REQUIREMENTS ACQUISITION

Collaborators: Steve Jacobs, John Georgas, Eck Doerry

OUTLINE

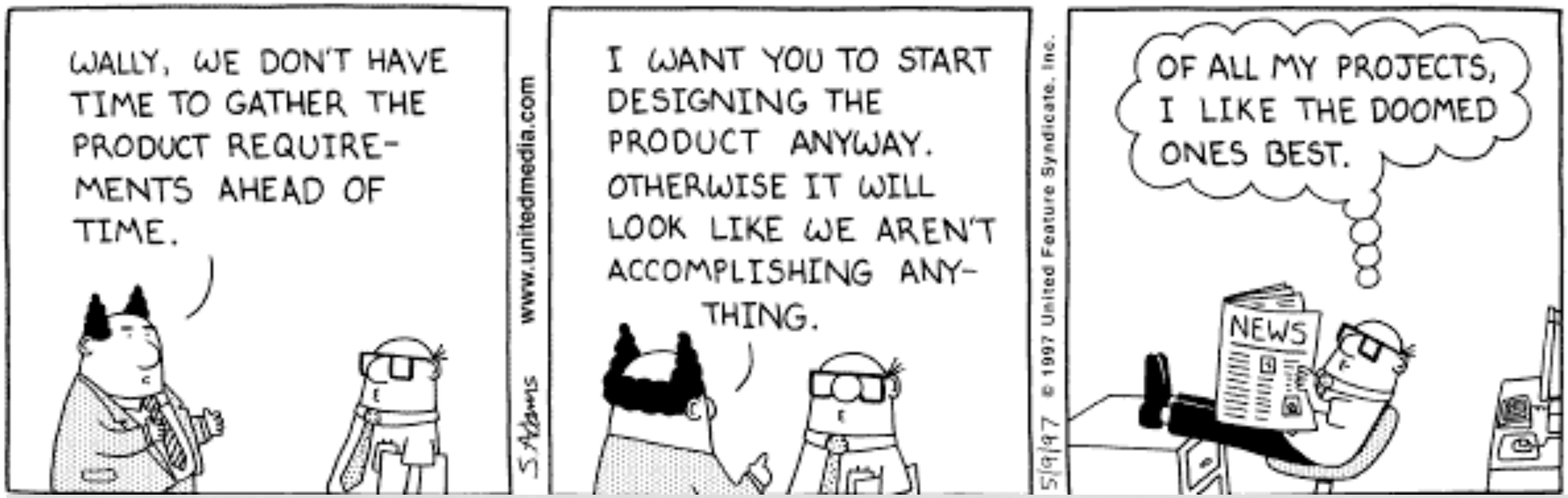
- **Elicitation**
- **Functional requirements**
- **Non-functional requirements**

http://vizconsult.wordpress.com/2011/03/09/requirements-definition/

- We are not going to talk about tools.
- Check INCOSE web site for tool information:

http://www.incose.org/productspubs/products/rmsurvey.aspx

WISDOM FROM DILBERT


WALLY, WE DON'T HAVE TIME TO GATHER THE PRODUCT REQUIREMENTS AHEAD OF TIME.
I WANT YOU TO START DESIGNING THE PRODUCT ANYWAY. OTHERWISE IT WILL LOOK LIKE WE AREN'T ACCOMPLISHING ANYTHING.
OF ALL MY PROJECTS, I LIKE THE DOOMED ONES BEST.
NEWS
www.unitedmedia.com
S.Adams
5/9/97 © 1997 United Feature Syndicate, Inc.

REQUIREMENTS: OVERVIEW

- **A high-level, *non implementation-specific* statement of a software system's intended functions or services**
  - Fundamental reference point between developer and customer
- **Functional requirements:**
  - Define system capabilities, what functions the system provides…
  - …without saying how they should be provided: What, not how
  - Impossible to do this perfectly, but worthwhile trying
- **Performance (non-functional) requirements:**
  - Computational performance
  - Usability (UI performance)
- **Other constraints:**
  - Platforms, implementation languages, etc.
  - Infrastructure that your piece must fit in.

REQUIREMENTS: QUALITY

According to the International Institute of Business Analysts (IIBA), **good requirements can be described via these criteria:**

- Requirements are complete. They must be as complete as possible with no open-ended parts or opportunity for interpretation.
- Requirements are testable. One must be able to create a test or some sort of proof that the requirement has been met.
- Requirements must be consistent with each other with no conflicts between what they are specifying.
- Requirements must be design-free. Software requirements should be specified in what the system must or must not do, but not in how the software will ensure the requirement is met; that's design.
- Requirements must be unambiguous. No wishy-washy statements nor (conceptually) anything that can be interpreted differently than intended.

REQUIREMENTS ELICITATION

- **Primary means:** ***Interviewing***
  - A structured discussion between client stakeholders and developer.
  - Covers all stakeholders, especially end-users!
- **Closed vs. open interviews**
  - **Open:** No pre-defined agenda
    - exploring issues
    - Used *early* to explore domains and challenges.
  - **Closed:** Set agenda of questions
    - Used to drill deeper as team fleshes out particular functional areas and builds business understanding.
- No meeting will be purely open or closed
    - Shift fluidly back and forth as needed.

"Your resume says you are very professional and have experience in requirements elicitation. Have you considered a career in fiction writing?"

http://www.modernanalyst.com

INTERVIEWING: BEST PRACTICES

- **Understand and clearly define the user domain**
  - Prepare ahead! Read up on client's business, the competition, the area.
  - Clarify all relevant domain terminology (domain dictionary)
  - You are not turning into domain experts, must understand the dynamics!
- **Avoid use of software-specific terms in discussions with the customer**
- **Limit open exploration that is leading nowhere…gently!**
- **Be prepared with specific questions, issues to explore**
  - Use this to structure the discussion, bring you back when it drifts
  - You are the expert in what you're missing! Drive the discussion!
- **Present specific options and alternatives**
  - Not "what do you want?" but "is this what you mean/want?"
  - Early prototype is invaluable

EVOLUTION OF REQUIREMENTS

Customer Requirements

- What the customer expects the product (e.g., system) to do.
- May include part of original project description.
- Generally derived from end-user requirements
- Focus on domain-driven descriptions of functions needed

Product Requirements

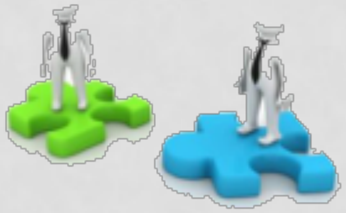
- Translation of the customer requirements into clear, concise, testable, verifiable requirements
- Addresses customer requirements and fills in technical elements needed to fully specify the product
- Good requirements-ese

Product Component Requirements

- Derivation of product requirements related to specific modules and product components.
- Basis for component design and implementation.

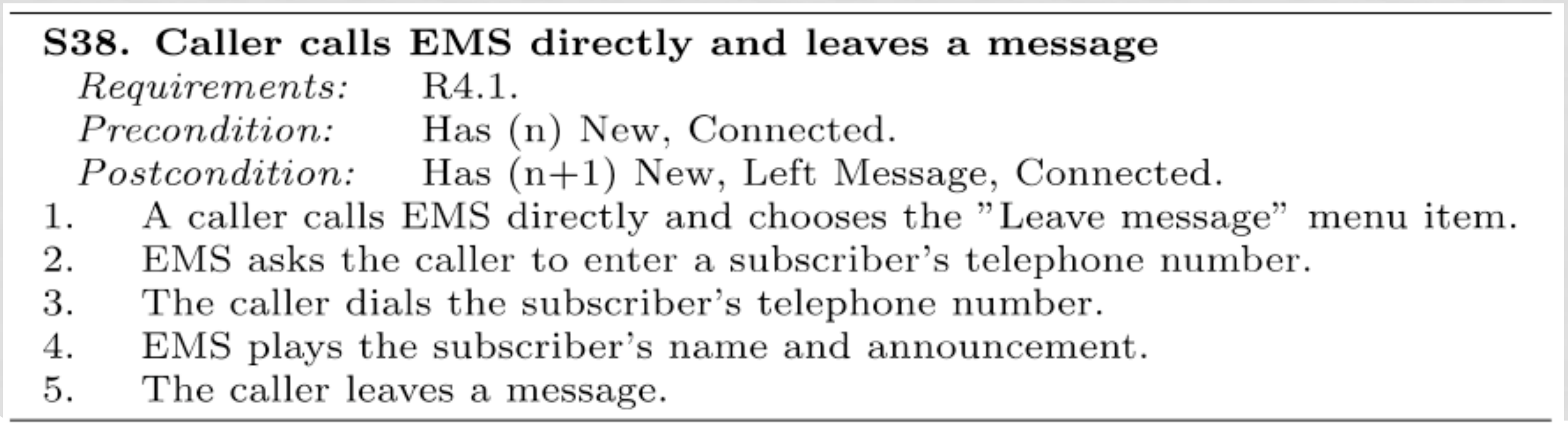
EVOLUTION OF REQUIREMENTS

- **Example 1: Domain-level user requirements**
  - "The user shall be able to search either all of the initial set of databases of hotels and itineraries or select a subset from it."
- **Example 2: Functional System Requirements**
  - "4.1 Administrator
    - 4.1.1 Login
      - 4.1.1.1 Administrators will be able to login with their username and password
      - 4.1.1.2 Administrators will be able to change their password
      - 4.1.1.3 Administrators will be able to create new user accounts"
- **Example 3: Product Component Requirements**
  - "Every order shall be allocated a unique identifier (ORDER_ID) which the user shall be able to copy to the order identifier text field."



USE CASES, USER STORIES

- **Basically: Step-by-step descriptions of specific usage scenarios**
  - Each use case covers a particular scenario
  - Necessarily incomplete. Can't trace all scenarios, but should cover all main ones
- **Elements:**
  - Informally: Descriptive walk-through of key usage scenarios.
  - Formally: Requirement association; goals; conditions; events.
  - Start with informal, via interviews; then transform to formal for write-up
- **See:** **http://www.cmcrossroads.com/article/defining-requirement-types-traditional-vs-use-cases-vs-user-stories?page=0%2C0**

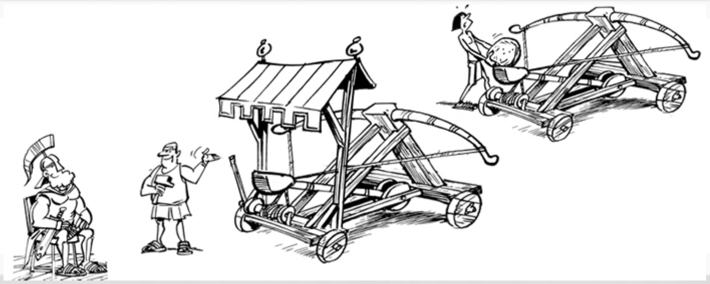


S38. Caller calls EMS directly and leaves a message

Requirements:	R4.1.
Precondition:	Has (n) New, Connected.
Postcondition:	Has (n+1) New, Left Message, Connected.

1. A caller calls EMS directly and chooses the "Leave message" menu item.
2. EMS asks the caller to enter a subscriber's telephone number.
3. The caller dials the subscriber's telephone number.
4. EMS plays the subscriber's name and announcement.
5. The caller leaves a message.

NON-FUNCTIONAL REQUIREMENTS



- Performance: Not the *what*, but the *how (fast/usable/etc.)*
  - Not just any implementation that technically provides all functions will do!
  - Documents expectations on speed and accuracy of system performance
    - Reliability
    - Computational performance
    - Usability; UI Performance
  - Focus on *clear numeric metrics, and measurable requirements!*

- Environmental Constraints
  - Describe context your software must live in
  - Constrain implementation options
    - Compatibility: Platforms, languages, interconnection
    - Evolvability: Requirements to ease future extension
    - Portability: platform independence, cross-platform (e.g. browser) function
    - Maintainability: code base management, documentation, commenting

HIGH-QUALITY REQUIREMENTS

- **How to we know that requirements are met?**
  - Is every functional requirement met in the implementation?
  - Are non-functional and performance requirements all met?

- **Verifiability**
  - The quality of being able to be verified or falsified
    - For requirements: ability to concretely verify each one
  - Informal verification: Check/document that all reqs. met
  - Formal verification: Experimental or empirical verification
    - Test suites/harnesses that have tests for each requirement
    - Ideally fully-automated
    - Usability must be empirically tested (user testing)

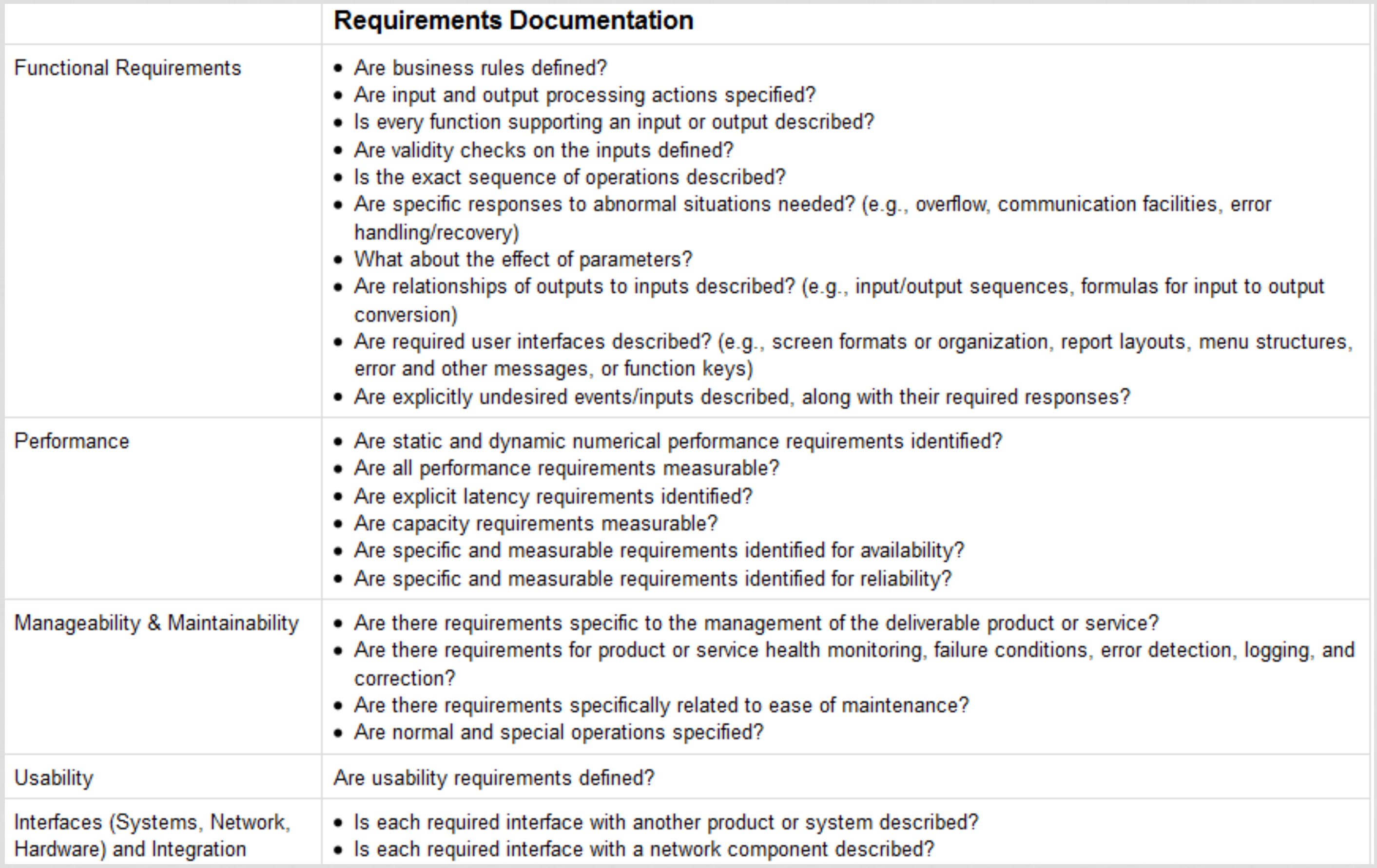
©KING, ModernAnalyst.com
WHY SHOULD I CLEAN MY ROOM?
WHAT'S THE CURRENT PROCESS FOR DUSTING?
ISN'T HOMEWORK MORE IMPORTANT?
HOW DO I KNOW WHEN IT'S CLEAN?
JUST FORGET IT.

http://www.modernanalyst.com

- ***Software Assurance***
  - a planned and systematic set of activities that ensures that software processes and products conform to requirements, standards, and procedures.

- **Goal: Build requirements that can be clearly verified!**

TOOLS: REQUIREMENTS CHECKLIST



	Requirements Documentation
Functional Requirements	• Are business rules defined? • Are input and output processing actions specified? • Is every function supporting an input or output described? • Are validity checks on the inputs defined? • Is the exact sequence of operations described? • Are specific responses to abnormal situations needed? (e.g., overflow, communication facilities, error handling/recovery) • What about the effect of parameters? • Are relationships of outputs to inputs described? (e.g., input/output sequences, formulas for input to output conversion) • Are required user interfaces described? (e.g., screen formats or organization, report layouts, menu structures, error and other messages, or function keys) • Are explicitly undesired events/inputs described, along with their required responses?
Performance	• Are static and dynamic numerical performance requirements identified? • Are all performance requirements measurable? • Are explicit latency requirements identified? • Are capacity requirements measurable? • Are specific and measurable requirements identified for availability? • Are specific and measurable requirements identified for reliability?
Manageability & Maintainability	• Are there requirements specific to the management of the deliverable product or service? • Are there requirements for product or service health monitoring, failure conditions, error detection, logging, and correction? • Are there requirements specifically related to ease of maintenance? • Are normal and special operations specified?
Usability	Are usability requirements defined?
Interfaces (Systems, Network, Hardware) and Integration	• Is each required interface with another product or system described? • Is each required interface with a network component described?

See entire Requirements Checklist example at:
https://wiki.cac.washington.edu/display/pmportal/Requirements+Checklist

GOOD REQUIREMENTS ARE *SMART*

- **Specific -**
  - It must address only one aspect of the system design or performance
  - It must be expressed in terms of the need (what and how well), not the solution (how).
- **Measurable -**
  - Performance is expressed objectively and quantitatively
  - E.g., an exact space telescope pointing precision requirement (in degrees) can be tested and thus verified prior to launch.
- **Achievable -**
  - It must be technically achievable at costs considered affordable
  - E.g., James Webb Space Telescope early designs specified an aperture requirement eventually de-scoped due to technical issues with deployment.
- **Relevant -**
  - It must be appropriate for the level being specified
  - E.g., requirement on the solar cells should not be designated at the spacecraft level.
- **Traceable -**
  - Lower level requirements (children) must clearly flow from and support higher level requirements (parents).
  - Requirements without a parent are referred to as orphans, and need to be assessed for necessity of inclusion.

SMART Requirements; Mike Mannion, Barry Keepence; ACM SIGSOFT, SE Notes Vol 20, No. 2, April 1995

WRITING GOOD REQUIREMENTS

When writing effective requirements, remember the following basic concepts:

- Make sure each requirement is necessary, verifiable, and achievable.
- Write clearly, simply, concisely and unambiguously.
- Make sure each requirement is unique and traceable.
- Use only one "shall" per statement.
- Specify "what's required," not "how to do it".
    - Do not specify a design constraint unless it is necessary to do so.
- Avoid buzz words and project-speak.
- Keep the language active and positive vs. passive and negative.
- Be consistent with your choice of phrasing throughout.
- Do not assume the reader will know what you meant. Focus on careful complete description. Repeat definitions as necessary.

CONCLUSION

- ***Clear and complete requirements are key to project success***
  - *Aligns client and develop understanding and expectations of function and performance*
  - *Gets you paid, helps you win in court.*

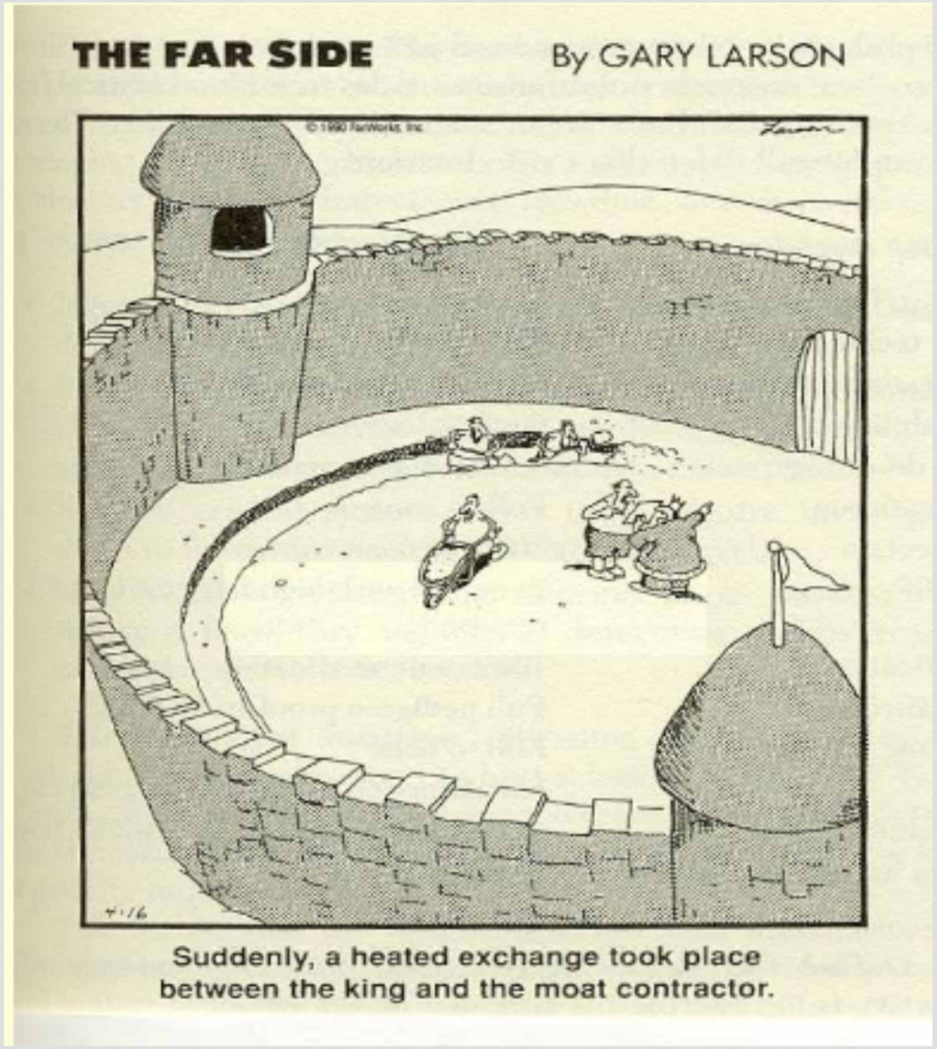
- ***It is hard to write good requirements! A real skill…***
  - *Important to have a clear process, strong structure, and commitment*
  - *Many guidelines and good reference sources*
  - *You get better with it as you practice it (painful mistakes…)*

- ***Learn to help your client understand what they really need!***
  - *You are the technical expert: Guidance on best cost/benefit solution*
  - *Desirement* – something that would be *nice to have* but is not mandatory for product success
  - Requirement – something that *must be done* for the product to be successful

http://www.incose.org/chicagoland/docs/WritingEffectiveRequirements.pdf

REQUIREMENTS: PARTING SHOT

Don't let this be you and your client!



Suddenly, a heated exchange took place between the king and the moat contractor.