# Design and Analysis of Software Systems

## Requirements Engineering

## (Week 5)


INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

HYDERABAD

# Why Requirements Engineering ?

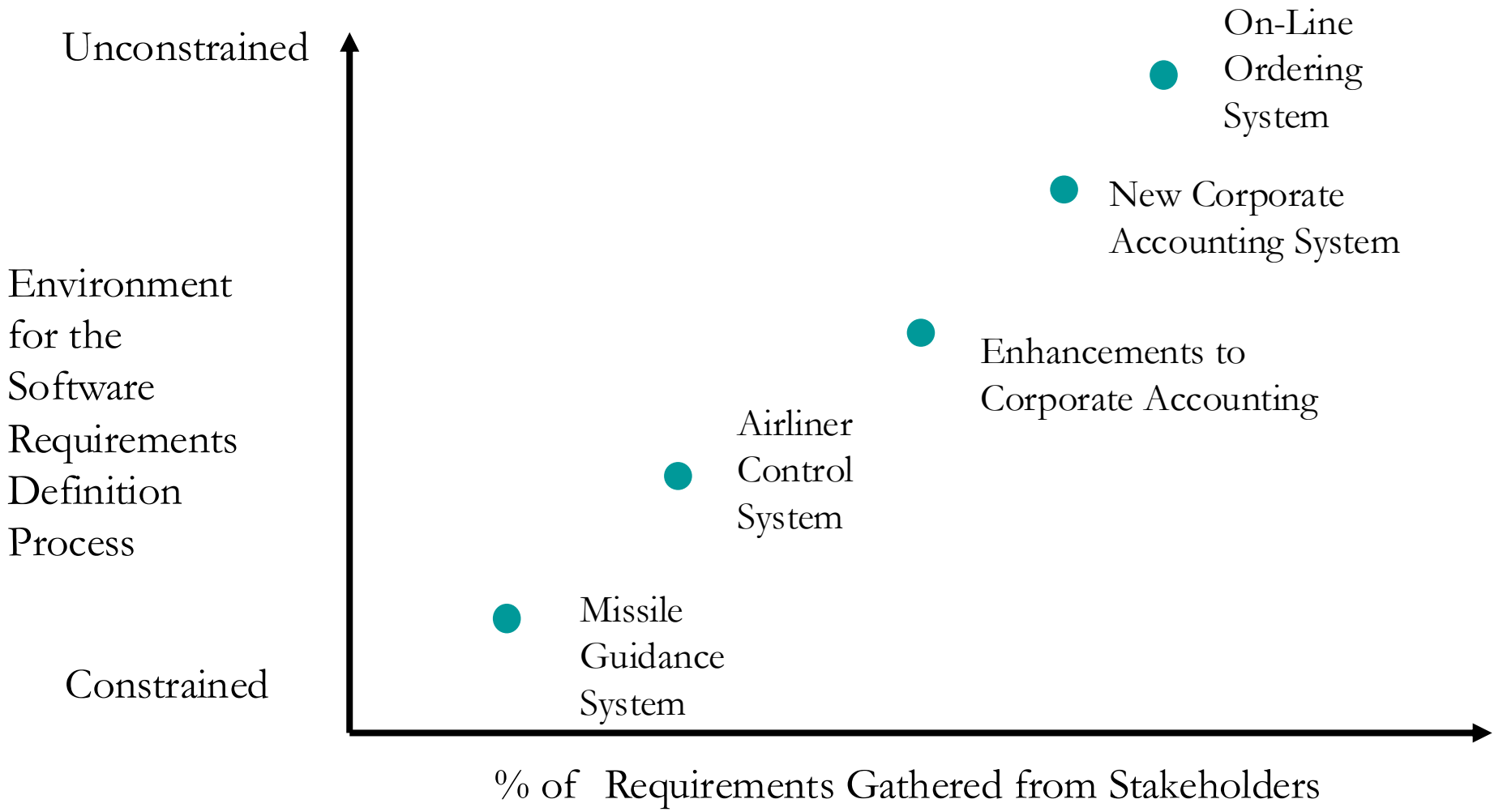
A **requirement** is a statement that specifies what a system must do or qualities it must possess to satisfy stakeholder needs.

Factors that cause projects to fail

- Requirements/Planning → 30 – 40%
- Project Management/Estimation → 20 – 30%
- People & Organization → 15 - 25%
- Technical/Design → 10 - 20%
- Testing/Quality Assurance → 5- 15%
- Change Management/Organizational Adoption → 5 -15%

# Sources of Requirements

Stakeholders, Existing systems/artifacts, external constraints/regulations, market research, technical & organizational sources, etc

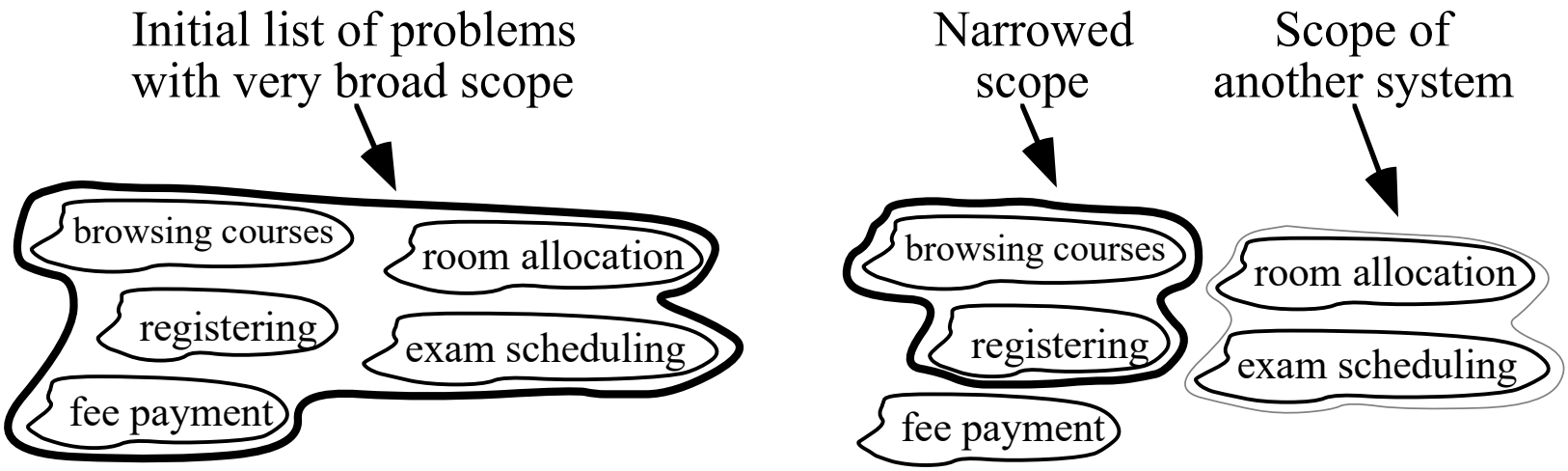


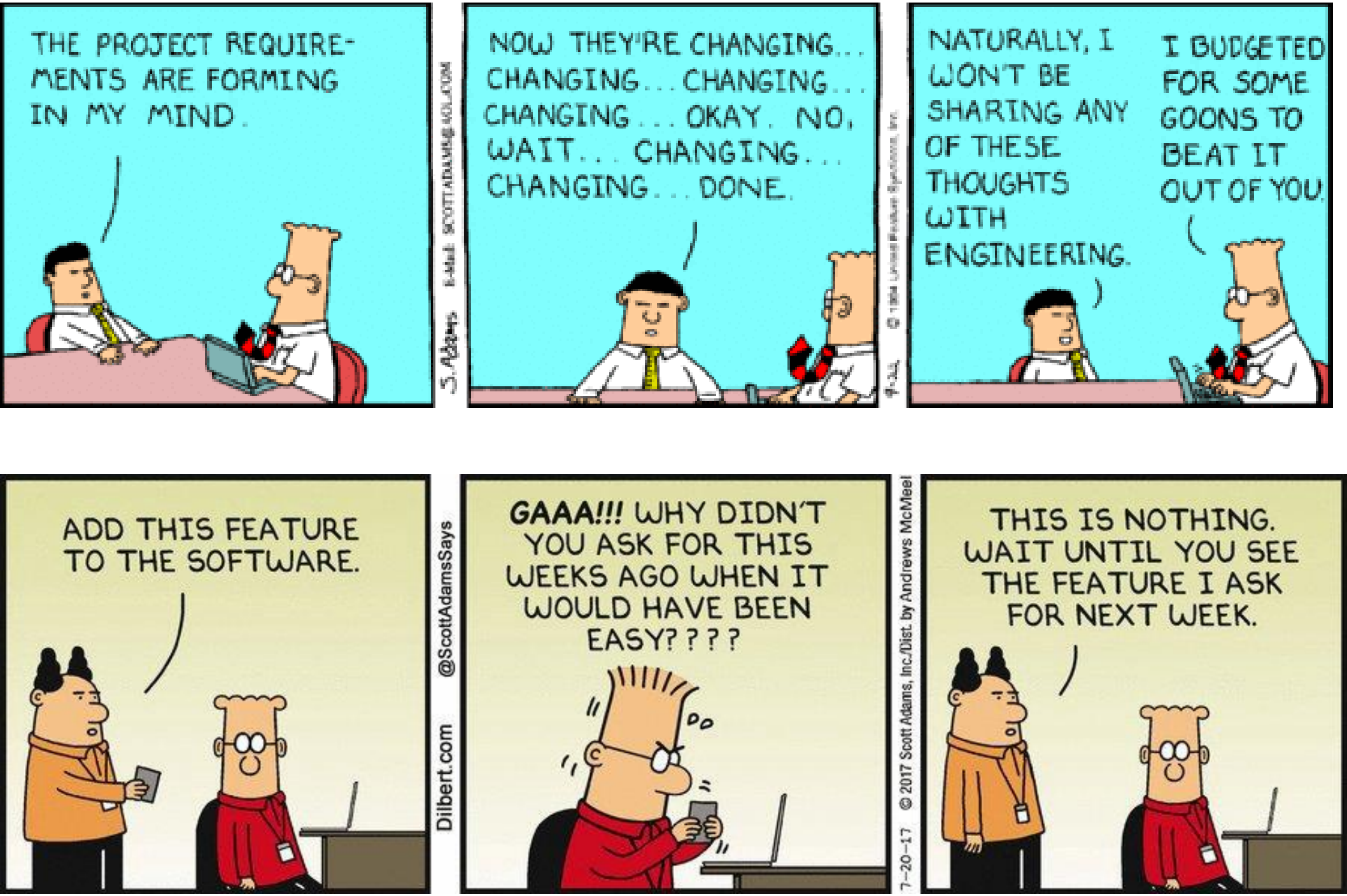

*Source: Software Requirements – SEI-CM-19-1.2, John Bracket*

# Defining the Scope

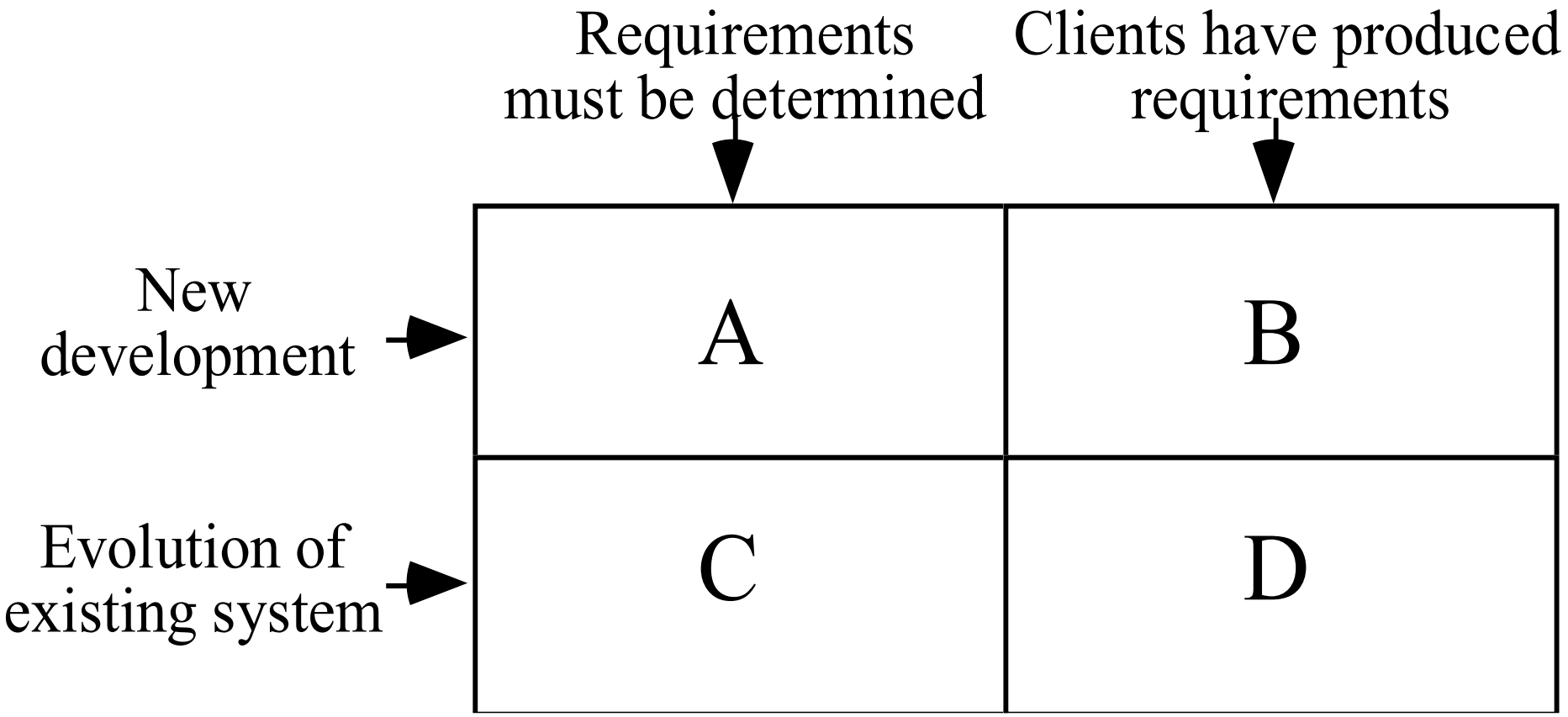
- Narrow the *scope* by defining a more precise problem
  - List all the things you might imagine the system doing
    - Exclude some of these things if too broad
    - Determine high-level goals if too narrow

- Example: A university registration system




THE PROJECT REQUIREMENTS ARE FORMING IN MY MIND.
NOW THEY'RE CHANGING... CHANGING... CHANGING... OKAY. NO, WAIT... CHANGING... CHANGING... DONE.
NATURALLY, I WON'T BE SHARING ANY OF THESE THOUGHTS WITH ENGINEERING.
I BUDGETED FOR SOME GOONS TO BEAT IT OUT OF YOU.
ADD THIS FEATURE TO THE SOFTWARE.
Dilbert.com @ScottAdamsSays
GAAA!!! WHY DIDN'T YOU ASK FOR THIS WEEKS AGO WHEN IT WOULD HAVE BEEN EASY????
7-20-17 © 2017 Scott Adams, Inc./Dist. by Andrews McMeel
THIS IS NOTHING. WAIT UNTIL YOU SEE THE FEATURE I ASK FOR NEXT WEEK.

# The Starting Point for Software Projects



# Types of requirements

- Requirements are broadly categorized into two types:
  - **Functional requirements**
  - **Non-functional requirements**

There may be other requirements from "Domain", "Business", "Transition", etc.

# Functional Requirements

Specifies behaviors/services (describe *what* the system should do)

Example: In e-commerce website, a functional requirement is

**'The system shall allow users to add items to their shopping cart.'**

Or

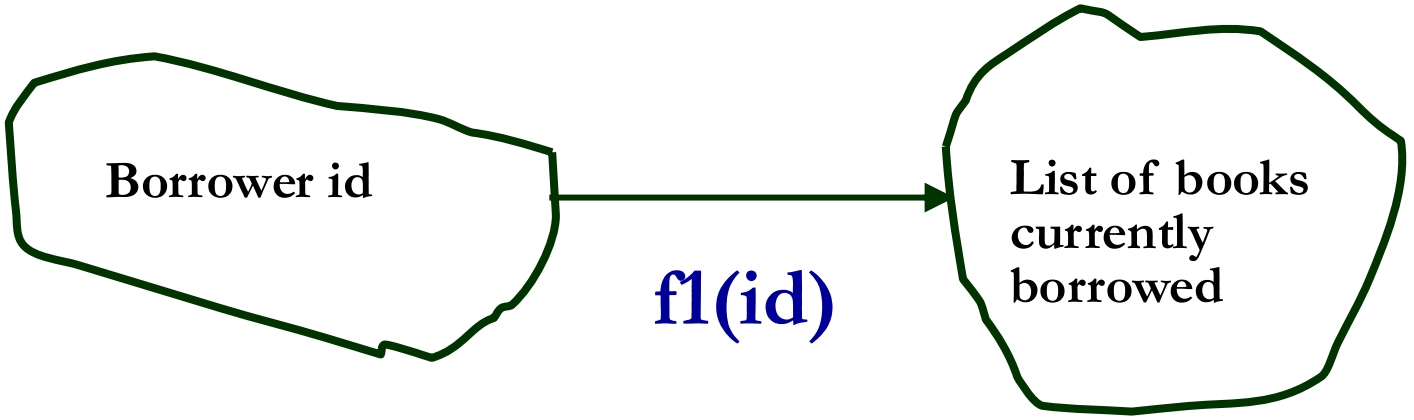
**'The system shall process refunds when requested by authorized personnel.'**

Clearly specifying functional requirements can help reduce **scope creep** !

# Example: Functional Requirement

## f1: find Books

- Input:
  - a borrower's id:
- Output:
  - details of the books that the borrower has currently checked out.



# Non-Functional Requirements (NFR)

Include characteristics of the system which cannot be expressed as functions. Specifies Qualities/Constraints

- Performance, Security, Usability, etc.
- Technical constraints, Business constraints, Process constraints

Example: In e-commerce website, a non-functional requirement is

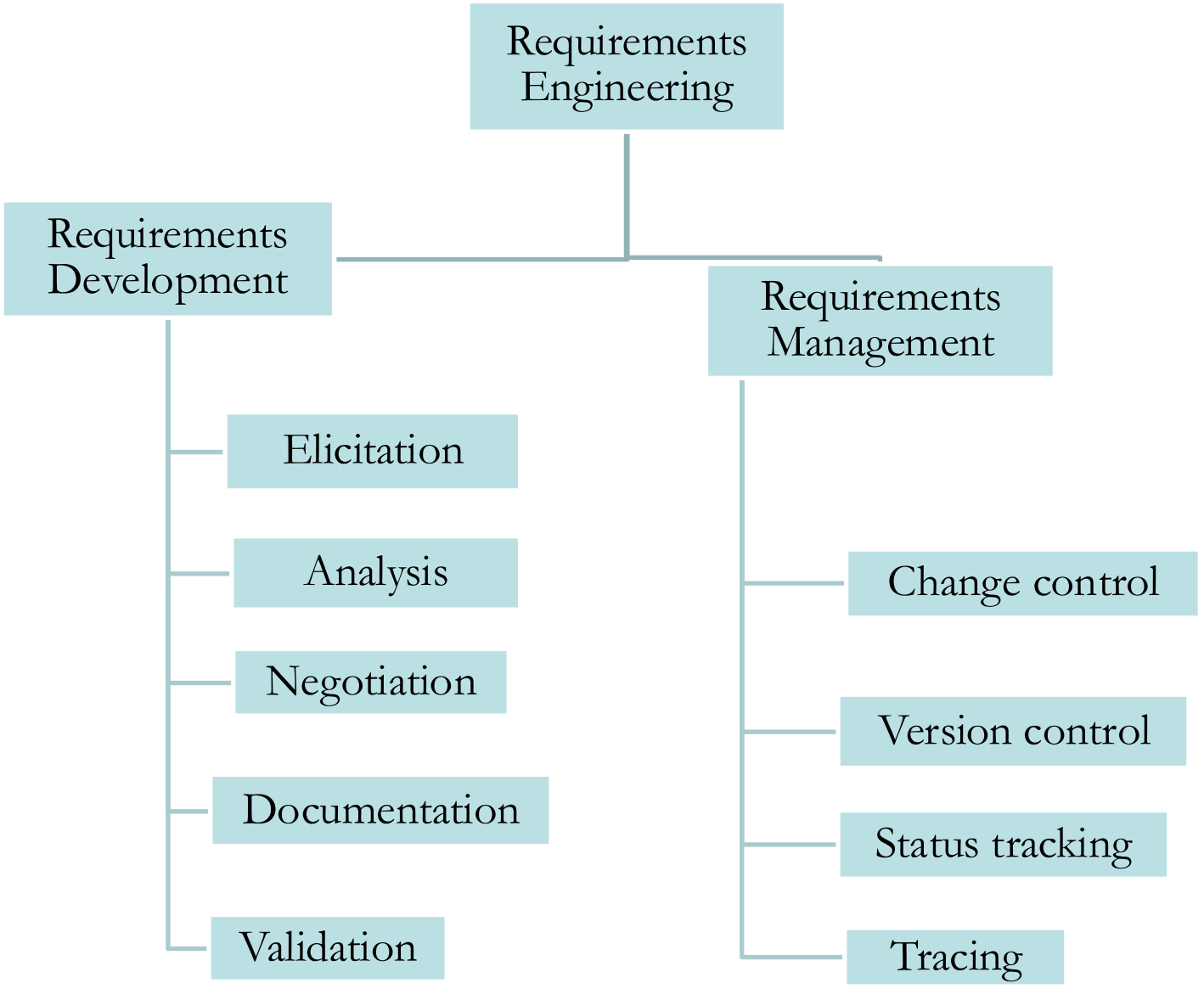
**'The system shall support at least 100,000 concurrent users.'**

Or,

**'The website shall be available 99.99 % of the time,'**

Clearly specifying non-functional requirements **can help measure** how well the system is doing what its supposed to do !

# Requirements - Activities



# Requirements Gathering and Analysis

# Requirements Gathering Activity

Activity: You are tasked to build a new campus library app

Each team (4 students) must write 4-5 features in the sheet of paper

MoSCoW Method (Must-have, Should-have, Could-have, Won't-have)

# Requirements Gathering/Requirements Elicitation

- **Interviews** - structured vs. unstructured; elicitation goals, question design.
  - Ask about specific details
  - Ask about the stakeholder's vision for the future
  - Ask if they have alternative ideas
  - Ask for other sources of information
  - Ask them to draw diagrams

  Example: interview product owner and lead developer to capture API requirements.

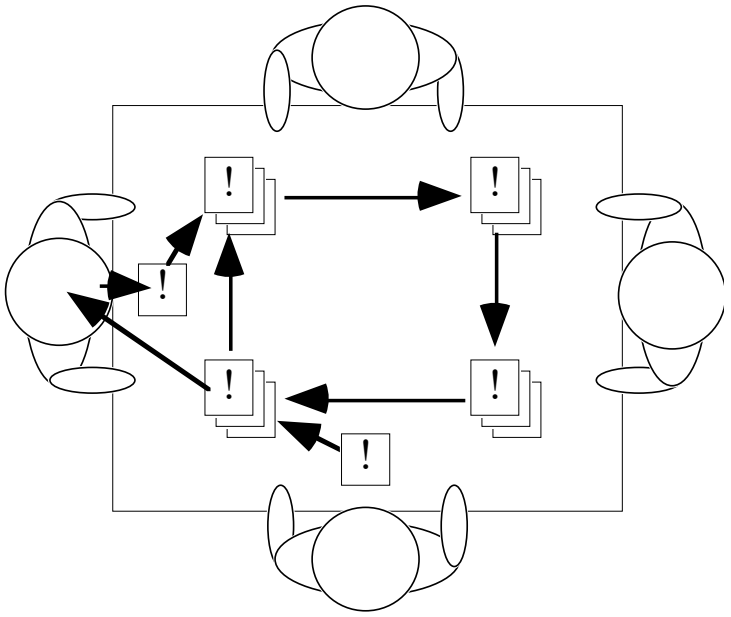
- **Observation / Contextual Inquiry** - Ethnographic methods, task analysis, capturing tacit knowledge.
  - Read documents and discuss requirements with users
  - Shadowing important potential users as they do their work - ask the user to explain everything he or she is doing
  - Session videotaping

  Example: observe staff using legacy LMS to identify pain points for migration.

# Gathering Requirements

**Workshops / Joint Application Development (JAD)** - facilitated sessions, role of moderator, resolving conflicting requirements.

- Appoint an experienced moderator
- Arrange the attendees around a table
- Decide on a 'trigger question'
- Ask each participant to write an answer and pass the paper to its neighbour



# Gathering Requirements

- Prototyping
  - The simplest kind: *paper prototype*.
    - a set of pictures of the system that are shown to users in sequence to explain what would happen
  - The most common: a mock-up of the system's UI
    - Written in a rapid prototyping language
    - Does *not* normally perform any computations, access any databases or interact with any other systems
    - May prototype a particular aspect of the system

# Example problems and requirements elicitation

University students miss assignment deadlines due to unclear submission workflows across LMS and email.

Contextual inquiry: shadow 5 students during one week of assignment handling to observe steps, tools used, and pain points; document task flows and exceptions.

Hospital nurses spend excessive time locating available infusion pumps, delaying patient care.

Workshops + Field observation: run a 2-hour workshop with nurses and biomedical engineers, then observe peak-shift workflows to validate device tracking requirements.

Small retailers cannot reconcile daily sales across POS and online storefronts, causing inventory mismatches.

Document analysis + Stakeholder interviews: review reconciliation reports and interview store manager + e-commerce admin to capture data fields, timing, and error cases.

# Requirements Gathering Table - Example

| Requirement ID | Requirement Title | Description | Priority | Type | Source / Stakeholder | Acceptance Criteria | Notes / Dependencies |
|---|---|---|---|---|---|---|---|
| R-001 | User Registration | Allow users to create an account using email and password, with email verification. | H | Functional | Product Owner, Marketing | User can register, receive verification email, and activate account; error shown for duplicate email. | Depends on SMTP service; GDPR consent checkbox |
| R-002 | User Login | Authenticate users via email and password; support "Forgot Password". | H | Functional | Product Owner | Users can log in with valid credentials; password reset email sent within 5 minutes. | Rate-limit login attempts |
| R-003 | Create Task | Users can create tasks with title, description, due date, priority, and tags. | H | Functional | End Users | Task saved and visible in user's task list with correct fields. | Requires user session |

# Requirements Quality Control

- less rework
- fewer unnecessary features
- lower cost
- faster development
- fewer miscommunication
- reduced scope creep
- better deliverables
- minimize change control

# Requirements Quality Control

- Goal: Check if the identified requirements represent all expectations about the system of various stakeholders, and do not contradict each other.
- Process: through Validation and Verification activities
- Metrics can be used

# Verification & Validation Activities

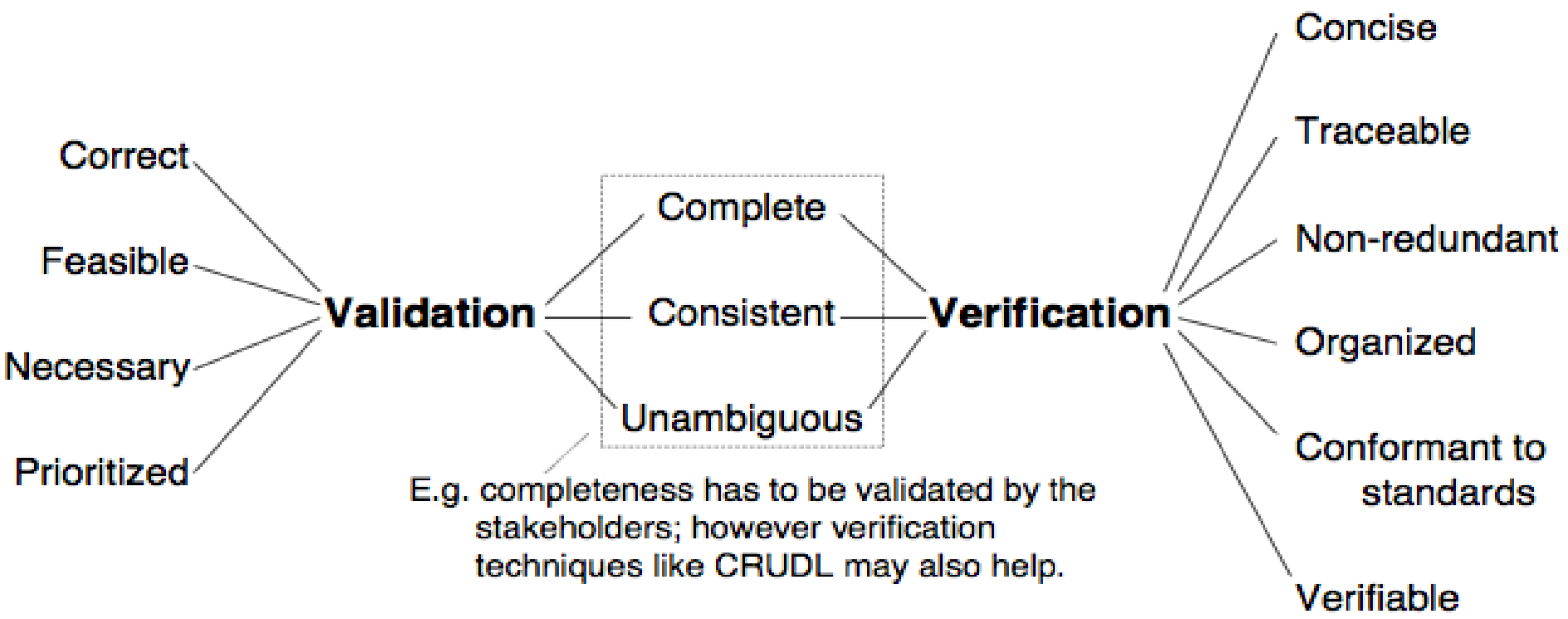
- Validation
  - "Am I building the right *product?*"
  - Checking a work product against higher level work products or authorities who framed this product
  - Focus on requirement qualities that affect the product
  - Requirements are validated by stakeholders

- Verification
  - "Am I *building* the product right?"
  - Checking a work product against some standards and conditions imposed on this *type* of product and the *process* of its development
  - Focus on requirement qualities that affect the development process
  - Requirements are verified by the analysts mainly

# Desired Qualities in Requirements

Purpose of requirements analysis:

- Clearly understand the user requirements,
- Ensure quality characteristics are met




***Look as a whole and at a statement level***

# Inconsistent Requirement

Some part of the  requirement contradicts with some other part.

Example (E-commerce site):

1. **Product Availability**:
    1. Requirement A: Products are displayed as available even if there's only one left in stock to create a sense of urgency.
    2. Requirement B: Users cannot add a product to their cart if the stock quantity is less than five.
2. **Pricing Rules**:
    1. Requirement C: Discounted prices should be displayed with the original price crossed out and the new discounted price prominently shown.
    2. Requirement D: Prices are automatically adjusted based on user behavior, with the goal of maximizing revenue

# Incomplete Requirement

Some requirements have been omitted - Possibly due to oversight.

- Example (E-commerce site):

  - Implement user authentication for the e-commerce website.

  **Enhanced Requirement:**

  Precondition: User must be registered in the system via an email id and password.

  "Users should be able to log in with their registered email and password. After three unsuccessful login attempts, users should be locked out of their accounts for 30 minutes. Implement password reset functionality that sends a password reset link to the user's registered email address"

# Bad Requirements…

- *A mail should be displayed within 3 seconds of clicking on mail*
- *User should be able to add a new mail server during peak hours within a small downtime*
- *Business services should not be interrupted during the peak hours*
- *User should be able to change the look and feel of how the mailbox is displayed*

Why do you think they are given as examples of bad requirements?

# Quality Requirements

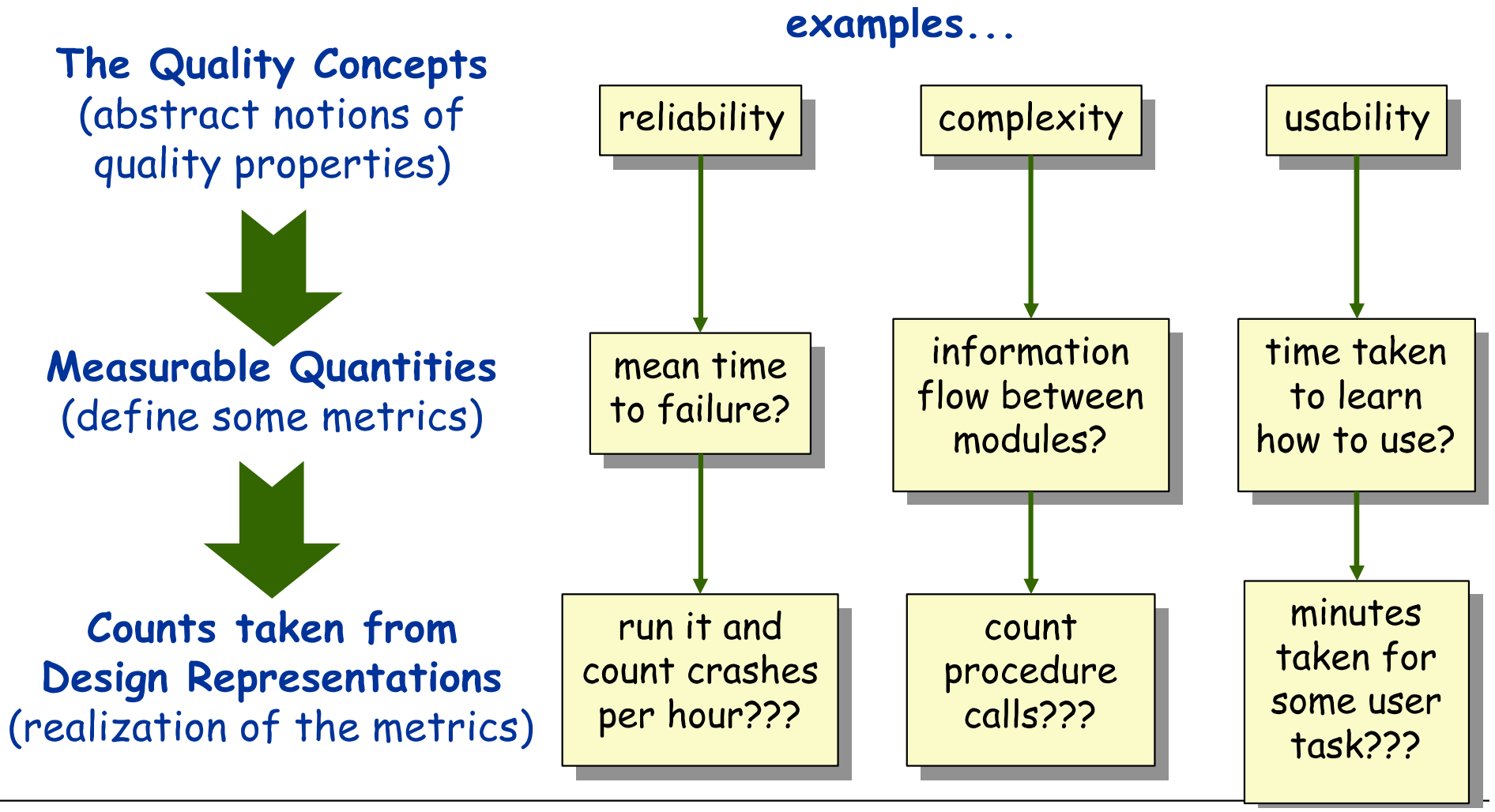
- Correct – only user representative can determine
- Feasible – get reality check on what can or cannot be done technically or within given cost constraints.
- Necessary – trace each requirement back to its origin
- Unambiguous – one interpretation
- Verifiable – how to you know if the requirement was implemented properly?
- Prioritized – function of value provided to the customer

# Quantifying Requirements

- The ability to identify measurable, mutually properties of similar requirements and the elicitation of corresponding values for each requirement

  - Auditability
  - Capacity
  - Configurability
    - Internationalization
    - Personalization
    - Variability
  - Correctness
    - Accuracy
    - Precision
    - Current
  - Dependability
    - Availability
    - Reliability
    - Robustness
    - Safety
    - Security
    - Survivability
  - Efficiency
  - Interoperability
  - Performance
    - Response time
    - Latency
    - Through put
    - Schedule-ability

# Quantifying Requirements



# Example Quantifications

| Quality | Metric |
|---|---|
| Speed | transactions/sec<br>response time<br>screen refresh time |
| Size | Kbytes<br>number of RAM chips |
| Ease of Use | training time<br>number of help frames |
| Reliability | mean-time-to-failure,<br>probability of unavailability<br>rate of failure, availability |
| Robustness | time to restart after failure<br>percentage of events causing failure |
| Portability | percentage of target-dependent statements<br>number of target systems |

# Writing Example #1

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

- Incomplete – What are the status messages and how are they supposed to be displayed?
- Ambiguous – What part of the product? Regular interval?
- Not verifiable

# Example #1 – Rewritten spec

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.

# *Writing Example #2*

"The product shall switch between displaying and hiding non-printing characters instantaneously."

- Not Feasible – computers cannot do anything instantaneously.
- Incomplete – conditions which trigger state switch
- Ambiguous – "non-printing character"

# Example #2 – Rewritten Spec

"The user shall be able to toggle between displaying and hiding all HTML markup tags in the document being edited with the activation of a specific triggering condition."

- Note that "triggering condition" is left for design

# The Specification Trap

*The Landing Pilot is the Non-Landing Pilot until the 'decision altitude' call, when the Handling Non-Landing Pilot hands the handling to the Non-Handling Landing Pilot, unless the latter calls 'go-around,' in which case the Handling Non-Landing Pilot continues handling and the Non-Handling Landing Pilot continues non-handling until the next call of 'land,' or 'go-around' as appropriate . In view of recent confusions over these rules, it was deemed necessary to restate them clearly.*

British Airways memorandum, quoted in *Pilot Magazine*, December 1996

# Difficulties and Risks in Requirements analysis

- Requirements change rapidly
  - *Perform incremental development, build flexibility into the design, do regular reviews*
- Attempting to do too much
  - *Document the problem boundaries at an early stage, carefully estimate the time*
- It may be hard to reconcile conflicting sets of requirements
  - *Brainstorming, JAD sessions, competing prototypes*
- It is hard to state requirements precisely
  - *Break requirements down into simple sentences and review them carefully, look for potential ambiguity, make early prototypes*

# Requirements Specification

# Software Requirements Specification (SRS)

Main aim of requirements specification:

- Systematically organize the requirements arrived during requirements analysis.
- Document requirements properly.

Can be documented in

- Natural language
- Structured natural language
- Formal specifications

# SRS Document

- The SRS document is known as black-box specification:
  - The system is considered as a black box whose internal details are not known.
  - Only its visible external (i.e. input/output) behavior is documented.


- SRS document concentrates on:

  **What** needs to be done

  Carefully avoids the solution ("**how to do**") aspects.

- SRS document serves as a contract

  Between **development team** and the **customer**

# Properties of a Good SRS Document

- Correctness
- Completeness
- Consistency
- Unambiguous / Precise
- Verifiable / Testable
- Modifiable / Maintainable
- Traceable
- Ranked for priority
- Understandable by stakeholders
- Feasible / Realistic
- Concise / Organized
- Uniform terminology and style
- Include functional and non-functional requirements
- Include acceptance criteria and success metrics
- Versioning and change history

# Natural Language SRS

| Field | Content |
|---|---|
| System | Online Bookstore — Customer Purchasing |
| Scope | Customers can browse, search, add books to cart, and checkout. |
| N-L1 | The system shall allow customers to create an account using their email address and a password. |
| N-L2 | Customers should be able to search for books by title, author, or ISBN and see results sorted by relevance. |
| N-L3 | The system shall process payments via credit card and display an order confirmation. |
| N-L4 | The system should be fast and user friendly. |

# Structured Natural Language SRS

| Field | Content |
|---|---|
| System | Online Bookstore — Customer Purchasing |
| ID | R-001 |
| Title | User Registration |
| Statement | The system shall allow a new customer to register using a unique email and password. |
| Preconditions | None |
| Postconditions | A customer account record is created; verification email queued. |
| Priority | High |
| Acceptance Criteria | Given a valid unused email and password meeting policy, when user submits registration, then account is created and verification email sent within 30 seconds; duplicate emails rejected with HTTP 409. |
| Notes | Structured fields improve clarity, testability, and traceability. |

# Structured Natural Language – example structure

As a [Type of USER],

[Function to Perform (some goal)]

so that [Business Value (some reason)]

Example: As a user, I can indicate folders not to backup so that my backup isn't filled up with things I don't need saved

# Use templates for Specifying Correctly

- The <System_name> shall <behavior> if <conditions>, where <quality factor>.
- Upon <conditions>, the <System_name> shall <behavior> where <quality factor>.
  - Ex: The ATM shall reject withdrawal requests if the amount requested is not divisible by 20.

- The system_name shall produce <output> for use by <nodes>, for use by <nodes>, if <conditions>, using <inputs/outputs> where <quality factor>.

  - The ATM shall produce a receipt for use by bank patrons if a transaction is completed. The receipt will include a unique transaction code, amount, date, time and location.

# Formal SRS

| Field | Specification |
|---|---|
| System | Online Bookstore — Shopping Cart Module |
| Types | BOOK = { isbn: String, title: String, price: Real } ; CART = sequence of BOOK |
| State (CartState) | cart: CART<br>total: Real<br>Invariant: total = Σ_{b ∈ cart} b.price |
| Operation: AddBook | Precondition: true<br>Input: b: BOOK<br>Postcondition: cart' = cart ++ [b] ∧ total' = total + b.price |
| Operation: RemoveBook | Precondition: 1 ≤ index ≤ |
| Operation: GetTotal | Precondition: true<br>Output: total = Σ_{b ∈ cart} b.price |
| Error Handling | AddBook: if b.price < 0 ⇒ error BAD_PRICE<br>RemoveBook: if index invalid ⇒ error INDEX_OUT_OF_RANGE |
| Invariants & Constraints | ∀ b ∈ cart: b.price ≥ 0 ; |

# Room Booking System (Graded Activity)

- The college wants a system to manage campus events and room bookings. The system should be easy to use and support students, faculty, and possibly external users. Users should be able to book classrooms, labs, seminar halls, and open spaces.
- The system should avoid booking conflicts, but important events may sometimes get priority. Some bookings may require approval, while others should be confirmed quickly. Events can be planned in advance or at the last minute, and any changes or cancellations should be communicated. Certain rooms have facilities like projectors or internet that should be considered.
- The system should be secure, handle many users without being slow, and be available most of the time. Reports on room usage would be useful, and the system should support future changes.

**Deliverables**

- **Functional Requirements** (8-10)
- **Non-Functional Requirements** (5–6)
- **Assumptions**
- **Out-of-Scope Items**