Software Requirements Part 1

Objectives

- Explain the concept of the CMM and CMMI process initiatives.
- Draw and explain the requirements engineering process
- Recognize the relationship between different types of requirements within the realm of software engineering
- Compare and contrast operational requirements, quality of service requirements, parametric requirements, design, and implementation requirements.
- Critique the wording of a requirement and constraints.

SEI CMM

- Capability Maturity Model - Integration
  - Software Engineering Institute
  - Carnegie Mellon University
    - 0 – Incomplete (chaotic, unpredictable)
    - 1 – Performed (weak process at best)
    - 2 – Managed (disciplined process)
    - 3 – Defined (standard, consistent)
    - 4 – Quantitatively Managed (predictable)
    - 5 – Optimizing (continuous improvement)

- Most organizations at level 0 or 1

Low levels

- Many now reaching levels 3-5
  - Motorola (most groups at level 3 or above)
  - Aerospace Firms (many at levels 4 & 5)
    - Winning contracts because their bids mean something

Quo

Status

- Mature process is reliable and high quality

Disciplined process

Where to start?

Software Development Related Problems

(Kung: Object Oriented Software Engineering)

- What is a requirement?

Something SW must meet or do

Function / Feature

Your Time

What is a requirement

Formal Definition

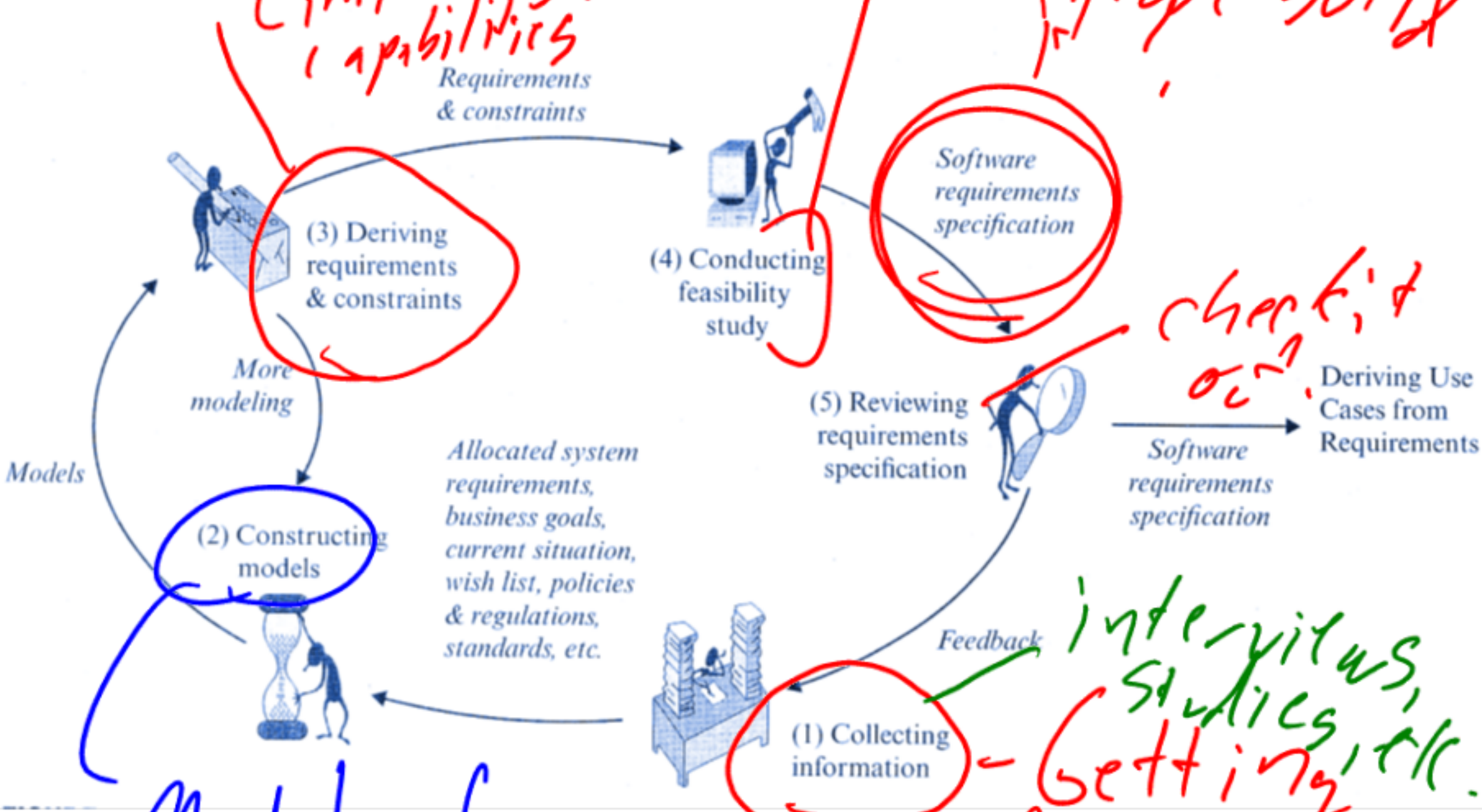
- "A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification or other formally imposed document. The set of all requirements forms the basis for subsequent development of the system or system component." [IEEE std.]

What is a requirement?

- Requirements are specifications of what a system must do
  - Tells you WHAT the system must do, not HOW to do it

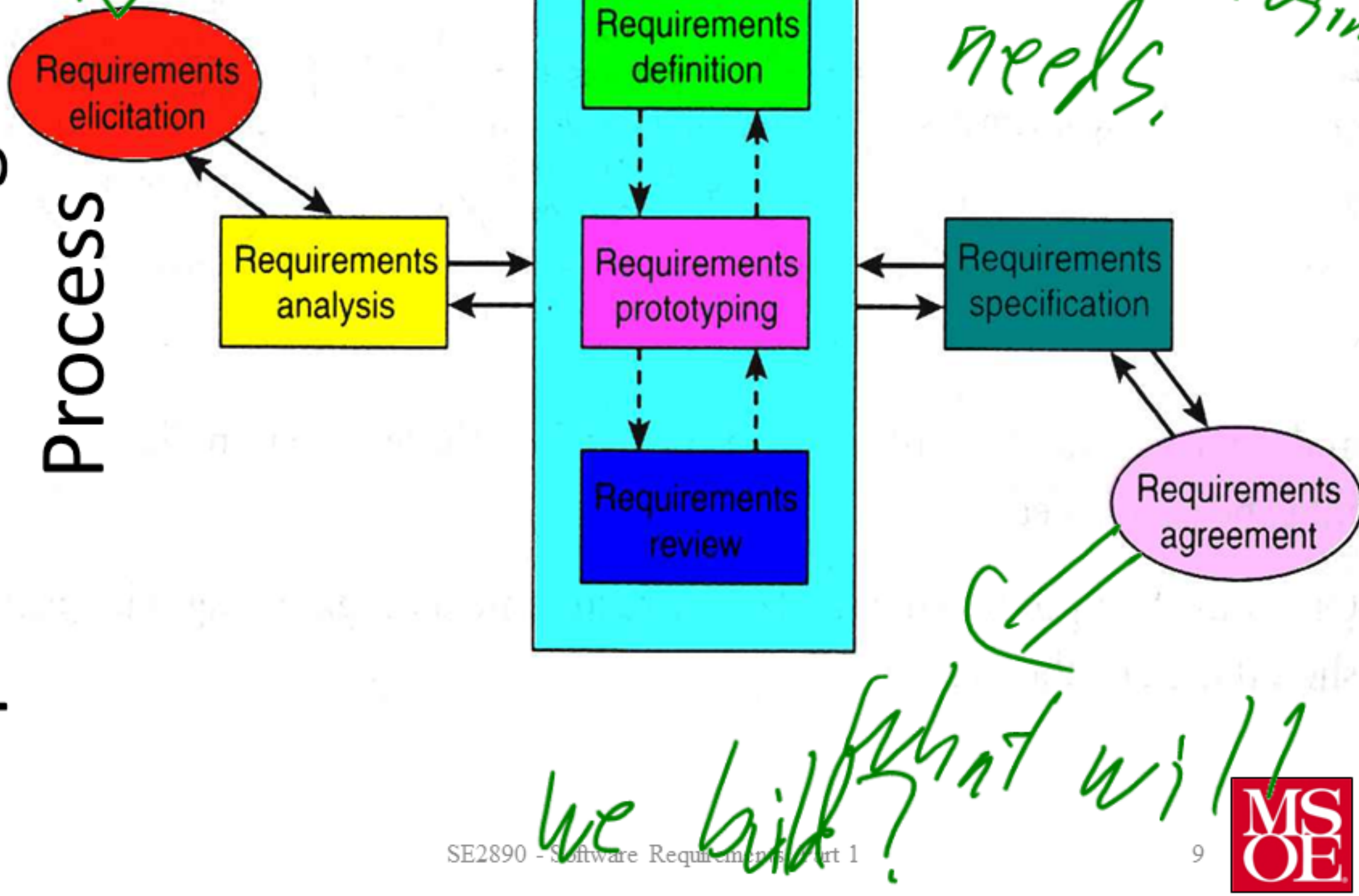
May overly constrain the system.

Requirements Definition Flow



Requirements Engineering Process

Requirements elicitation = figuring out what the customer needs.



we build what will?

Requirements Engineering

- Elicitation
  - Interviews and QA with the customer to determine what they want in their system
- Requirements Analysis
  - The requirement statements are checked for accuracy
  - requirements are categorized and prioritized
- Requirements definition
  - Requirements are formally written up
- Requirements prototyping
  - Construction of prototype systems to demonstrate to users
  - Typically done for UI portions of a system
- Requirements Review
  - Requirements are shared with the user for their input.
- Requirements specification
  - Final documentation of the requirements is delivered
- Requirements agreement
  - Customer signs off on the requirements document

Critique a Requirement

- With your partner, I want you to critique the following requirement for a desktop program…

Not a Good Requirement.

Scheduler 2[redacted]5 shall have a well defined human interface.

Vague / not descriptive

Good usability

Critique a requirement

- This is for a web based system that will keep track of swimming times for the MSOE Swim team.

Bad requirement - Microsoft

- The SwimSTAT product shall be developed in Visual Studio and shall be deployed on a Sun Solaris web server running Apache 1.0.

How to do one's job, not what.

A Second classification

- Functional Requirements
  - what the product must do (actions that it takes) to make it useful to the user; a behavior exhibited by the product
  - Example: The product shall produce an amended de-icing schedule when a change to a truck status means that previously scheduled work cannot be carried out as planned – From the text
- Nonfunctional Requirements
  - properties, qualities and attributes that the product must have
  - Example: All patient information stored by the system must be kept private at all times.
- Constraints:
  - A restriction on the degree of freedom you have in providing a solution.
  - Example: The product must run on Windows 7

Good requirements

- **Correct**
- **Unambiguous**
- **Complete**
- **Consistent**
- **Ranked for Importance**
- **Verifiable**
- **Modifiable**
- **Traceable**

Good requirements

- **Correct** - This is like motherhood and apple pie. Of course you want the specification to be correct. No one writes a specification that they know is incorrect. We like to say - "Correct and Ever Correcting." The discipline is keeping the specification up to date when you find things that are not correct.
- **Unambiguous** - An SRS is unambiguous if, and only if, every requirement stated therein has only one interpretation. Again, easier said than done. Spending time on this area prior to releasing the SRS can be a waste of time. But as you find ambiguities - fix them.
- **Complete** - A simple judge of this is that is should be all that is needed by the software designers to create the software.
- **Consistent** - The SRS should be consistent within itself and consistent to its reference documents. If you call an input "Start and Stop" in one place, don't call it "Start/Stop" in another.
- **Ranked for Importance** - Very often a new system has requirements that are really marketing wish lists. Some may not be achievable. It is useful provide this information in the SRS.
- **Verifiable** - Don't put in requirements like - "It should provide the user a fast response." Another of my favorites is - "The system should never crash." Instead, provide a quantitative requirement like: "Every key stroke should provide a user response within 100 milliseconds."
- **Modifiable** - Having the same requirement in more than one place may not be wrong - but tends to make the document not maintainable.
- **Traceable** - Often, this is not important in a non-politicized environment. However, in most organizations, it is sometimes useful to connect the requirements in the SRS to a higher level document. Why do we need this requirement?

Requirements Feud

- I'll give you a requirement. You must
  - Categorize the requirement
  - Tell me whether it is a good requirement or not
  - How might the requirement be improved.

Function

"The product shall provide status messages at regular intervals not less than every 60 seconds."

Quality attribute

Kind: Functional

Ambiguous? What are status messages?

Clarity

Round #1

Round #1 -> Improved

- 1. Status Messages.
- 1.1. The Background Task Manager shall display status messages in a designated area of the user interface at intervals of 60 plus or minus 10 seconds.
- 1.2. If background task processing is progressing normally, the percentage of the background task processing that has been completed shall be displayed.
- 1.3. A message shall be displayed when the background task is completed.
- 1.4. An error message shall be displayed if the background task has stalled.

Windows Progress bar

Round 2

- The Over temperature LED shall be illuminated within 10ms of the thermocouple reading a temperature greater than 100 degrees C or the reception of an Over Temperature Diagnostic Test message via the network interface.

- Audio shall change to the selected source quickly when a source change is necessary.

Type: Functional Usage Usage

Roun d3

Not very solid

Round 4

- All source code for project Whizbang shall be developed using ISO C 9899-1999 without extensions.

Constraint (Limiting) providing point of view

⇒ How to do one's job.

Round 5

- Web application should be able to serve the user queries as early as possible and within acceptable latency measurements.

Other guidelines for requirements

- Keep sentences short
- Read from the developers perspective
- Write testable requirements
- Write requirements individually
- Write at a consistent level of detail
- Avoid redundancy

Other guidelines for requirements

- Keep sentences short
  - Active voice, proper grammar
- Read from the developers perspective
  - If you need clarification, so will the developer
- Write testable requirements
  - If you can not test it, its not a good requirement
- Write requirements individually
  - And and or tend to indicate requirements have been aggregated
- Write at a consistent level of detail
- Avoid redundancy
  - Problems with document maintenance

What do we do with requirements?

- Software Requirements Specification (SRS)
  - Complete description of the behavior of the software to be developed.
  - Includes use cases that describe all of the interactions that the users will have with the software.
  - functional requirements
    - define the internal workings of the software
  - nonfunctional requirements
    - impose constraints on the design or implementation

Use Case Diagram