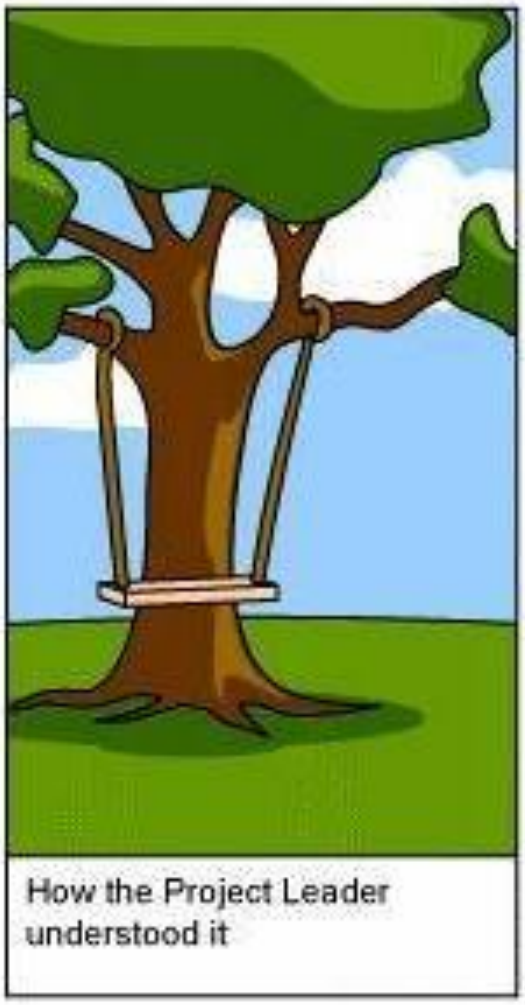
Software Requirements

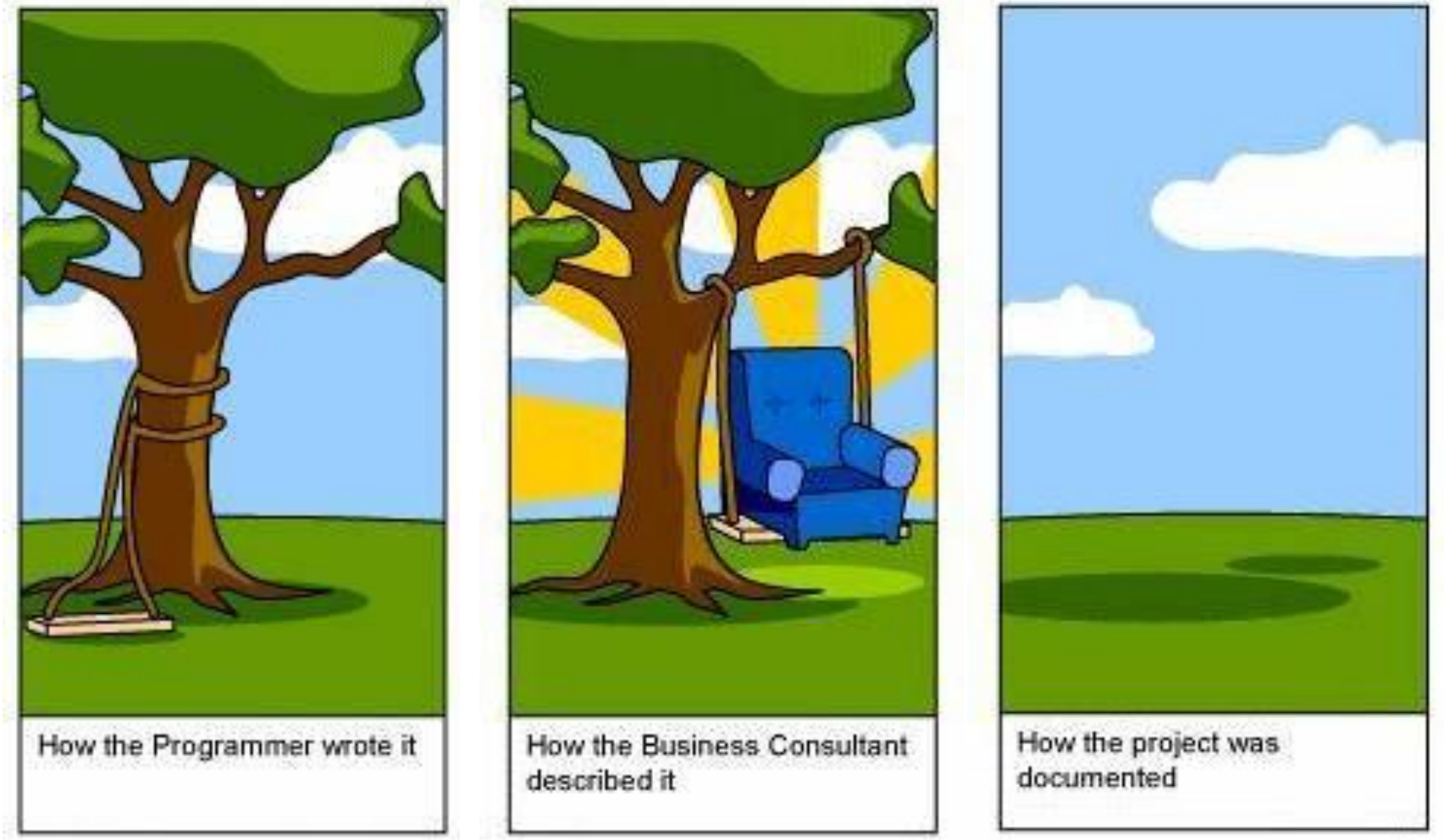
What's the big deal about requirements?

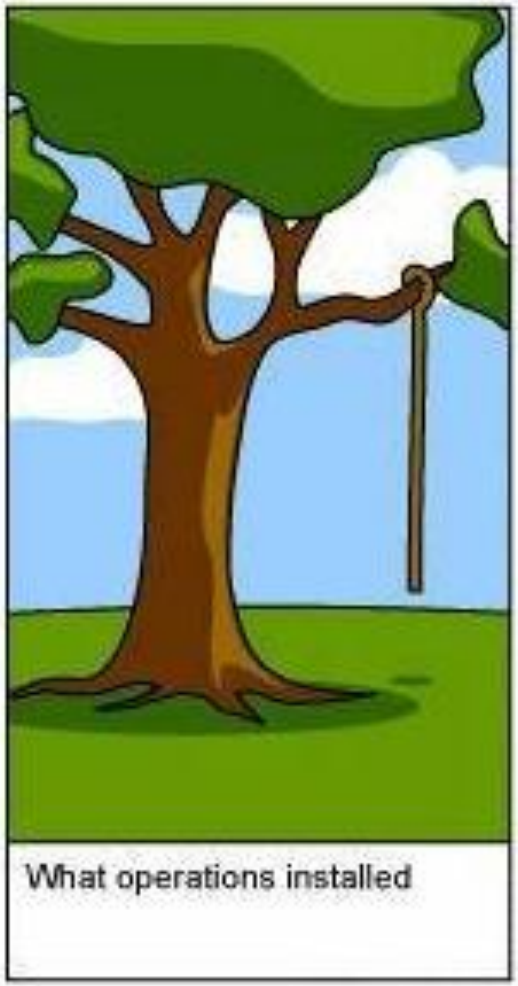
The Tree Swing Project

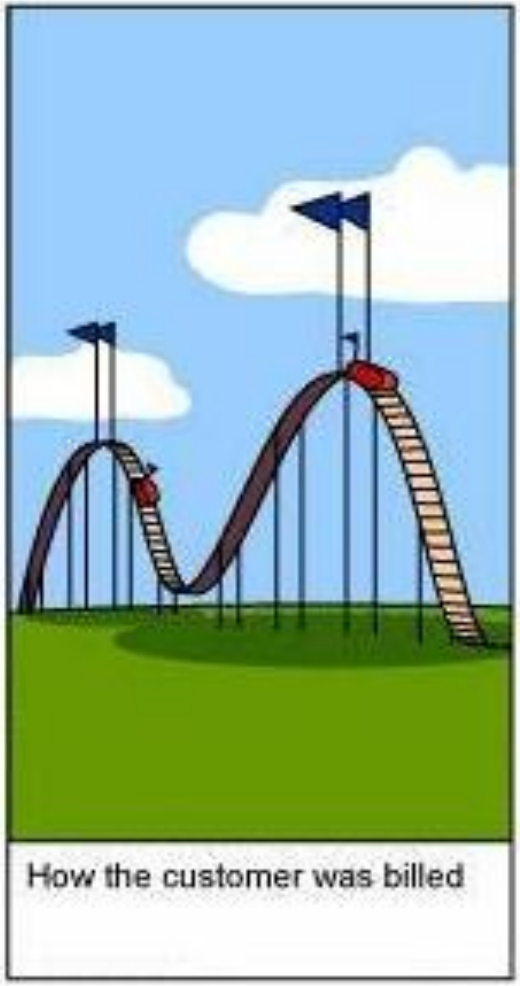

How the customer explained it

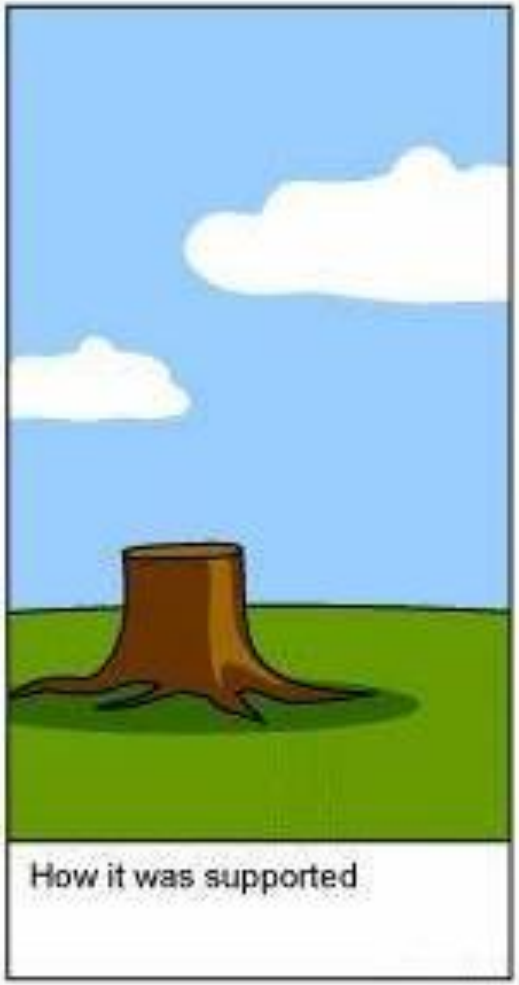

How the Project Leader understood it


How the Analyst designed it


How the Programmer wrote it
How the Business Consultant described it
How the project was documented


What operations installed


How the customer was billed


How it was supported


What the customer really needed

Requirements Engineering

Requirements are…a specification of what should be implemented. They are descriptions of how the system should behave, or of a system property or attribute. They may be a constraint on the development process of the system.

- Ian Sommerville and Pete Sawyer

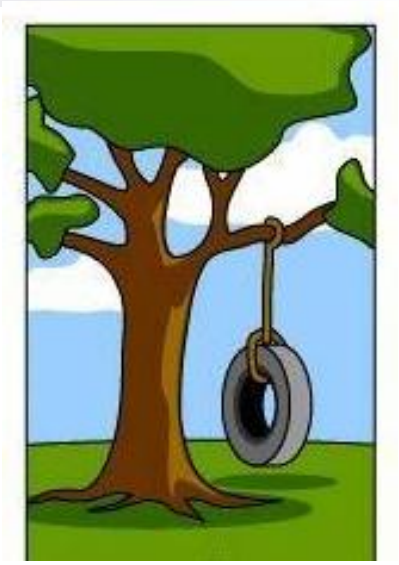
Understanding what you intend to build **before you're done** building it

- Karl Wiegers

Typical Requirements Activities

1) System scope definition
2) Requirements elicitation
3) Requirements specification
4) Open issues
5) Documentation
   - System Requirements Specification (SRS)
6) Validation
7) Requirements management

1) System Scope Definition



- What's in, what's out
- Tree Swing
  - Do we supply the tree?
  - Do we supply the hanging mechanism?
  - If not, how do we "interface" with them?
- Developer Responsibilities
  - Anything inside of the system
  - Any interfaces to external systems

2) Requirements Elicitation

- From whom?
  - Stakeholders
    - customer
    - developers
    - maintainers
    - end-users
    - … anyone else with a stake in the successful development and use of the system

- Automatic Teller Machine (ATM)
- Blackboard
- myUMBC (for you to do)

- How?
  - Interviews
  - Workshops/meetings
  - Surveys
  - Apprentice with the end-user
  - Prototyping

3) Requirements Specification

- What makes a requirement good?
  - What do we specify in a requirement?
  - How do we express it?
  - How do we know it's "good?"

Online Intro Programming Language Course

- The system shall provide quick feedback to quiz responses.
  - *The system shall provide feedback to quiz responses within 0.1 seconds.*
- The system shall provide a comprehensive help feature for the programming language's syntax.
  - *The system shall provide a help feature for all programming language syntax defined in "Introduction to Java Programming."*
- The system shall be user friendly.
  - ***Don't bother with this one!***

These requirements are not testable.

Online Intro Programming Language Course

- The system shall be user friendly.
- The system shall check for user input errors.
- The quiz questions collection shall not be corrupted upon system failure.

These requirements are obvious.

Online Intro Programming Language Course

- The system shall allow the user to take chapter quizzes, providing feedback after a quiz is completed and allowing the user to re-try any incorrect or incomplete problems.
  - *The system shall allow the user to take chapter quizzes.*
  - *The system shall provide feedback after a quiz is completed.*
  - *The system shall allow the user to re-try any incorrect or incomplete problems.*

This requirement is an **amalgamation** of requirements.

Online Intro Programming Language Course

The system should provide a table of contents for the online help feature.

- *The system **shall** provide a table of contents for the online help feature*.

- Avoid vague words such as “should,” “may,” “rapid,” “often,” “robust,” “optimize,” “intuitive,” “efficient”
  - Besides, they’re not testable either

This requirement is weakly worded.

Online Intro Programming Language Course

- The system shall allow a course administrator to grant course privileges to course users. He/she can then grant other lower level privileges.
  - *The system shall allow a course administrator to grant course privileges to course users.*
  - *The system shall allow a course administrator to grant …*

This requirement is ambiguous.

Characteristics of Good Requirements (Pfleeger)

- Are the requirements …
  - correct?
  - consistent?
  - complete?
  - realistic?
  - all needed by the customer?
  - verifiable?
  - traceable?

Requirements Expression

- *Natural language
  - English, etc.
- Structured natural language
  - *Use case specification
- Formal specification language
  - Backus-Naur
- Diagrams
  - Data flow diagram
  - State diagram
  - *Use case diagram
- Tables
  - Decision tables
  - State transition tables

*These are the ones that I want you to be concerned with.

Functional vs. Non-functional Requirements

- Remember *services* and *constraints*?
  - Functional (FR) Services
    - Describes an interaction between the system and its environment (Pfleeger)
    - *Gets the user closer to his/her end goal (Mitchell)*
  - Non-functional (NFR) Constraints
    - A restriction on the system that limits our choices for constructing a solution to the problem (Pfleeger)
    - *Is domain-independent (Mitchell)*

Examples – Let's Classify These

- Online Intro Programming Language Course
  - *The system shall provide feedback to quiz responses within 0.1 seconds.*
  - *The system shall provide a help feature for all programming.*
  - *The help feature shall use the language syntax defined in "Introduction to Java Programming."*
  - *The system shall allow the user to take chapter quizzes.*
  - *The system shall provide feedback after a quiz is completed.*
  - *The system shall allow the user to re-try any incorrect or incomplete problems.*

FR or NFR?

Typical NFR Categories

- Reliability/availability
- Security
- Documentation
- Training
- User interface
- Performance/response time
- Development standards
- Compatibility
- Portability
- Scalability
- Extensibility
- Other “ilities” (http://en.wikipedia.org/wiki/Ilities)

Let's Categorize These NFRs

- The user interface shall be text-based. User interface
- The system shall be password protected. Security
- A User Manual shall be provided. Documentation
- The system shall allow a minimum of 1,000 simultaneous users. Load
- The system shall be available 24 hours per day, 7 days per week. Availability
- A single five-hour classroom training session shall be provided. Training

Customer Constraints

- Conditions that the customer absolutely insists on
- Examples:
  - Particular hardware
  - Particular operating system
  - Particular user interface standards

4) Open Issues

- Anything that has not been decided or resolved by the “end” of the requirements phase
  - Document
    - Description of the issue
    - Resolution plan
    - Resolution date
  - Be honest

5) Typical Documentation

- System Requirements Specification (SRS)
  - Introductory material
    - What will be presented in the document?
    - Who is the intended audience?
    - What references were used for writing the document?
    - Who needs the system?
    - Why do they need it? What needs will it fulfill?
  - System scope
  - Functional and non-functional requirements
  - Customer constraints
  - Open issues
  - Deliverables

See the SRS template on the CMSC 345 web site

6) Validation

- Establishing that the requirements will meet the customer's needs
- Developer and customer review
- Review by other stakeholders
- Are the requirements
  - correct?
  - consistent?
  - complete?
  - realistic?
  - all needed by the customer?
  - verifiable?
  - traceable?

These were the earlier characteristics of a "good" requirement.

7) Requirements Management

- Prioritize
- Configuration management
  - Applies to many things:
    - source code
    - documents
    - requirements
    - other …
- Document or requirements numbering scheme
- Tools
  - Source code
    - CVS (open source), Subversion (open source), Visual SourceSafe (Microsoft), others
  - Documents
    - CVS (open source), RCS (open source), Google Docs, others
  - Requirements
    - spreadsheet, database, custom tools

References

- Pfleeger, Shari L., *Software Engineering: Theory and Practice*. 2nd ed. 2001, Upper Saddle River: Prentice Hall.
- Sommerville, Ian and Pete Sawyer, *Requirements Engineering: A Good Practice Guide.* 1997*:* Wiley.
- Wiegers, Karl, *When Telepathy Won't Do: Requirements Engineering Key Practices*. Cutter IT Journal, 2000.
- Wiegers, Karl, *Karl Wiegers Describes 10 Requirements Traps to Avoid*, Software Testing and Quality Engineering, 2000. **2**(1).