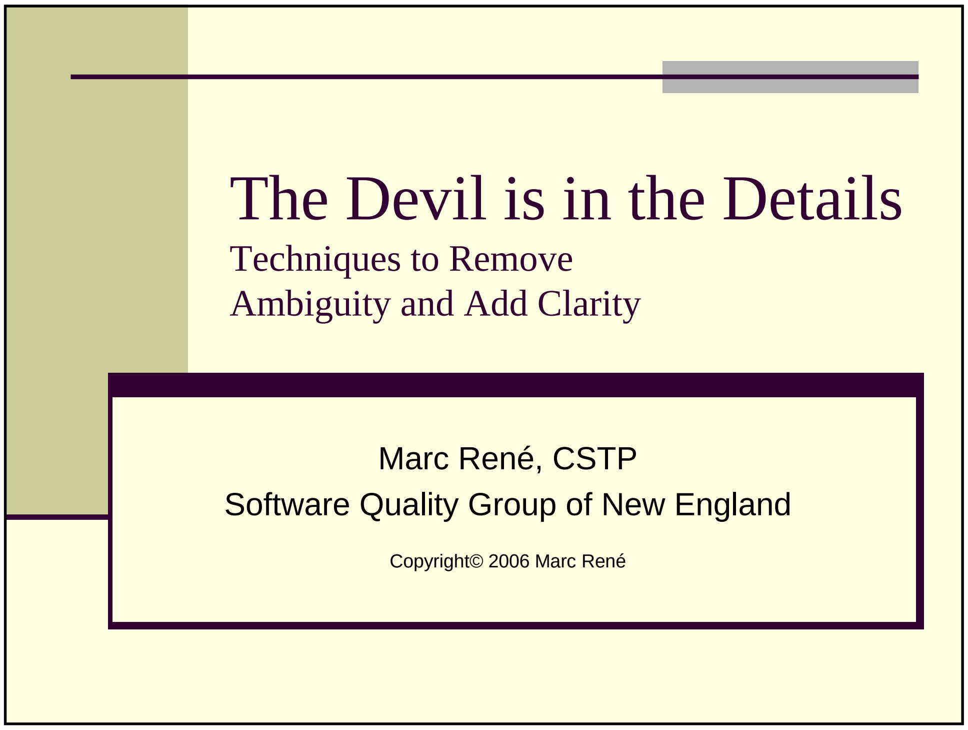
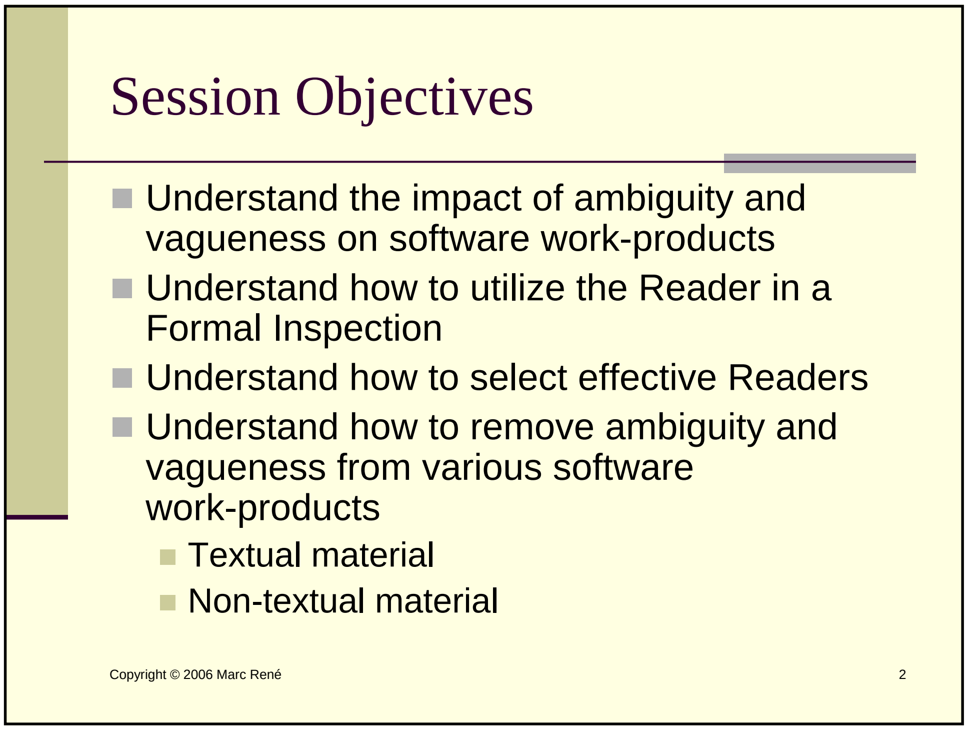


# The Devil is in the Details

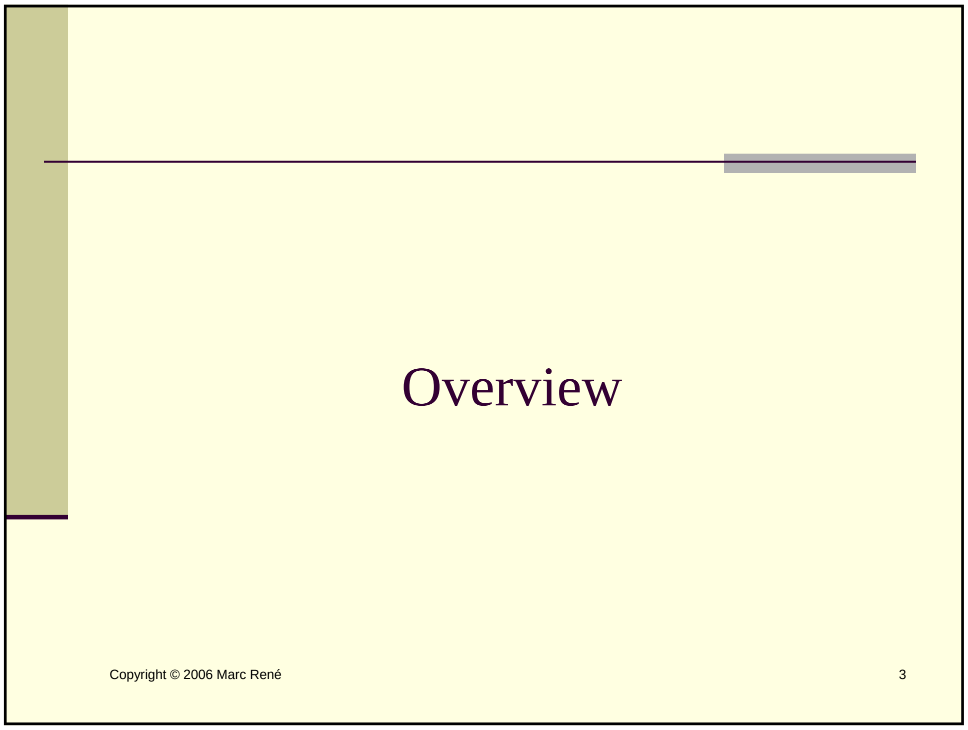
Techniques to Remove Ambiguity and Add Clarity

Marc René, CSTP
Software Quality Group of New England

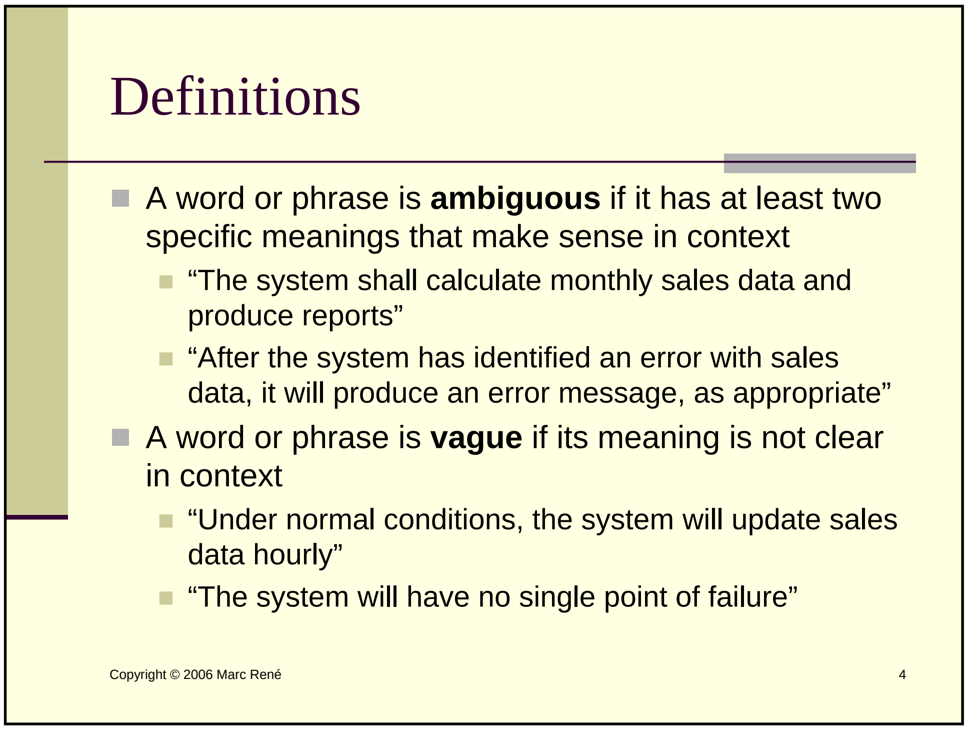


## Session Objectives

- Understand the impact of ambiguity and vagueness on software work-products
- Understand how to utilize the Reader in a Formal Inspection
- Understand how to select effective Readers
- Understand how to remove ambiguity and vagueness from various software work-products
  - Textual material
  - Non-textual material



## Overview



## Definitions

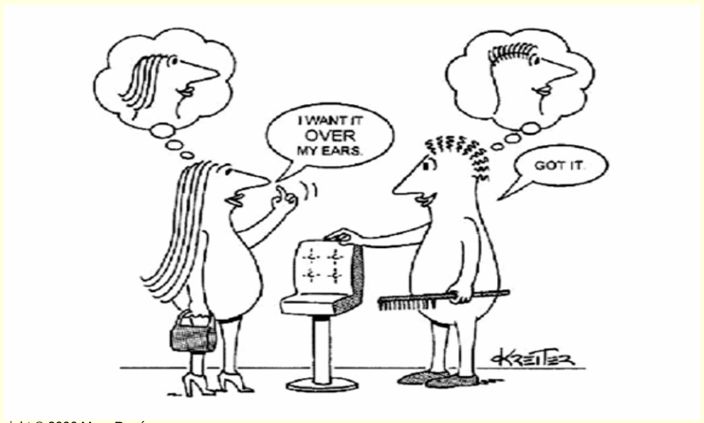
- A word or phrase is **ambiguous** if it has at least two specific meanings that make sense in context
  - “The system shall calculate monthly sales data and produce reports”
  - “After the system has identified an error with sales data, it will produce an error message, as appropriate”
- A word or phrase is **vague** if its meaning is not clear in context
  - “Under normal conditions, the system will update sales data hourly”
  - “The system will have no single point of failure”

## Problems with Wording

There was a law in Kansas that read:

When two trains approach each other at a crossing, they both shall come to a full stop and neither shall start up until the other has gone.

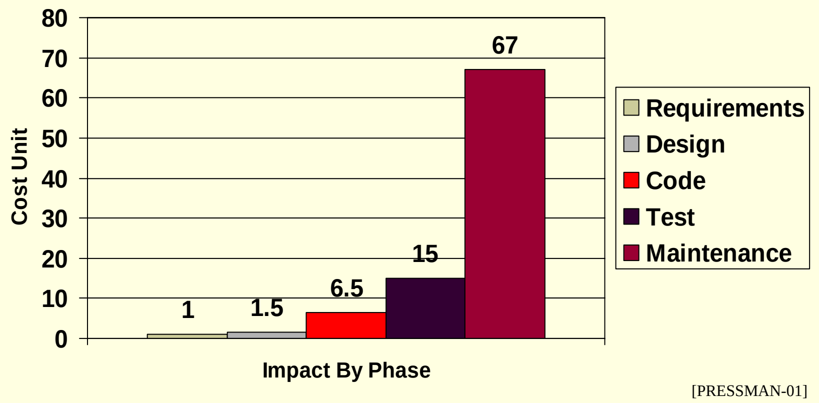
## Problems with Perspective



## Defects in the Software Life Cycle

| Phase | Defect Injection | Defect Prevention/Removal |
|---|---|---|
| **Requirements** | Incorrect, missing, or unclear requirements | Structured analysis, prototyping, test planning, and peer reviews |
| **Design** | Incorrect or unclear translation of requirements; Incorrect, missing, overly complex, or unclear design | Prototyping, test planning, and peer reviews |
| **Implementation** | Incorrect translation of design; Incorrect syntax or semantics; Complex implementation; Incorrect documentation; Bad unit test fixes | Defensive programming, peer reviews, static analysis, and unit testing |
| **Formal Test** | Incorrect or incomplete fixes | Testing |
| **Maintenance** | Incorrect or incomplete maintenance fixes | Customer use |

## Defects - Cost Impact of Change



[PRESSMAN-01]

## Ambiguity Example

A true story

The phrase “XYZ is an optional feature” appeared in a requirements document.

Designer
- “This is an optional requirement. I don’t have to implement it.”

Customer
- “XYZ is a feature that will be provided, and I can choose when I want to use it.”

Product Manager
- “This is a feature that will be available as a separate package, and the customer can pay an additional amount for it when they purchase the main product.”

[RENE-04]

## Ambiguity Example – Part 2

Still a true story

Performed a Formal Inspection – but the Reader read verbatim

What happened?

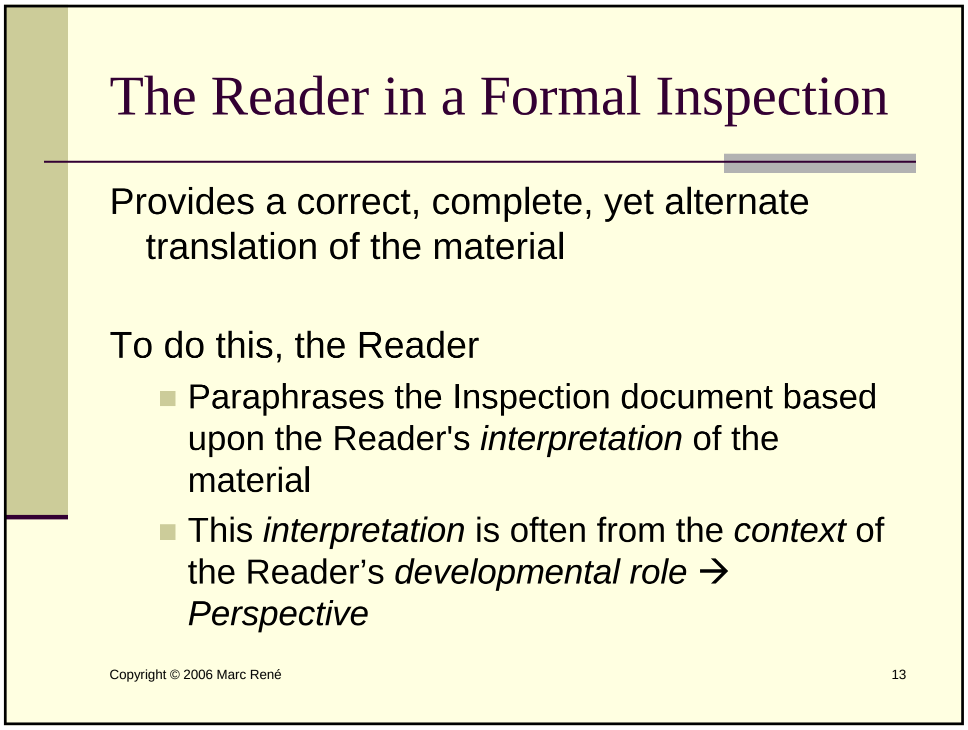
- Interpretations left unchallenged
- Each “knew” what the requirement meant, so no questions were raised
- Developer did not implement
- Missing feature caused significant problems
- Adding the feature → considerable rework

[RENE-04]

## Formal Inspections

## What is a Formal Inspection?

- An ‘organized review’ that is facilitated by a trained Moderator
- Systematic examination of a work-product(s) :
  - Verify the work-product satisfies specifications
  - Verify the work-product satisfies specified quality characteristics
  - Evaluate conformance to specified standards, guidelines, plans, and procedures
  - Detect and document defects
- Roles: Moderator, Reader, Author, Reviewers, Scribe

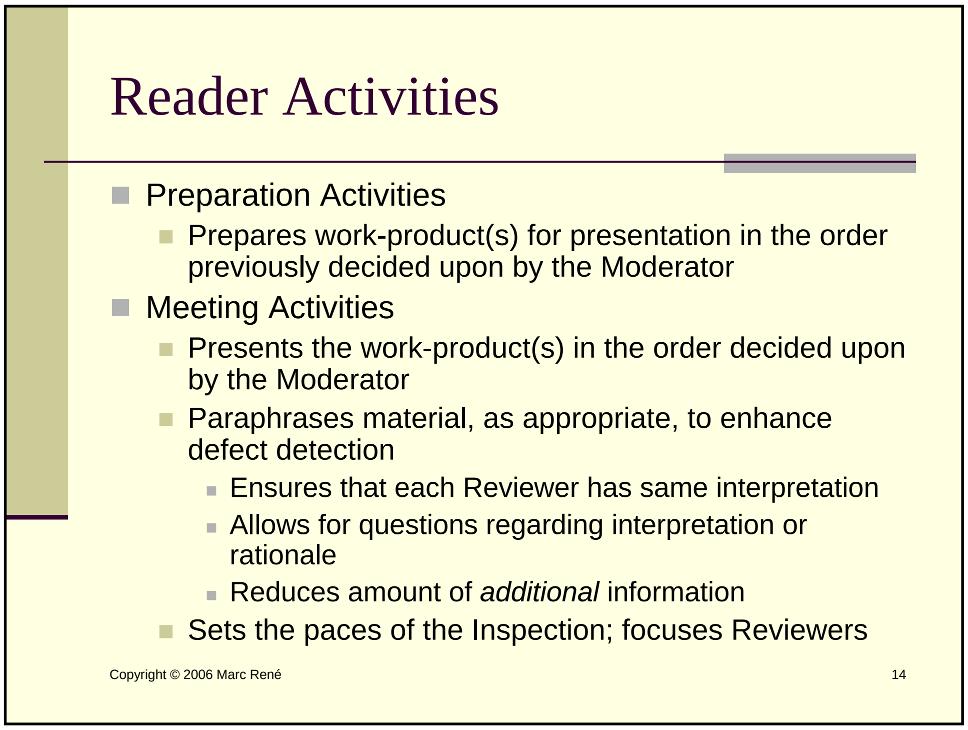


## The Reader in a Formal Inspection

Provides a correct, complete, yet alternate translation of the material

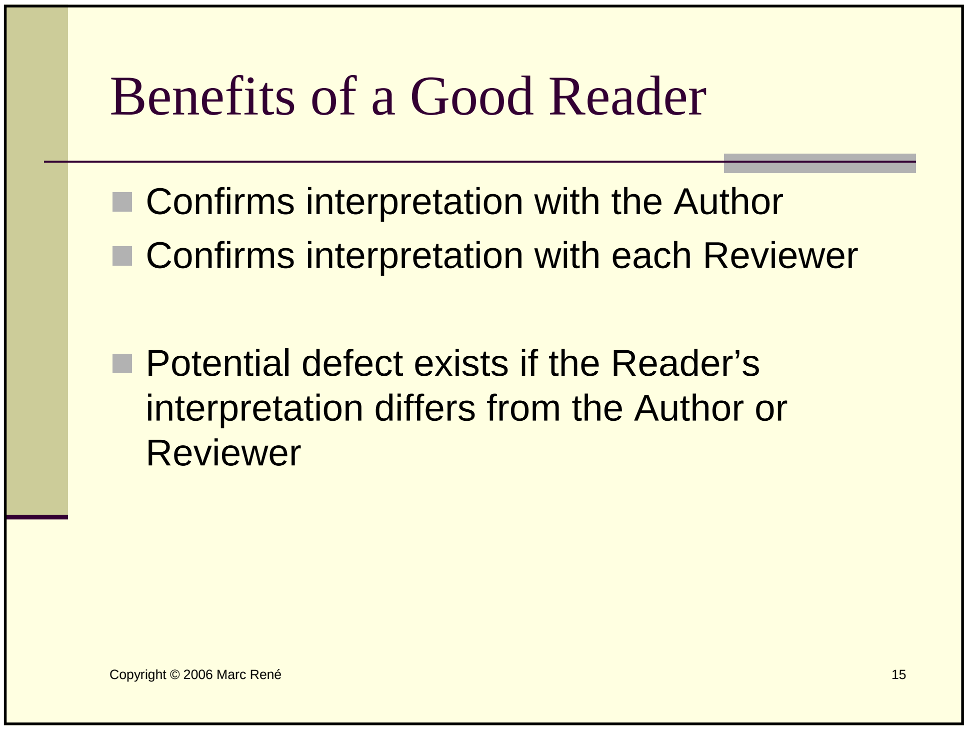
To do this, the Reader

- Paraphrases the Inspection document based upon the Reader's *interpretation* of the material
- This *interpretation* is often from the *context* of the Reader’s *developmental role* → *Perspective*



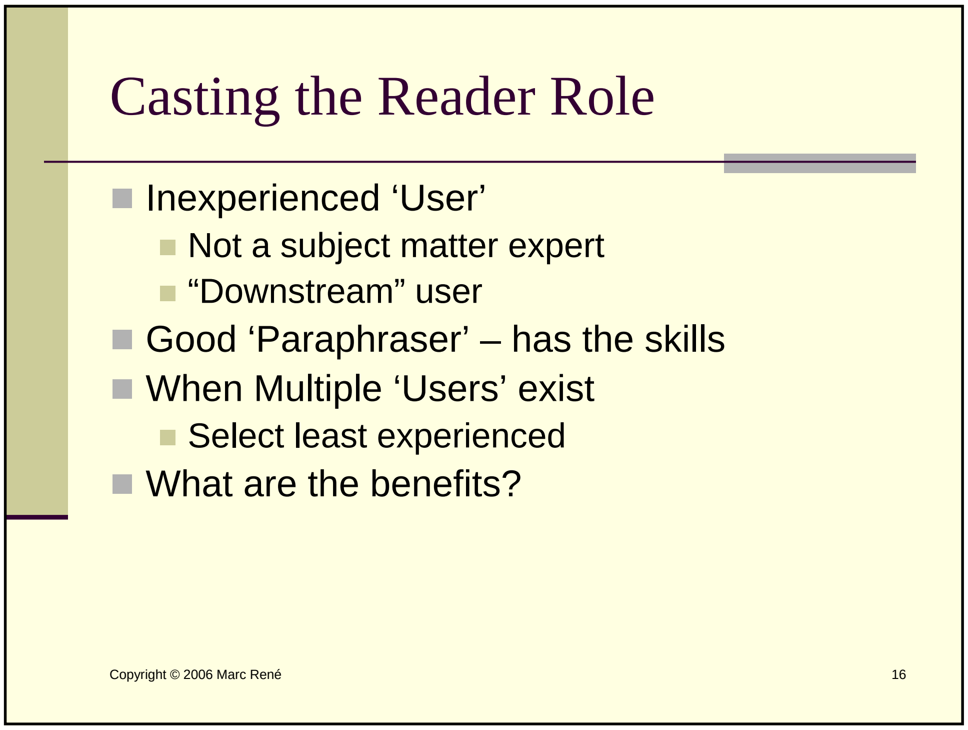
## Reader Activities

- Preparation Activities
  - Prepares work-product(s) for presentation in the order previously decided upon by the Moderator
- Meeting Activities
  - Presents the work-product(s) in the order decided upon by the Moderator
  - Paraphrases material, as appropriate, to enhance defect detection
    - Ensures that each Reviewer has same interpretation
    - Allows for questions regarding interpretation or rationale
    - Reduces amount of *additional* information
  - Sets the paces of the Inspection; focuses Reviewers



## Benefits of a Good Reader

- Confirms interpretation with the Author
- Confirms interpretation with each Reviewer
- Potential defect exists if the Reader's interpretation differs from the Author or Reviewer



## Casting the Reader Role

- Inexperienced 'User'
  - Not a subject matter expert
  - "Downstream" user
- Good 'Paraphraser' – has the skills
- When Multiple 'Users' exist
  - Select least experienced
- What are the benefits?

## Things to Avoid as a Reader

- Presenting the material word-for-word
- Moving from section to section asking if anyone has “any questions” without any effort to interpret the material
- Phrasing statements like, “I think the next section is obvious – any questions?”
- Adding details or providing elaboration not explicitly provided in the Inspection material

## Techniques

## Consider Perspective

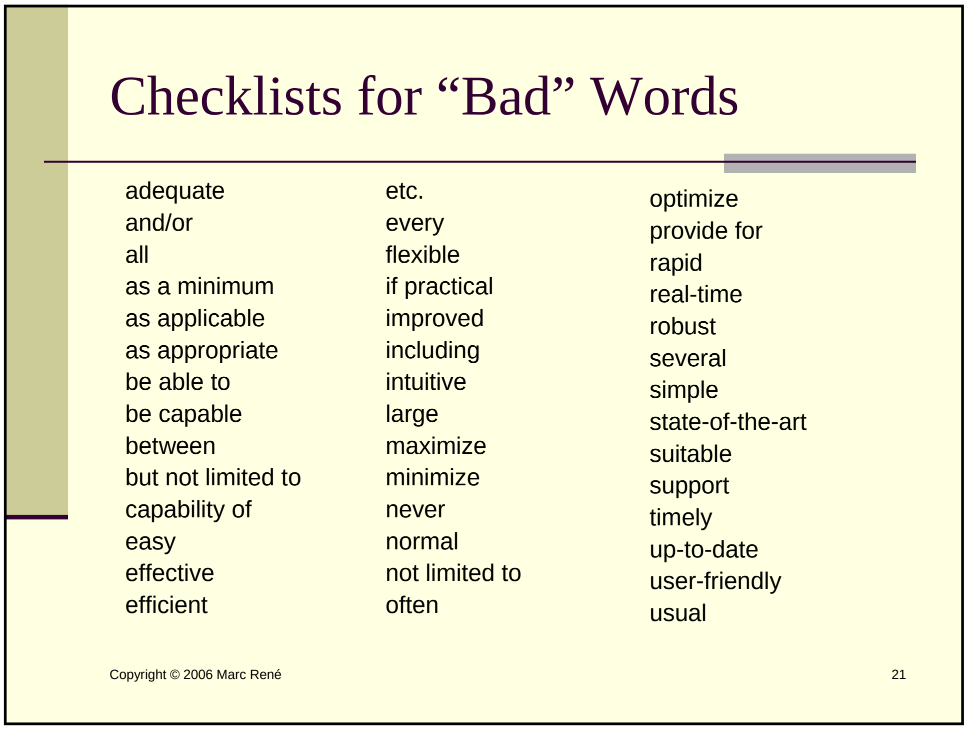
- Who will use this document?
  - Customer
  - Downstream users of the work-product
  - Marketing/Product Managers
  - Technical Services – Maintainers
- How will each user use this document?
- What information does each user need from this document?
- How will each user interpret this information?

[WENSEL-03]

## "Development Role" Perspective

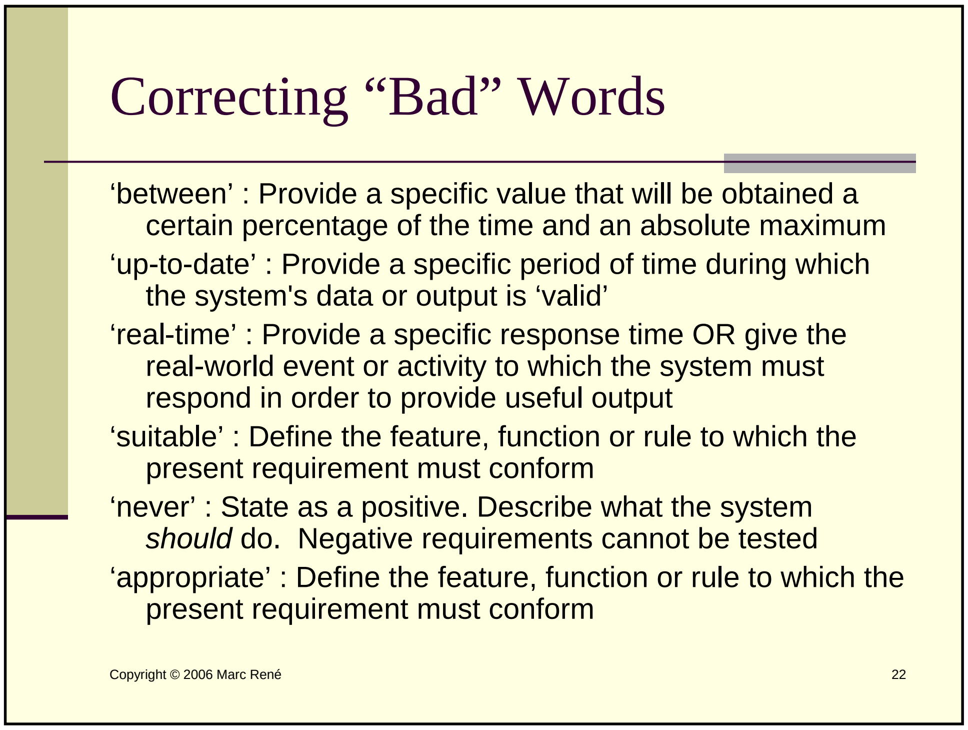
- How will the document be used?
- What information is required for it to be used?

| | |
|---|---|
| Plans | ➔ Project Team Member |
| Requirements | ➔ Designer |
| Design | ➔ Coder |
| Test Procedures | ➔ Testers |
| Code | ➔ Maintainers |
| User's Guide | ➔ Help Desk/Users |



## Checklists for “Bad” Words

adequate
and/or
all
as a minimum
as applicable
as appropriate
be able to
be capable
between
but not limited to
capability of
easy
effective
efficient
etc.
every
flexible
if practical
improved
including
intuitive
large
maximize
minimize
never
normal
not limited to
often
optimize
provide for
rapid
real-time
robust
several
simple
state-of-the-art
suitable
support
timely
up-to-date
user-friendly
usual



## Correcting “Bad” Words

‘between’ : Provide a specific value that will be obtained a certain percentage of the time and an absolute maximum

‘up-to-date’ : Provide a specific period of time during which the system's data or output is ‘valid’

‘real-time’ : Provide a specific response time OR give the real-world event or activity to which the system must respond in order to provide useful output

‘suitable’ : Define the feature, function or rule to which the present requirement must conform

‘never’ : State as a positive. Describe what the system *should* do. Negative requirements cannot be tested

‘appropriate’ : Define the feature, function or rule to which the present requirement must conform

## Develop ‘Test Cases’

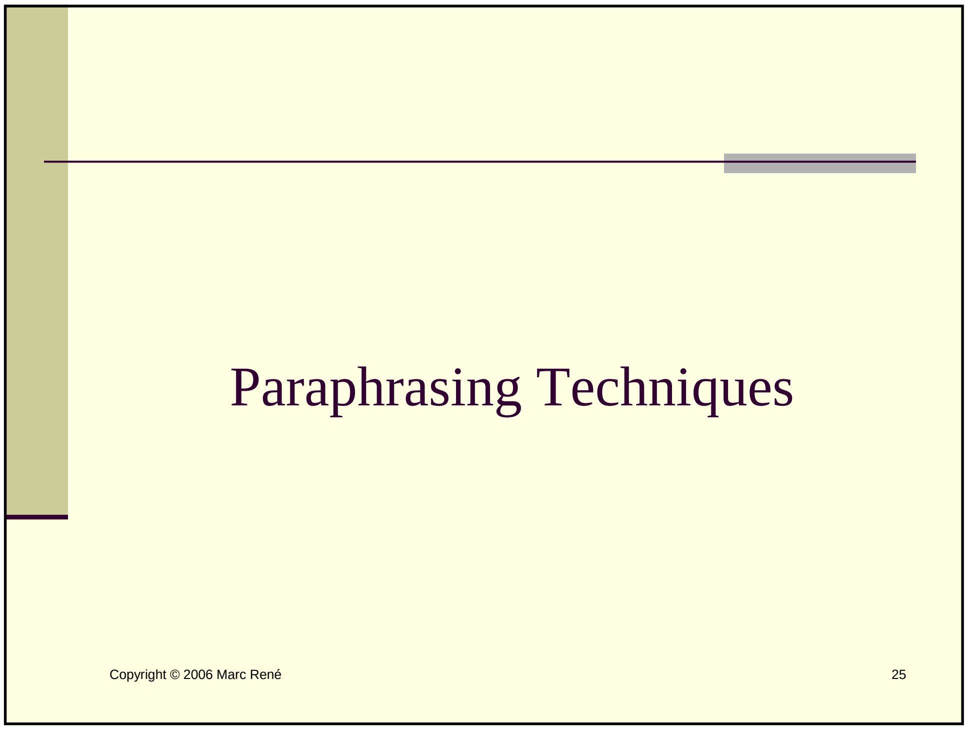
- *Conceptual* ‘test cases’ clarify how the software should behave under certain conditions
  - Use cases
  - Functional requirements
  - Design
  - Code
- Reveal ambiguities and missing information
- Leads to a document that supports comprehensive test case generation

[WIEGERS-PI]

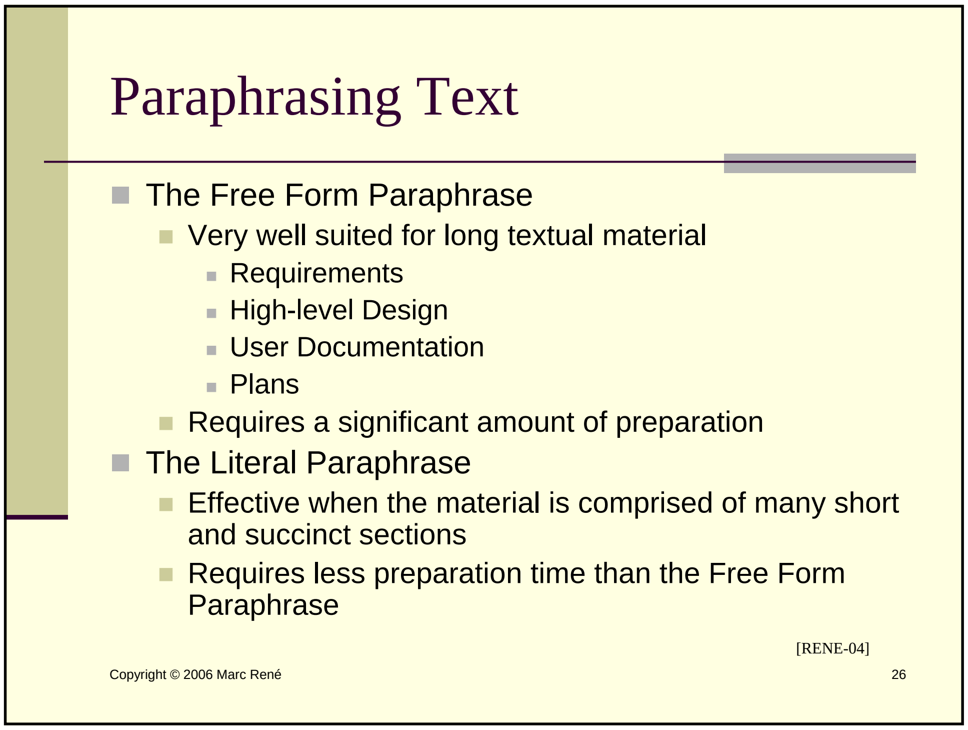
## Develop ‘Test Cases’ Example

Requirement: A valid ATM user must be able to withdraw up to $200 or the maximum amount in the account

1. Withdraw $200 with $1000 in account
   Pass
2. Withdraw $200 with $100 in account
   Fail?
3. Withdraw $200 with $1000 in account 3 times in one day
   Pass?
4. Withdraw $1000 with $1000 in account
   Pass?
5. Withdraw $20.15
   Pass?
6. Other ideas?



## Paraphrasing Techniques



## Paraphrasing Text

- The Free Form Paraphrase
  - Very well suited for long textual material
    - Requirements
    - High-level Design
    - User Documentation
    - Plans
  - Requires a significant amount of preparation
- The Literal Paraphrase
  - Effective when the material is comprised of many short and succinct sections
  - Requires less preparation time than the Free Form Paraphrase

[RENE-04]

## Free Form Method

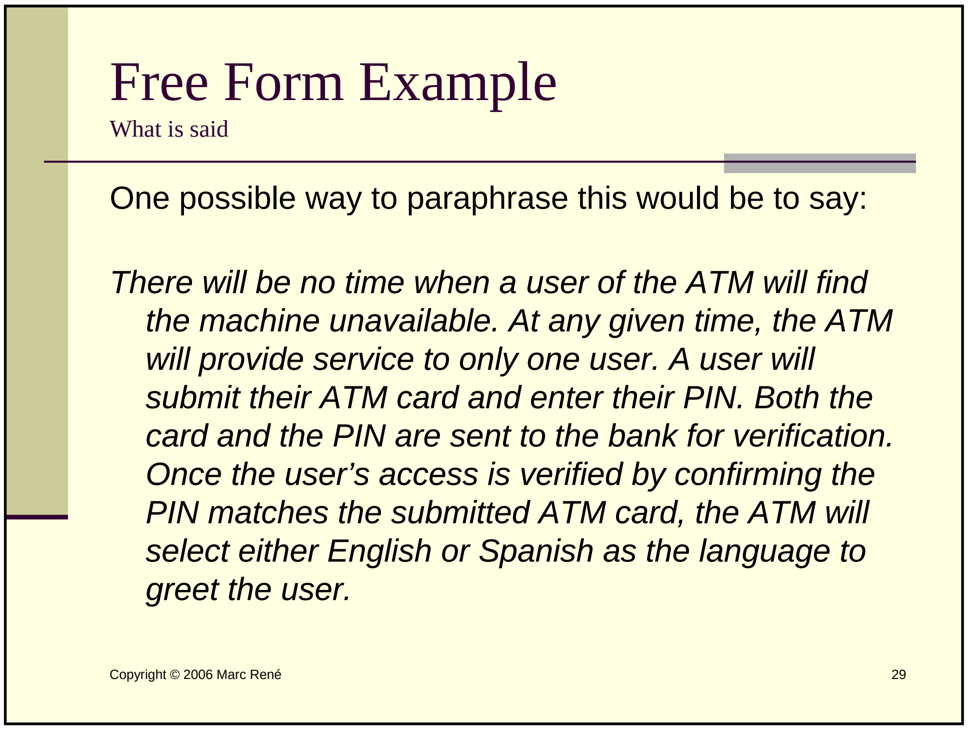
Taking the Inspection material one section at a time:

- Read the section until full meaning is understood
- Set the original aside
- Create notes describing the material in the context of its use in the next phase of development
  - Extract, distill, and compress essential content that is relevant to the use of the material
- Using the notes and referring back to the original material, verify the notes paraphrase the material accurately
  - Ensures all of the essential information from the section is captured
- Use these notes in the Inspection meeting

## Free Form Example

What is written

The ATM will be available 24 hours a day, 7 days a week, 365 days a year. The ATM will service one customer at a time. The customer will insert an ATM card and then enter a personal identification number (PIN), which are sent to the bank for verification. After verifying the ATM card/PIN combination, the user is greeted in English or Spanish.

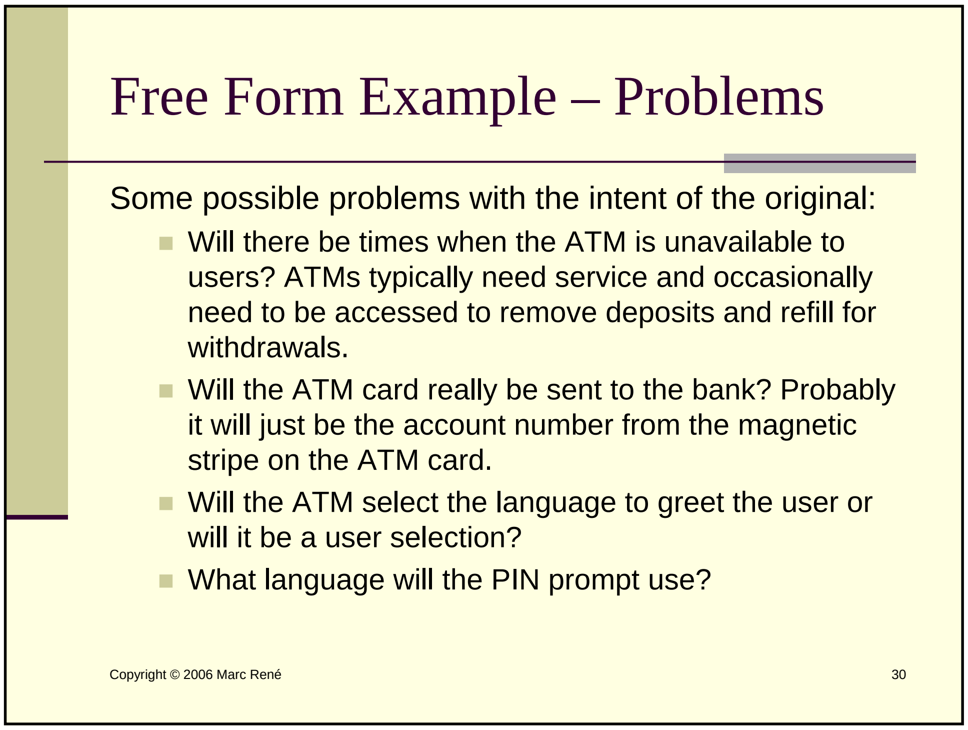


## Free Form Example

What is said

One possible way to paraphrase this would be to say:

*There will be no time when a user of the ATM will find the machine unavailable. At any given time, the ATM will provide service to only one user. A user will submit their ATM card and enter their PIN. Both the card and the PIN are sent to the bank for verification. Once the user's access is verified by confirming the PIN matches the submitted ATM card, the ATM will select either English or Spanish as the language to greet the user.*



## Free Form Example – Problems

Some possible problems with the intent of the original:

- Will there be times when the ATM is unavailable to users? ATMs typically need service and occasionally need to be accessed to remove deposits and refill for withdrawals.
- Will the ATM card really be sent to the bank? Probably it will just be the account number from the magnetic stripe on the ATM card.
- Will the ATM select the language to greet the user or will it be a user selection?
- What language will the PIN prompt use?

## Literal Method

- Taking the Inspection material one section at a time:
  - Substitute words and phases with synonyms based upon personal  experience
  - Break long sentences into smaller ones
  - Elaborate on potentially ambiguous words or phrases
  - Whenever it makes sense, change the order of ideas.
    - This may provide a more logical or concise comparison of alternative situations
      Note: This step may not always be possible or necessary

## Literal Example

The vending machine will not require the user to deposit exact change.

*At no time will the vending machine require exact change.*

The vending machine will only accept U.S. coins.

*The following U.S. Coins will be accepted by the vending machine (pennies, nickels, dimes, quarters,half-dollars and the three types of dollars).*

## Paraphrasing Non-Textual Material

- Tables, figures, screens, diagrams, and graphs
  - Assumption: During the preparation activity the Inspectors will have already determined if there are inconsistencies between the tables, figures and graphs and the supporting text
- Source Code

## Tables, Figures, Screens, …

- For figures, diagrams, or pictures
  - Follow arrows describing each action or relationship
  - If time-based, follow the most common case
- For tables
  - Select either the horizontal or the vertical axis of the table and paraphrase the contents of the other
- For screen layouts or some other graphical image
  - Summarize the content and context of usage
  - Describe typical usage patterns
  - Added advantage of identifying usability issues
    - Reviews the intended use of the screen

[RENE-04]

## Source Code – Rationale

- Paraphrasing extends to providing the context or rationale for individual code statements
- To present the material in a more compact form
  - Step back from the actual code and present an overview of the source code or the design the source code represents
  - Describe the reason for the particular section

[RENE-04]

## Source Code - Preparation

- Need to understand intent of the source code
  - Distill the design
- Primary input is the analysis of the source code and the associated comments
  - Note: Do not rely solely on the information contained in comments as comments may not match the source code (a possible defect)
- Research all interfaces
- For complex or embedded logic paths
  - Use flow graphs
  - State transition diagrams
  - Other pictorial representations

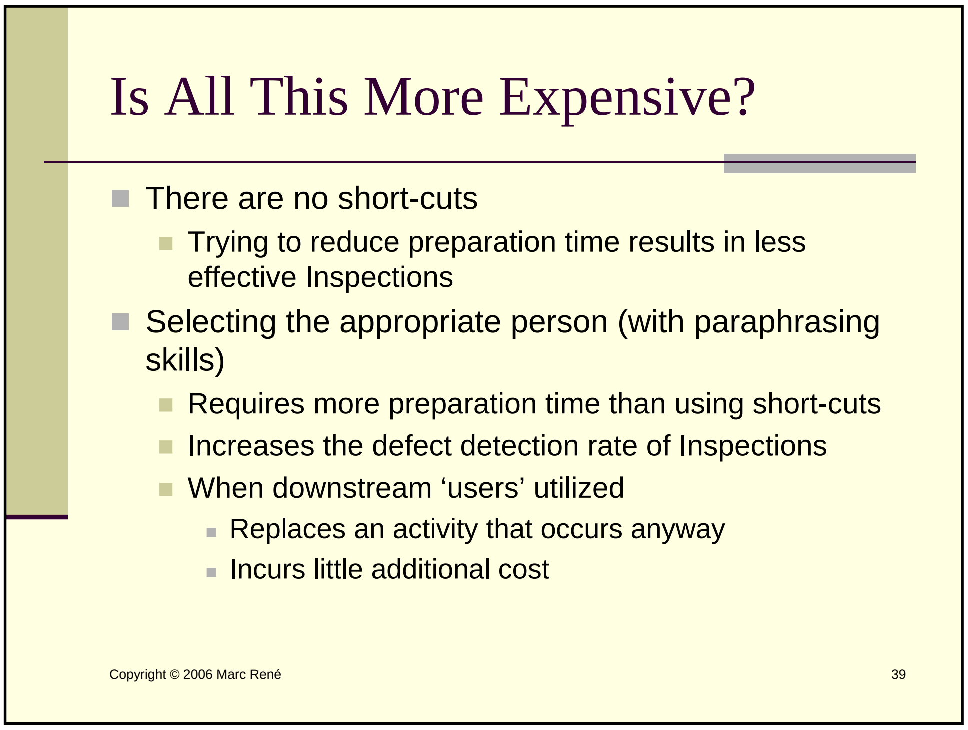
[RENE-04]

## Source Code - Presentation

- Order may not be the same order that it is printed
  - Start with the area that is the primary entry point
  - Move to the modules in the execution order (depth-first)
- Paraphrase each module completely before moving on to the next module
  - Ensures that each module is viewed as a complete entity
- When describing a module
  - Describe the input parameters
  - Output parameters or return value of the function
  - Paraphrase the body of the module

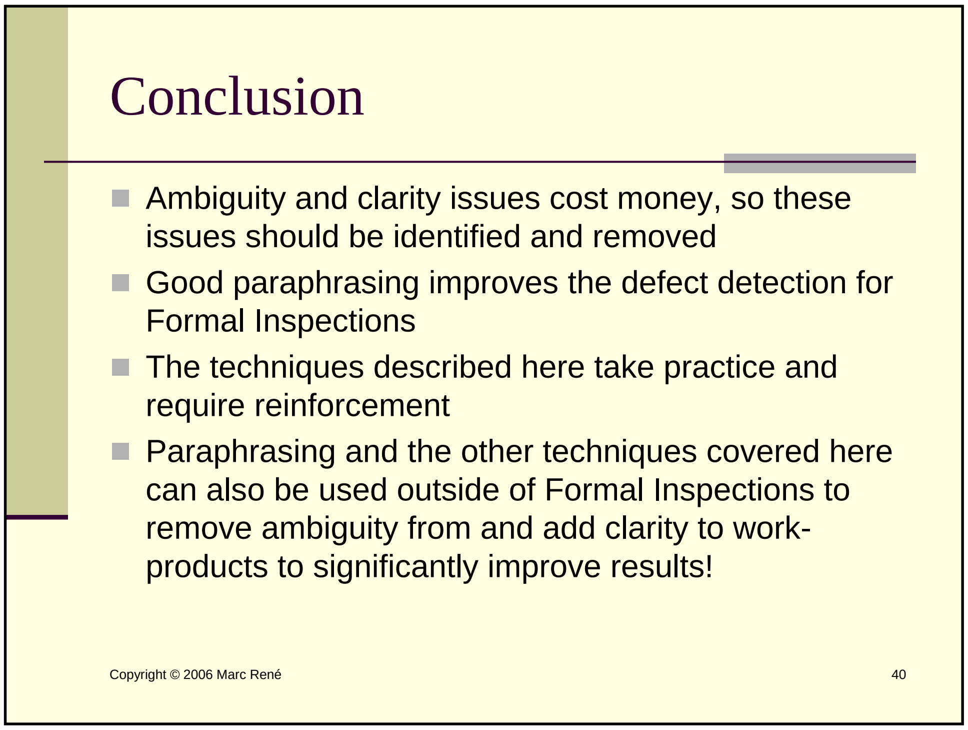
[RENE-04]

## Wrap-Up



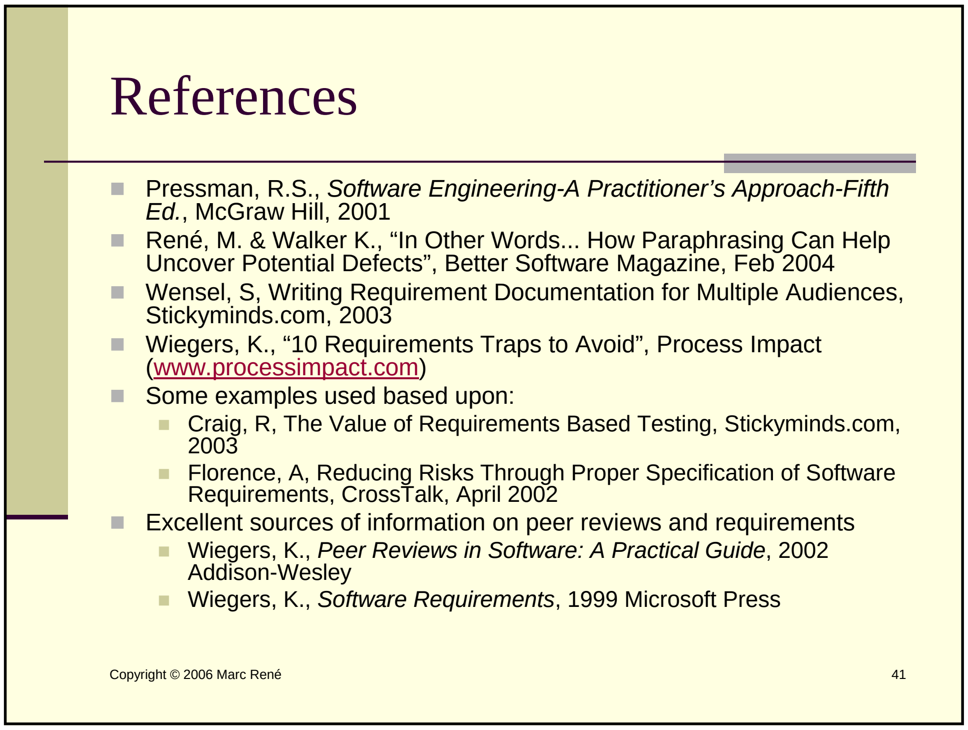
## Is All This More Expensive?

- There are no short-cuts
  - Trying to reduce preparation time results in less effective Inspections
- Selecting the appropriate person (with paraphrasing skills)
  - Requires more preparation time than using short-cuts
  - Increases the defect detection rate of Inspections
  - When downstream 'users' utilized
    - Replaces an activity that occurs anyway
    - Incurs little additional cost



## Conclusion

- Ambiguity and clarity issues cost money, so these issues should be identified and removed
- Good paraphrasing improves the defect detection for Formal Inspections
- The techniques described here take practice and require reinforcement
- Paraphrasing and the other techniques covered here can also be used outside of Formal Inspections to remove ambiguity from and add clarity to work-products to significantly improve results!



## References

- Pressman, R.S., *Software Engineering-A Practitioner's Approach-Fifth Ed.*, McGraw Hill, 2001
- René, M. & Walker K., "In Other Words... How Paraphrasing Can Help Uncover Potential Defects", Better Software Magazine, Feb 2004
- Wensel, S, Writing Requirement Documentation for Multiple Audiences, Stickyminds.com, 2003
- Wiegers, K., "10 Requirements Traps to Avoid", Process Impact (www.processimpact.com)
- Some examples used based upon:
  - Craig, R, The Value of Requirements Based Testing, Stickyminds.com, 2003
  - Florence, A, Reducing Risks Through Proper Specification of Software Requirements, CrossTalk, April 2002
- Excellent sources of information on peer reviews and requirements
  - Wiegers, K., *Peer Reviews in Software: A Practical Guide*, 2002 Addison-Wesley
  - Wiegers, K., *Software Requirements*, 1999 Microsoft Press