Carnegie Mellon
Software Engineering Institute

Why Isn't Someone Coding Yet (WISCY)?

Avoiding Ineffective Requirements

Agenda

- Requirements and Their Impacts
- Basic Requirements Definitions
- Comparison to Capability Maturity Model Integration® (CMMI®) Designations
- Requirements Development and Management

Didn't We Solve the Requirements Problem?[1]

- Sample of approximately 428 CMM-Based Assessments for Internal Process Improvement (CBA-IPI)
- Analyzed data from 1997 through August 2001
- Of the assessments conducted, only 33 percent fully satisfied the Requirements Management KPA

[Crosstalk, April 2002]

Didn't We Solve the Requirements Problem?[2]

"[Disciplines for **performance-based contracting** to be successful] start with requirements definitions, and that takes a skill set.. . . . It's a very difficult process to get a good set of requirements. There are not a ton of folks who are really good at that and you have to apply that very early in the process. **That is the first discipline.**"

--Ed Meagher, Acting CIO, Veteran's Administration

[Government Computer News, 2003]

Requirements and Their Impact

As Requirements Go, So Goes the Project

What is a Requirement?

- **Standard Definition**
  - *Something that the product must do or a quality that the product must have*
- **More Ways to Characterize**
  - *Something you discover BEFORE YOU START TO BUILD YOUR PRODUCT*

    --Robertson and Robertson (1999)
  - *Agreement reached between the customer and the developers on what the system will do*

Requirements: A Project Foundation[1]

❑ Quality Foundation

➲ *The greatest control on software quality can be exercised during requirements phases.*

[Stevens, 1999]

➲ *"Quality is conformance to requirements"*

[Philip Crosby, 2000]

➲ *"Quality is conformance to requirements. Everything else is bull...."*

[Forsha, 1992]

Requirements: A Project Foundation^2

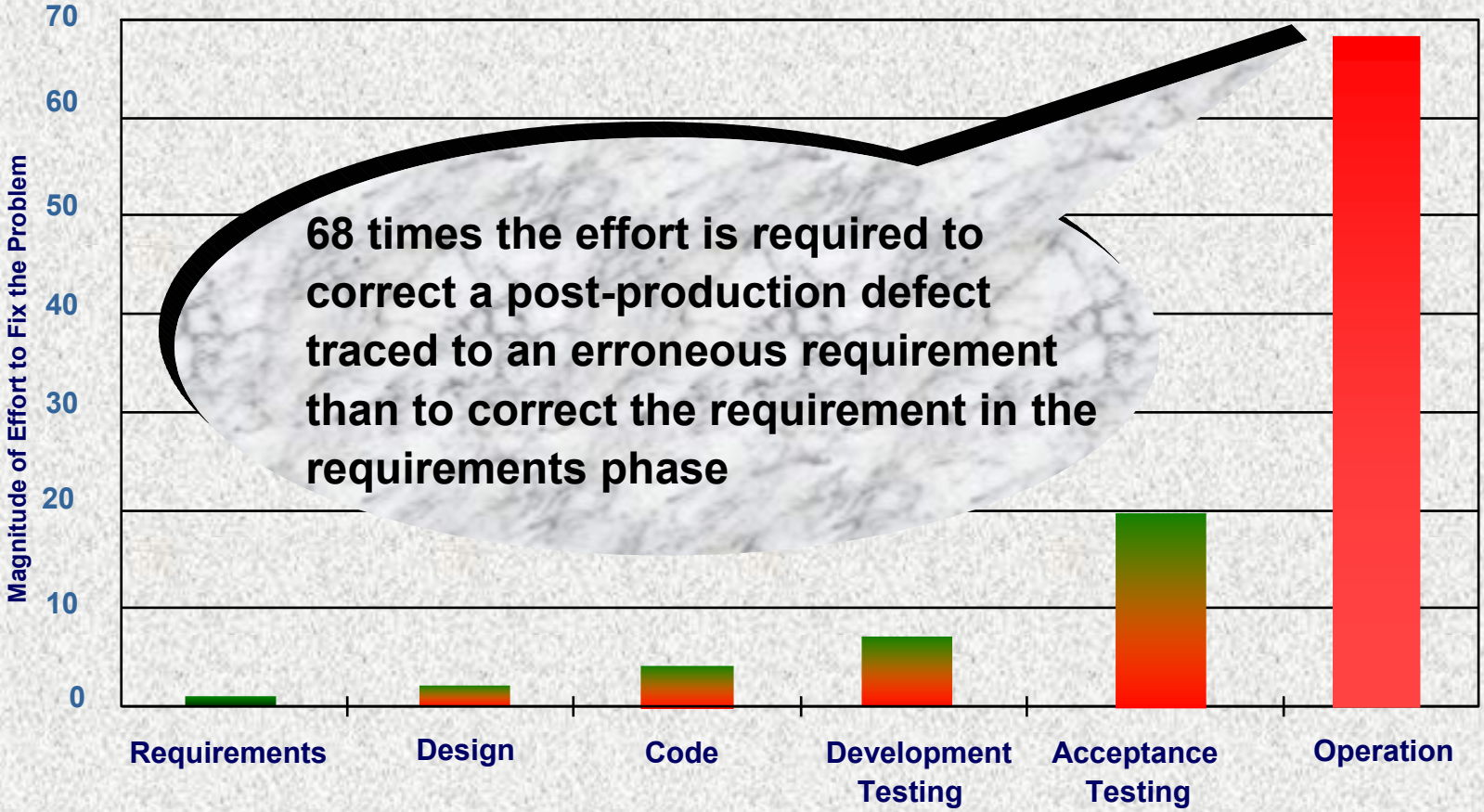
- **Planning Foundation**
  - *Clear and concise communication to all the team members*
  - *Alive and active throughout the lifecycle*
  - *Solution must reflect requirements*
- **ROI Foundation**
  - *BASIS FOR EFFORT ESTIMATES and thus cost and profit*

Size of the Problem

- 40 – 60% of errors in systems have been traced back to the requirements and analysis phase
- 70 – 85% of total revisions can be attributed to requirements errors

[Leffingwell, 1997]

Doing It Over . . .



[Boehm, 1981]

Carnegie Mellon
Software Engineering Institute

Requirements Development

Traditional Requirements Categories

- Business
- User
- Functional
- Non-functional

Requirement Type - Business

Meaning:

- What the organization hopes to achieve
- The business benefits that the product will offer

Eliciting the Business Requirements:

- How will this project (product) improve the business or organization?
- What will you be able to do that you cannot do now?

Requirement Type - User

Meaning

- What the user requires to complete tasks
- Business rules, data representation requirements, logical models, and acceptance criteria that user will employ

Eliciting the User Requirement

- Tasks that need to be accomplished?
- Required business rules?
- Deciding if the new system/product is working?

Requirement Type - Functional

Meaning

- What software system should do
- What it does to have effect on outside domain

Documenting the Functional Requirement

Functional Requirement FR#
Priority: *(Select High if must have, Medium if Important but not Critical, Low if Nice to Have)*
Description:
Related User Requirements UR#
Input information:
Output information:

Requirement Type – Non-Functional

Meaning

- Standards
- Regulations
- Constraints
- Interfaces
- Quality attributes that affect how the system must perform

Examples

- Enterprise Standards
- Government Regulations
- Platform
- Legacy Interfaces
- Usability
- Performance
- Scalability
- Security
- Flexibility
- Portability

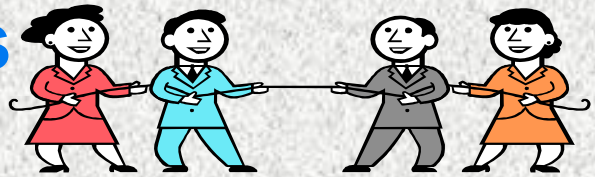
Successful Requirements Development

- Place high emphasis on requirements
  - *About 15% of the project life should be spent on requirements development activities before any final deliverable is built.*

  [Rubin, 1999]
- Use a variety of methods for obtaining requirements
  - *Unstructured interviews –no particular format*
  - *Structured interviews – specific questions and format*
  - *Observation – view and record user actions*
  - *Brainstorming – facilitated or non-facilitated group elicitation*
- Devise a consistent method for describing requirements

Elements of a Good Requirement

- Necessary
- Verifiable
- Feasible
- Clear and concise
- Complete
- Consistent
- Traceable
- No implementation bias



[Kar and Bailey, 1996]

Structure of a Requirement

- Keep sentences and paragraphs short
- Use active voice
- Use complete sentences with proper grammar, spelling and punctuation
- Use consistent wording
- Reduce ambiguity by avoiding vague and subjective terms
- Avoid comparative words and ambiguous language; quantify statement

Carnegie Mellon
Software Engineering Institute

Comparison to Capability Maturity Model Integration® (CMMI®) Designations

Model Overview[1]

- **Capability Maturity Model (CMM®)**
  - *Philosophy that quality processes enable quality products*
  - *Essential elements of effective processes for one or more bodies of knowledge*
  - *First CMM released in 1991 and targeted Software Engineering (SW-CMM)*
  - *Other discipline-specific CMMs created, e.g:*
    - **Systems Engineering**
    - **SA-CMM**
    - **Others . . .**

Model Overview[2]

- **Issues with multiple models**
  - *Hampered ability to focus improvements where multiple disciplines present*
  - *More costly in terms of training, appraisals, and improvement activities when applied within an organization*
- **Solution**
  - *An integration of four sources – SW, SE, IPPD, SS*
  - *Addresses multiple disciplines*
  - *Integrates training, appraisal support, and improvement activities*

Model Overview[3]

- **Capability Maturity Model Integration® (CMMI®)**
  - *Cohesive set of integrated models for organizations already using other CMMs, as well as for those new to the CMM concept*
  - *Consistent and compatible with the International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) 15504 Technical Report for Software Process Assessment*
  - *More information at http://www.sei.cmu.edu/cmmi/*

CMMI® and Requirements[1]

- Two principal process areas (PA)
  - *Maturity Level 2 – Requirements Management*
  - *New* *Maturity Level 3 - Requirements Development*
- Purpose of Requirements Development PA - produce and analyze:
  - *Customer requirements*
  - *Product requirements*
  - *Product-component requirements*
  - *Derived requirements*

CMMI® and Requirements[2]

- **Customer requirements**
  - *An understanding of what will satisfy stakeholders*
  - *Transformed stakeholder needs, expectations, constraints, and interfaces*
  - *May be stated in technical or non-technical terms*
  - *May also provide specific design requirements*

CMMI® and Requirements[3]

- Product requirements – a work product delivered to the customer
  - *More detailed and precise sets of requirements*
  - *Expressed in technical terms or parameters*
    - Functionality, including actions, sequence, inputs, and outputs
    - Qualities it must possess
    - Constraints that the system and its development must satisfy

[CMMI, Software Engineering Institute, 2003]

CMMI® and Requirements[4]

- **Product-component requirements – lower level components of the product**
  - ***Example - a car engine and a piston are product components of a car (the product)***
  - ***Complete specification, including fit, form, function, performance, and any other requirement***
  - ***Sufficiently technical for use in the design of the product component***

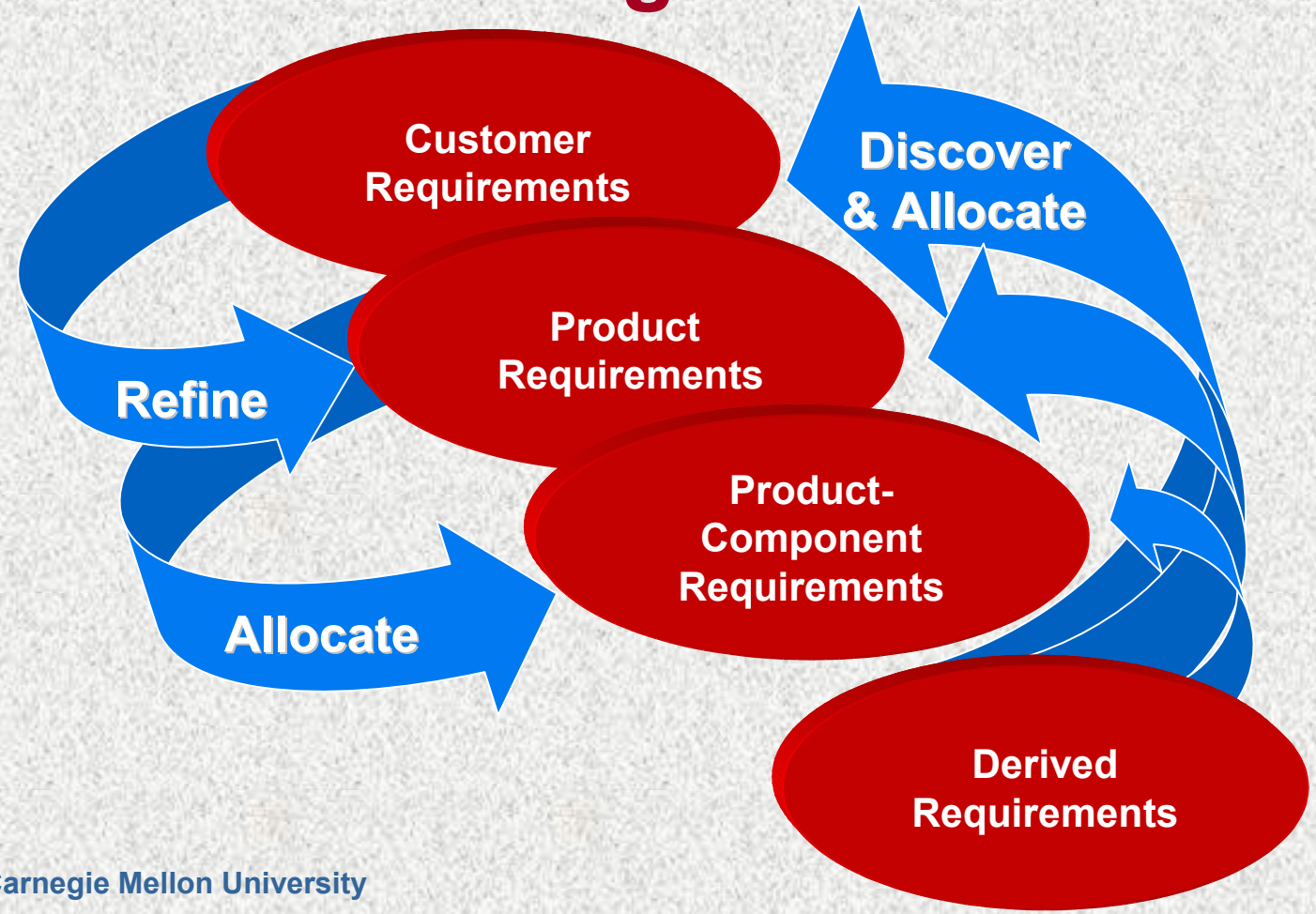
CMMI® and Requirements[5]

- Derived requirements – discovered and/or implied
  - *Not explicitly stated but inferred from:*
    - Customer requirements
    - Contextual requirements (e.g., applicable standards, laws, policies, common practices, and management decisions)
    - Contractual commitments such as data rights for delivered commercial off-the-shelf (COTS), and non-developmental items (NDIs); terms and condition, delivery dates, and milestones with exit criteria

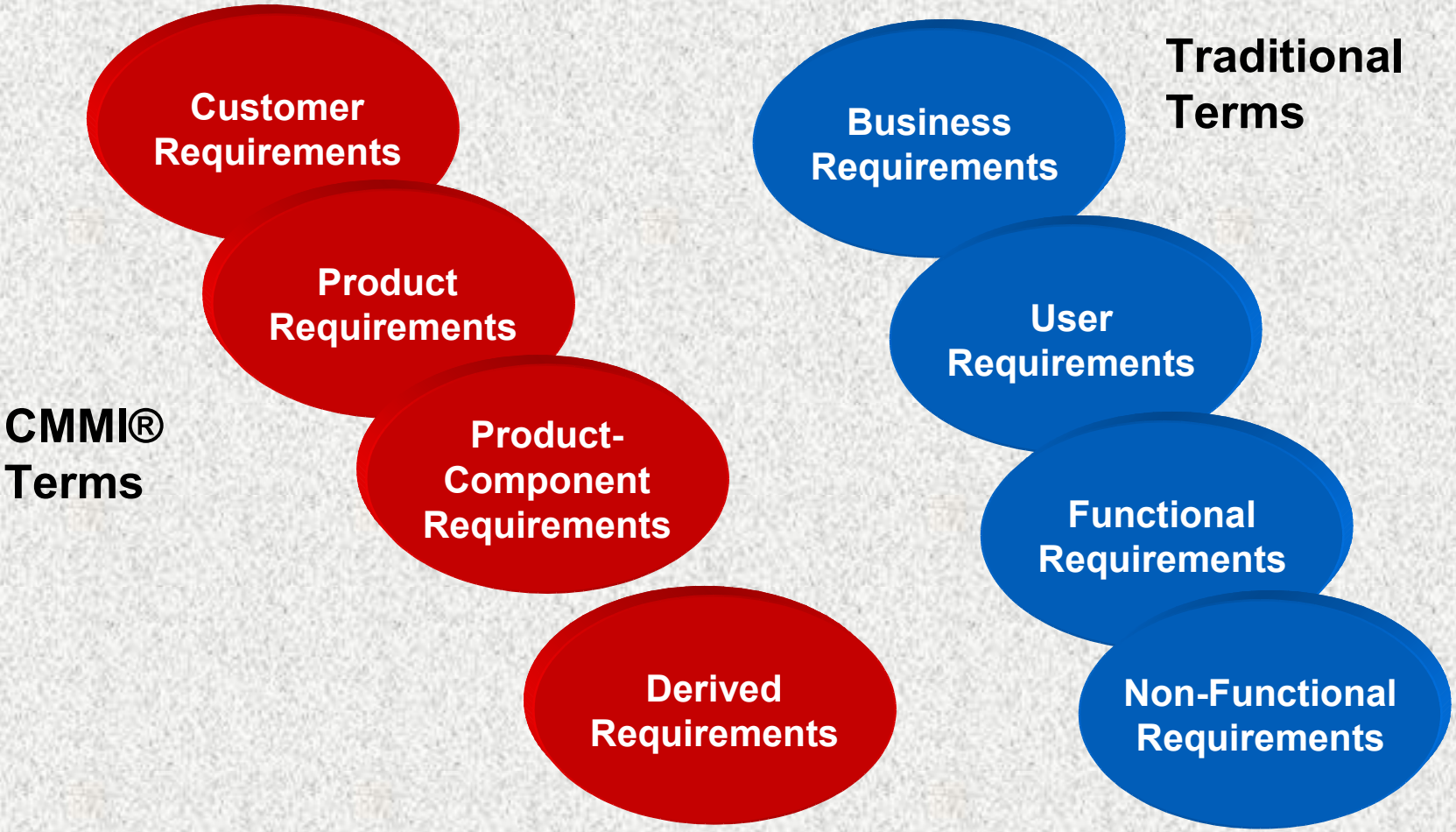
CMMI® and Requirements[6]

- Derived requirements (cont.)
  - *Factors arise as part of:*
    - Selected architecture
    - Design decisions
    - Developer's unique business considerations
  - *May also address the cost and performance of other life-cycle phases and other non-technical requirements*
    - Training requirements
    - Site requirements
    - Deployment schedules

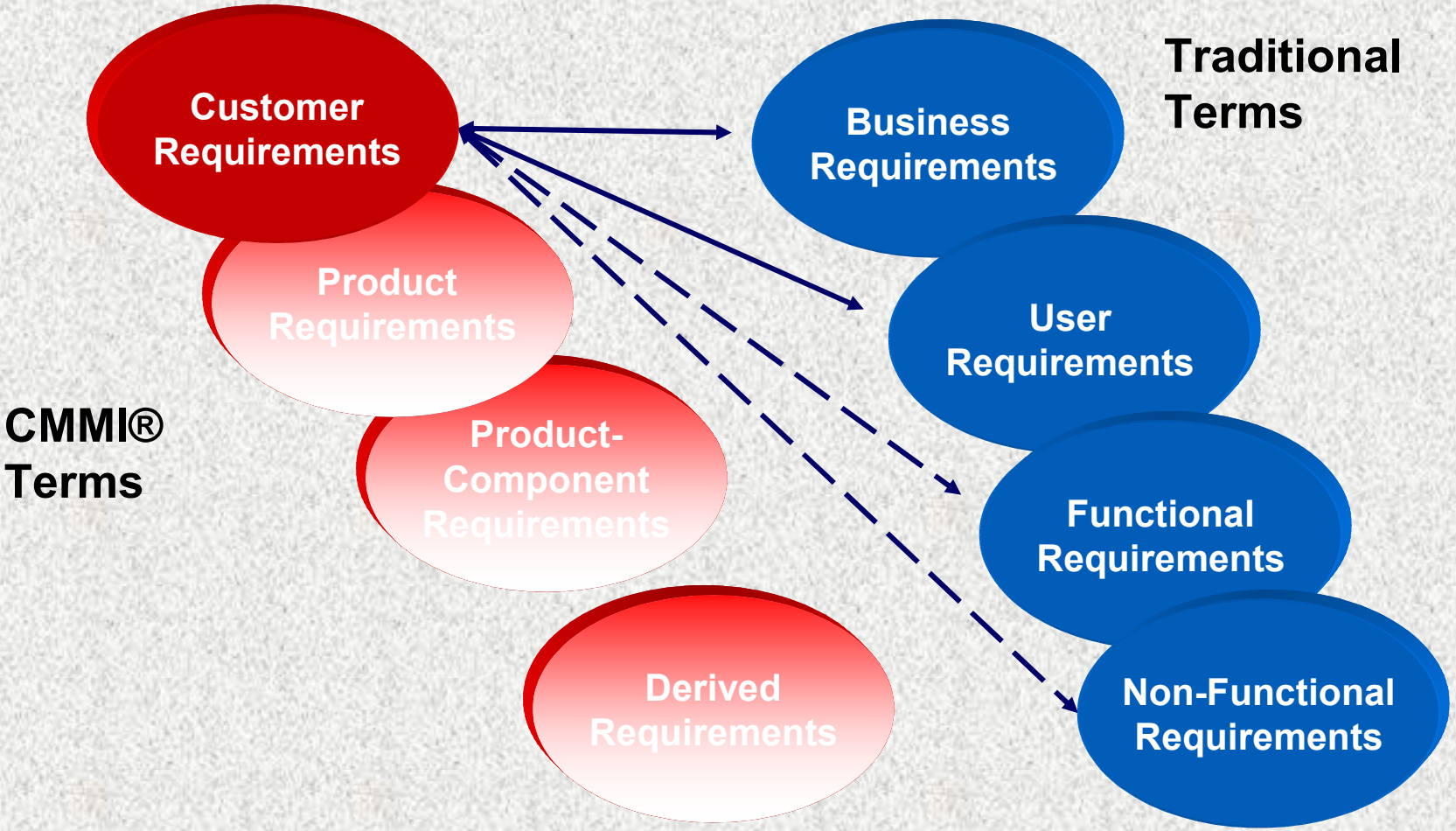
Relating CMMI® Requirements Categories



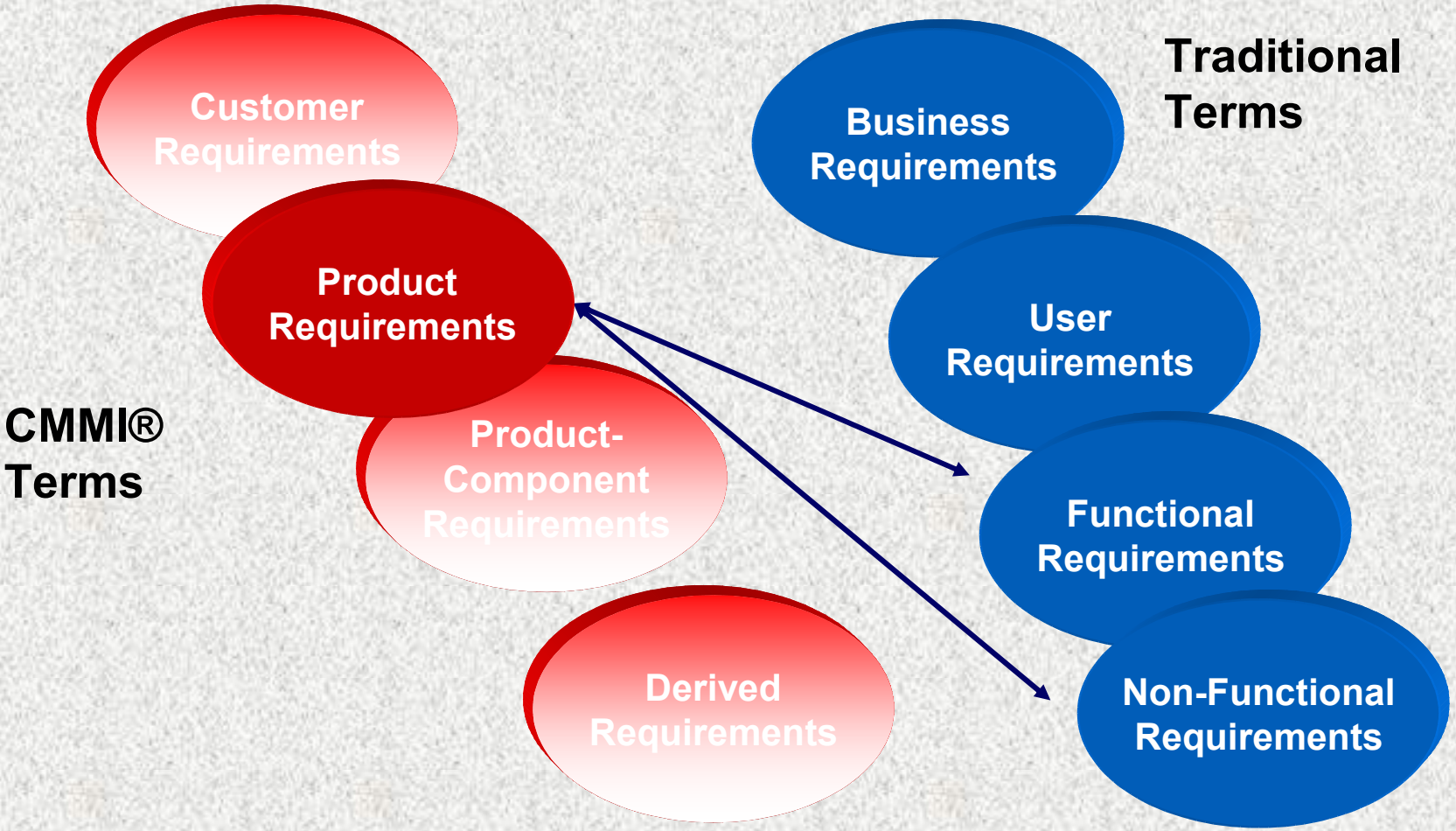
Relating Two Schemas



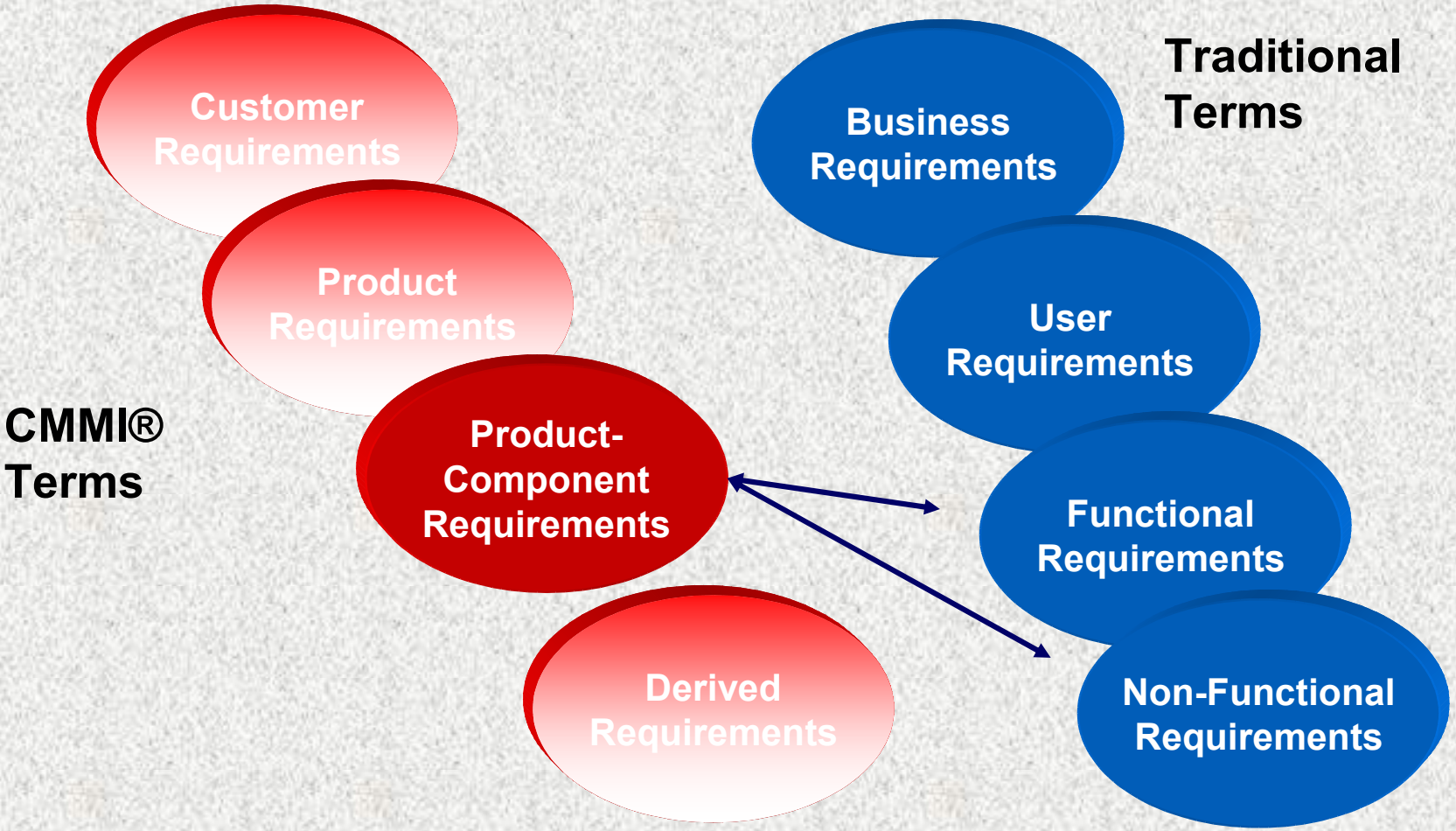
Relating Two Schemas



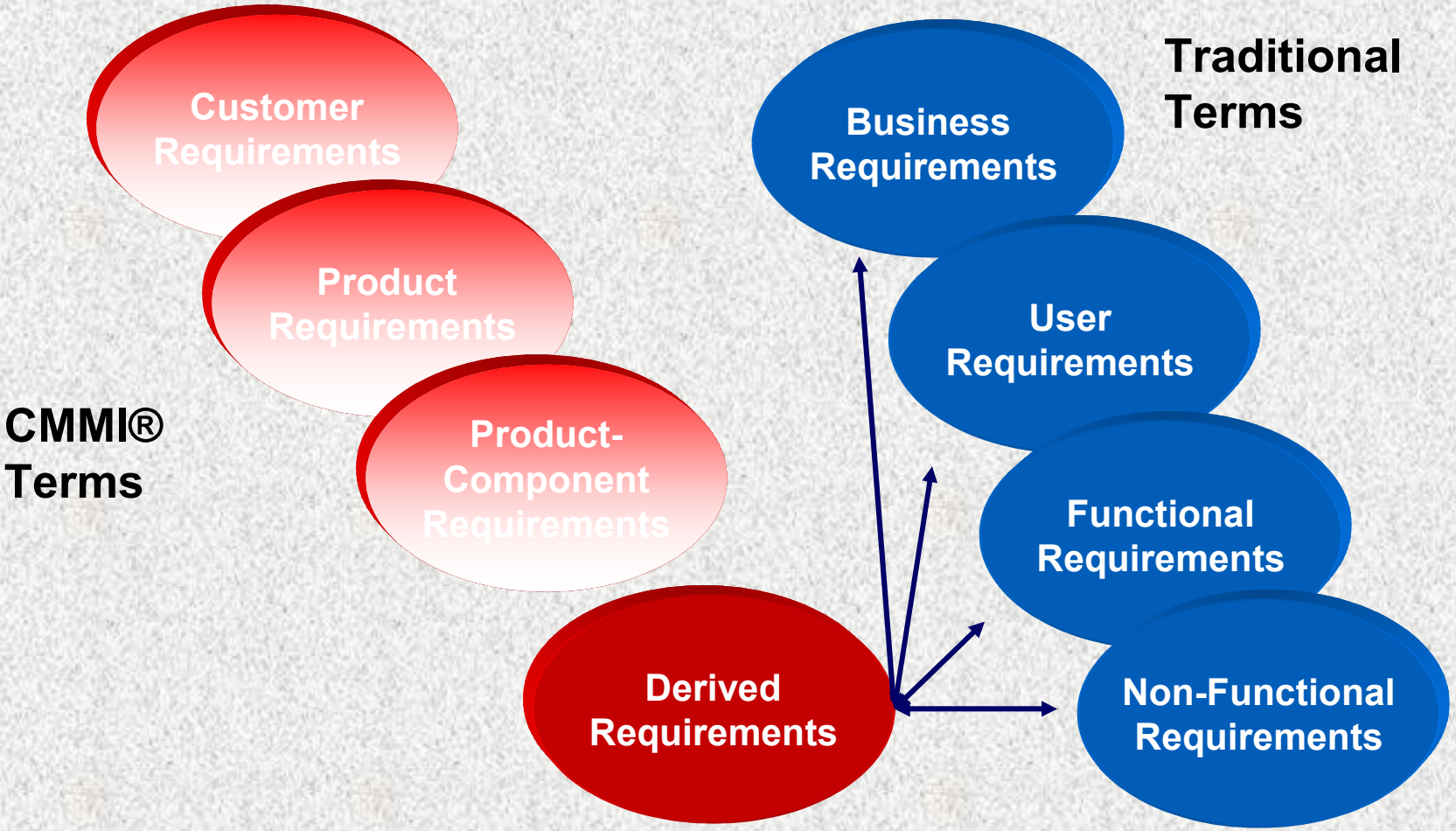
Relating Two Schemas



Relating Two Schemas

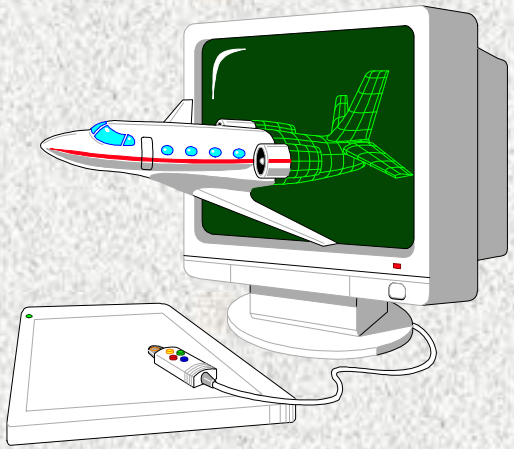


Relating Two Schemas



Carnegie Mellon
Software Engineering Institute

Implementing Requirements Management



Requirements Management Processes[1]

- Change Control
- Version Control
- Requirements Tracing
- Requirements Status
- Requirements Measures

Requirements Management Processes[2]

- **Change Control Processes – controlling and authorizing changes**
  - *Documentation and baseline of requirements*
  - *Submission and documentation of changes*
  - *Impact analysis and negotiation with stakeholders*
  - *Change Control Board Infrastructure*
  - *Update and recording of disposition of change request*

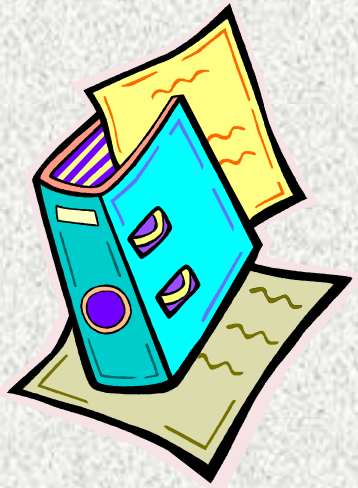
Requirements Management Processes[3]

- **Version Control Processes – ensuring correct version availability**
  - ***Configuration management of requirements repository***
  - ***Version maintenance and history throughout iterations***
  - ***Designated read, write, delete and update permissions***
  - ***Check In-Check out capability***
  - ***Labeling and annotation schemas***

Requirements Management Processes[4]

- **Requirements Tracing Processes – forward and backward requirements audit trail**
  - *Bidirectional linking to system elements*
  - *Capture of allocation rationale, accountability, and test/validation*
  - *Identification of inconsistencies*
  - *Capabilities to view/trace links*
  - *Verification of requirements*
  - *History of requirements changes*

[Kean, 1998]



Requirements Management Processes[5]

- **Requirements Status Processes – status of activity on requirements**
  - *Categories for status, e.g., proposed, approved, implemented, verified, deleted, and/or rejected*
  - *Methods of tracking*
  - *Escalation standards*

Requirements Management Processes[6]

- **Requirements Measures – metrics for requirements activities and status**
  - ***Requirements change requests – status, number, age***
  - ***Number of requirements in a particular status category***
  - ***Time spent on traceability and other requirements activities***

[Weigers, 2001]

Requirements Management Tools

- **Database-centric**
  - *Store all requirements, attributes, and traceability information in database*
  - *Examples are Caliber-RM, DOORS/ERS, RTM Workshop*
- **Document-centric**
  - *Treats word processing document as primary requirements container*
  - *May provide link to database or allow user to identify text as requirement*
  - *Examples are Requireit and RequisitePro*

Carnegie Mellon
Software Engineering Institute

Managing Customer Expectations

A Bill of Rights and a Bill of Responsibilities

Origin and Importance

- Developed by Karl E. Weigers for his book, Software Requirements
- Delineates what customer should expect from project team
- Clarifies what customer needs to commit to providing to project team

Customer Bill of Rights

1. Expect analysts to speak your language.
2. Expect analysts to learn about your business and your objectives.
3. Expect analysts to structure the information you present during requirements capture into a written software requirement specification.
4. Have developers explain all work products created from the requirements process.
5. Expect developers to treat you with respect and to maintain a collaborative and professional attitude throughout your interactions.
6. Have developers provide you with ideas and alternatives both for your requirements and for implementation of the product.
7. Describe characteristics of the product that will make it easy and enjoyable to use.
8. Be presented with opportunities to adjust your requirements to permit reuse of existing software components.
9. Be given good-faith estimates of cost, impacts, and trade-offs when you request a change in the requirements.
10. Receive a system that meets your functional and quality needs, to the extent that those needs have been communicated to the developers and agreed upon.

Customer Bill of Responsibilities

1. Educate analysts about your business and define business jargon.
2. Spend the time it takes to provide requirements, clarify them, and iteratively flesh them out.
3. Be specific and precise when providing input about the system's requirements.
4. Make timely decisions about requirements when requested to do so.
5. Respect a developer's assessment of the cost and feasibility of requirements.
6. Set priorities for individual requirements, system features, or use cases
7. Review requirements documents and prototypes.
8. Communicate changes to the project requirements as soon as you know about them.
9. Follow the development organization's defined process for requesting requirements changes.
10. Respect the processes the developers use for requirements engineering.

Bibliography

- Boehm, Barry W. (1981). Software Engineering Economics. Englewood Cliffs, NJ: Prentice-Hall.
- Crosby, Philip B. (2000). Define Quality? Philip Crosby Associates II, Inc.
- Forsha, Harry I. (1992) The Pursuit of Quality Through Personal Change. ASQC Quality Press.
- IEEE. (1998). IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications. Los Alamitos, CA: IEEE Computer Society Press.
- Kar, Pradip and Michelle Bailey. (1996). "Characteristics of Good Requirements." 1996 INCOSE Symposium.
- Kean, Liz. (1998). "Requirements Tracing – An Overview." Pittsburgh, PA: Software Engineering Institute.
- Leffingwell, Dean. (1997). "Calculating the Return on Investment from More Effective Requirements Management." American Programmer 10(4):13-16.
- Miller, Jason. (2003). "The Key Ingredient: Discipline." Government Computer News 22(32).
- Robertson, Suzanne and James Robertson. (1999). Mastering the Requirements Process. Great Britain: ACM Press.
- Rubin, Howard. (1999). "The 1999 Worldwide Benchmark Report: Software Engineering and IT Findings for 1998 and 1999, Part II." IT Metrics Strategies 5(3):1-13.
- Software Engineering Institute. (2003). Software Product Line Acquisition: A Companion to A Framework for Software Product Line Practice, Version 2.0. Pittsburgh, PA: Carnegie Mellon University.
- Stevens, Richard and James Martin. (1999). "What is Requirements Management?" QSS, Inc.
- Weigers, Karl E. (1999). Software Requirements. Redmond, WA: Microsoft Press.
- Weigers, Karl E. (2001). "Measuring Requirements Management – Getting to Know Your Requirements Data." Software Quality Engineering.
- Wilson, William M., Linda H. Rosenberg, and Lawrence E. Hyatt. Automated Quality Analysis Of Natural Language Requirement Specifications. Software Assurance Technology Center: NASA Goddard Space Flight Center.

Contact Information

Charlene C. Gross
Sr. Member Technical Staff
Software Engineering Institute
4301 Wilson Blvd.
Arlington, VA 22203
Ph. Number - 703-908-8205
Email – cgross@sei.cmu.edu

Questions?