Software Requirements Elicitation

Dr John H Robb, PMP, IEEE SEMC
UT Arlington
Computer Science and Engineering

Key Takeaway Points

- Requirements are capabilities that the system must deliver.
- *The hardest single part of building a software system is deciding precisely what to build—i.e., the requirements*. (Frederick P. Brooks, Jr.)
- Software requirements elicitation is aimed to identify the real requirements for the system – this may not be the same as what the customer asked for!

- This module provides an overview of the requirements process and then focuses on the requirements specification - which is the focus of the project

Overview of the Requirements Process

The Requirements Process

The Requirements Process consists of the following steps

A. Requirements Planning (estimating requirements work)
B. Requirements Elicitation (draw-out the requirements)
C. Requirements Analysis (do they work and work together?)
D. Software Requirements Specification (capture requirements)
E. Requirements Validation
F. Requirements Management (requirements will change - they must be managed)
G. Requirements status reporting

Most of the industry is particularly weak in all but D and E above and many are weak here as well.

Questions to discuss:

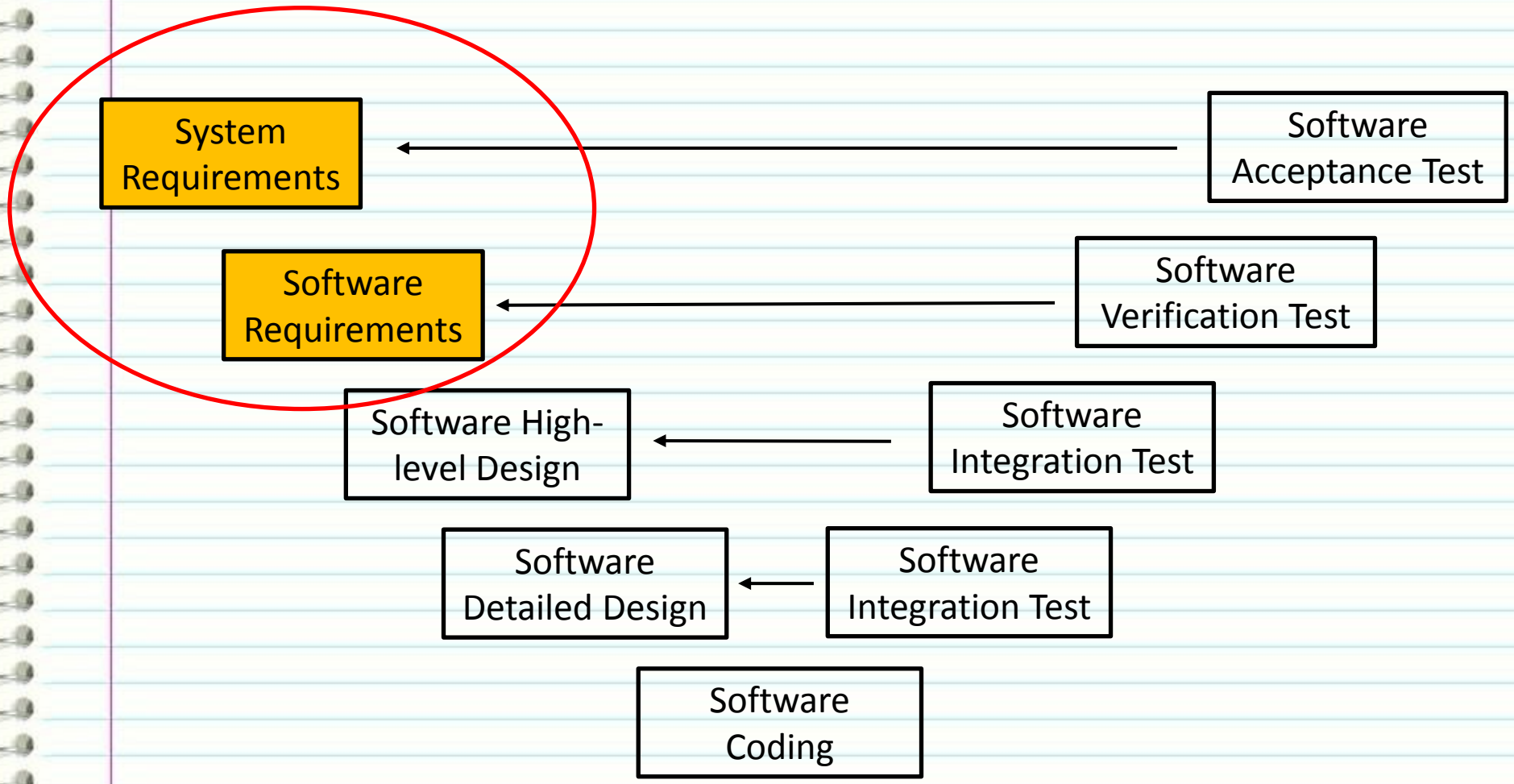
1. What are typical software estimation measures and how do they apply to software requirements?

What is Requirements Engineering?

- Challenge facing system and software engineers - “How can we ensure that we have specified a system that:
  - Properly meets the customer’s needs
  - Satisfies the customer’s expectations

- Requirements engineering provides mechanisms for:
  - Understanding what the customer wants
  - analysing need
  - assessing feasibility
  - negotiating a reasonable solution
  - specifying the solution
  - validating the specification
  - managing the transformation of the requirements into an operational system

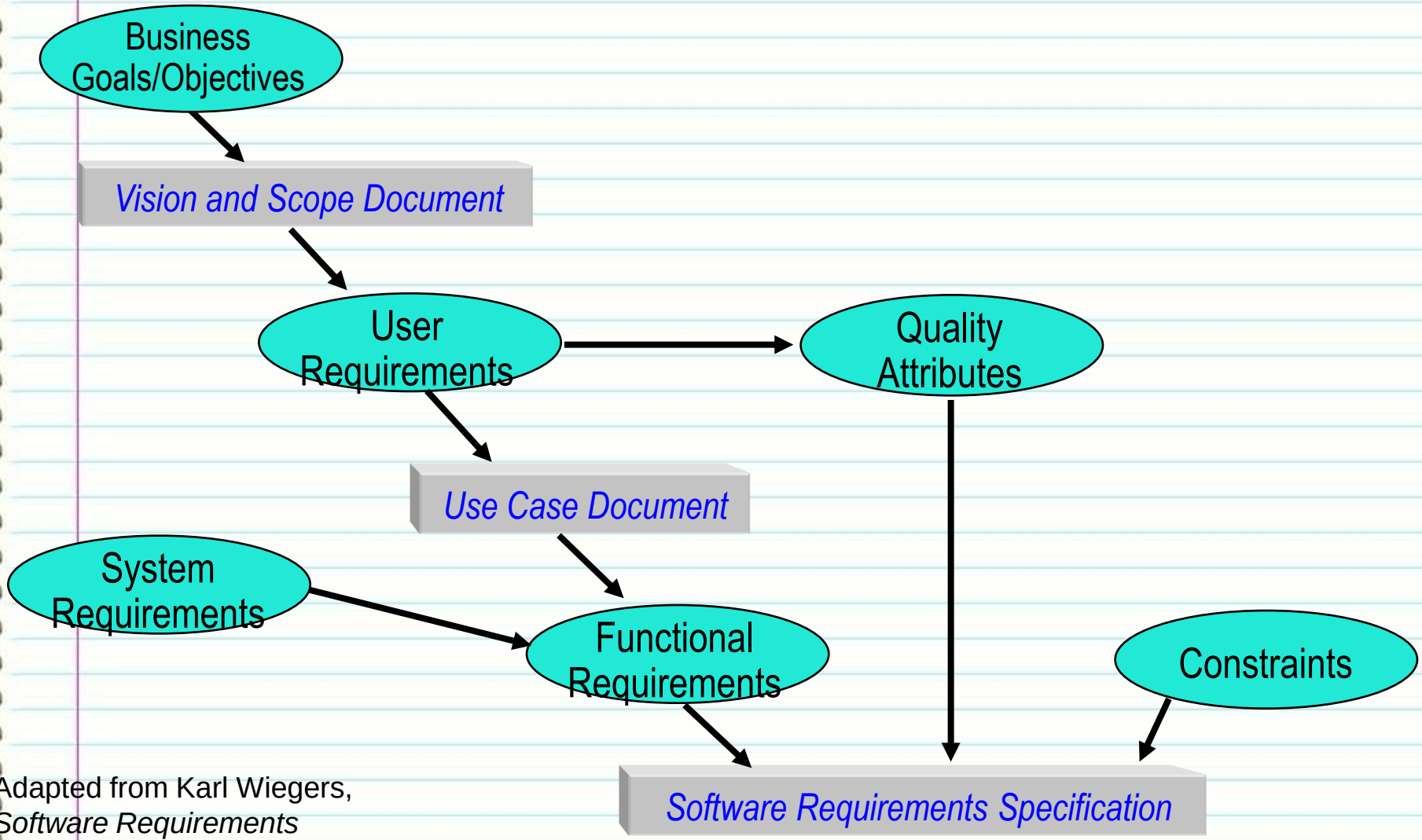
Requirements in the Life Cycle



Requirements Elicitation Activities

- Identifying problems and needs
- Constructing analysis models to help understanding
- Formulating system/software requirements
- Conducting feasibility study
- Checking the requirements and models for desired properties such as correctness, and consistency
- Specifying acceptance tests
- Formulating an iterative development plan

Software Requirements Specification



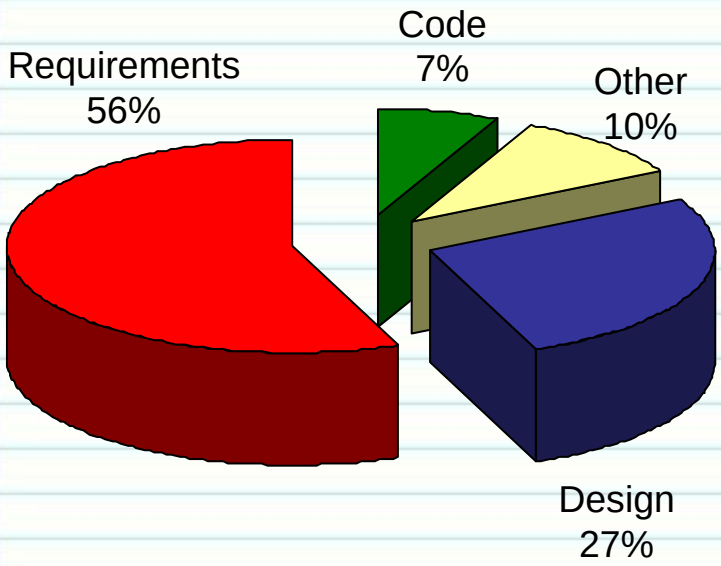
Adapted from Karl Wiegers, *Software Requirements*

General Problems with the Requirements Process

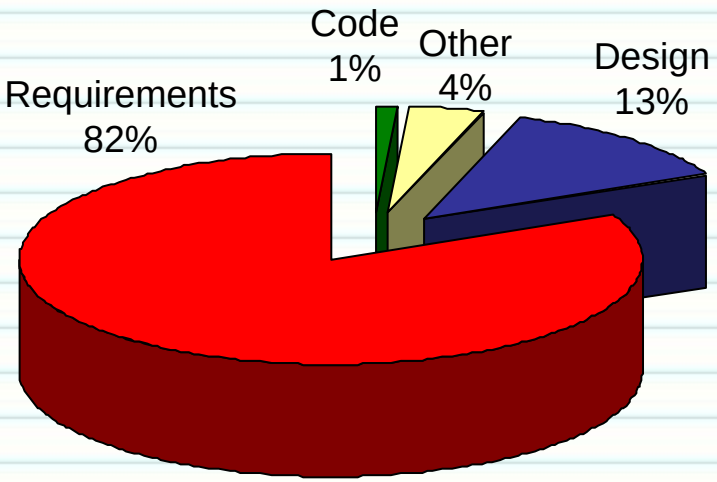
- Lack of the right expertise (software engineers, domain experts, etc.)
- Initial ideas are often incomplete, wildly optimistic, and firmly entrenched in the minds of the people leading the acquisition process
- Difficulty of using complex tools and diverse methods associated with requirements gathering may negate the anticipated benefits of a complete and detailed approach

Why Do We Manage Requirements ?

Distribution of Defects



Distribution of Effort to Fix Defects



(Martin & Leffinwell)

Requirements Elicitation

The book concentrates mostly on Requirements Elicitation

- Requirements are capabilities (stated as part of a contract) that the system must deliver.
- Requirements are documented in a requirements specification, which serves as part of the contract.
- Requirements elicitation is the process to identify and formulate the capabilities for the software system.
  - Identifying problems and needs
  - Constructing analysis models to help understanding
  - Formulating system/software requirements
  - Conducting feasibility study
  - Checking the requirements and models for desired properties such as correctness, and consistency
  - Specifying acceptance tests
  - Formulating an iterative development plan

Elicitation techniques

- interview
- Delphi technique
- brainstorming session
- task analysis
- scenario analysis
- ethnography
- form analysis
- analysis of natural language descriptions
- synthesis from existing system
- domain analysis
- Business Process Redesign (BPR)
- prototyping

Typical Elicitation Mistakes

- **Noise**
  - the presence of text that carries no relevant information to any feature of the problem.
- **Silence**
  - a feature that is not covered by any text.
- **Over-specification**
  - text that describes a feature of the solution, rather than the problem.
- **Contradiction**
  - text that defines a single feature in a number of incompatible ways.
- **Ambiguity**
  - text that can be interpreted in at least two different ways.
- **Forward reference**
  - text that refers to a feature yet to be defined.
- **Wishful thinking**
  - text that defines a feature that cannot possibly be validated.
- **Jigsaw puzzles**
  - e.g. distributing requirements across a document and then cross-referencing
- **Inconsistent terminology**
  - Inventing and then changing terminology
- **Putting the onus on the development staff**
  - i.e. making the reader work hard to decipher the intent
- **Writing for the hostile reader**
  - There are fewer of these than friendly readers

Source: Steve Easterbrook, U. of Toronto

Requirements Problems Can Be Disastrous!

- **Insufficient requirements specification and their ad hoc management**
- **Ambiguous and imprecise communication**
- Brittle architecture
- Overwhelming complexity
- **Undetected inconsistencies in requirements, design, and implementation**
- Poor and insufficient testing
- Subjective assessment of project status
- Failure to attack risk
- Uncontrolled change propagation
- Insufficient automation

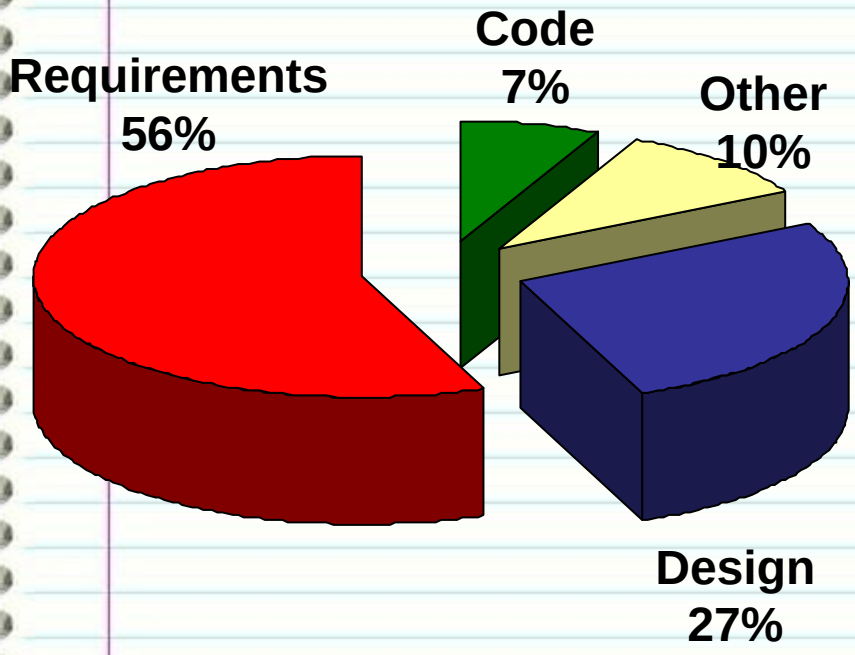
Statistics from NIST Report

- NIST (National Institute of Standards and Technology) has published a comprehensive (309 pages) and very interesting report on project statistics and experiences based on data from a large number of software projects[1]
  - 70% of the defects are introduced in the specification phase
  - 30% are introduced later in the technical solution process
  - Only 5% of the specification inadequacies are corrected in the specification phase
  - 95% are detected later in the project or after delivery where the cost for correction on average is 22 times higher compared to a correction directly during the specification effort
  - The NIST report concludes that extensive testing is essential, however testing detects the dominating specification errors late in the process

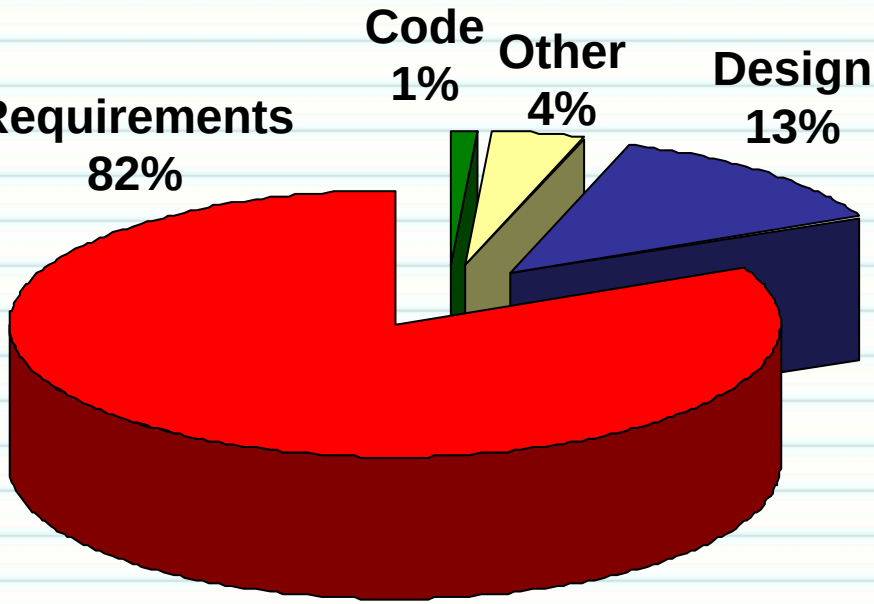
[1] http://www.nist.gov/public_affairs/releases/n02-10.htm (May 2002)

Why Focus on Requirements ?

- Distribution of Defects



- Distribution of Effort to Fix Defects



Source: Martin & Leffinwell

Back To Requirements Specification

Challenges of Requirements Elicitation

"The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing that detailed technical requirements, including all the interfaces to people, to machines, and to other software systems. No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later."

Frederick Brooks 1987

Challenges of Requirements Elicitation

As an analyst, I need to know what do you want?

I want you to design the software for me.

But what do you want to do with the software?

I don't know until you tell me what the software can do.

Well, I can design the software to do anything!

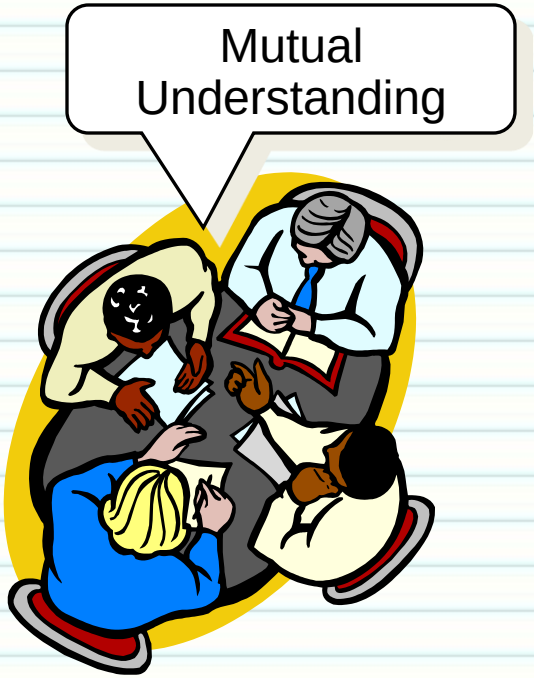
Can you design the software to tell you my requirements?!

Communication Barrier - Misunderstanding

What is a Requirement?

- Statement of some THING you want or need
  OR
  A characteristic of some THING you want or need

- A requirement is also…
  - A *Contractually Binding* Statement
  - Documentation of *Problem Space*
  - The *Means* We Use to Communicate



Types of Requirement

- Functional requirements – statements of information processing capabilities that the software system must possess.
- Nonfunctional requirements include
  - Performance requirements
  - Quality requirements
  - Safety requirements
  - Security requirements
  - Interface requirements

Examples of Functional Requirements

- For a car rental system:
  - The system *shall* allow a potential customer to inquire information and availability of rental cars using various combinations of search criteria including make, model, from date, to date, price range, and class (small size, medium size, large size, and luxury cars).
  - This is a better requirement than many - but several things are missing - what are they?
- For a study abroad system:
  - The system *shall* provide interactive as well as batch-processing means for an OIE (Office of International Education) staff to enter the exchange programs into the database.
  - This is a better requirement than many - but several things are missing - what are they?

Examples of Non-Functional Requirements

- Workload:
  - The system shall be capable of handling a typical workload of 10,000 (ten thousand) inquiries at the same time.
- Response time:
  - The system's response time shall not exceed 3 (three) seconds under the typical workload.
- Development Methodology (sometimes specified by the customer)
  - The System/software shall use the Object Oriented methodology
  - The Software shall be coded in Java

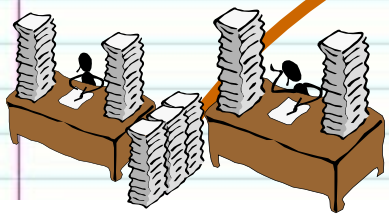
Examples of Quality Requirements

- Security:
  - The system must protect the contents of the website from malicious attacks and protect the privacy of its users.
- Platform independence:
  - The system must run on Windows 2000 and later, Unix, Linux, Mac and support popular relational database management systems including Oracle, SQL Server, Access, and MySQL.

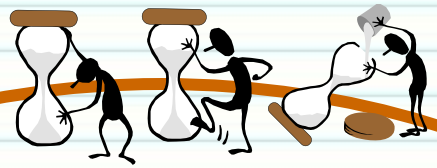
Examples of Quality/Interface Requirements

- User friendliness:
  - The system must provide a user friendly interface that conforms to commonly used web-application user interface look-and-feel and man-machine interaction conventions.
- User interface:
  - The system must support the following user interfaces:
    - web-based
    - stand-alone
    - voice
    - cellular phone

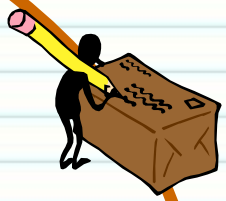
Requirements Elicitation Steps



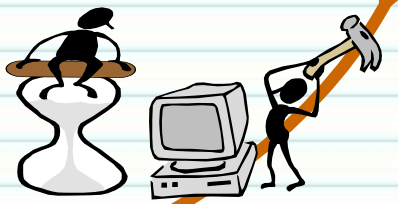
1. Collect information about the application.



2. Construct analysis models, if desired.



3. Derive requirements & constraints.



4. Conduct feasibility study.

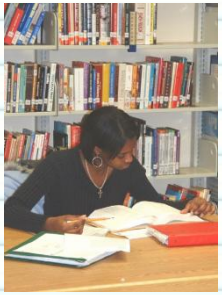
5. Review the requirements specification.

Information Collection Techniques

Customer presentation

business forms

operating procedures



regulations & standards

Literature survey

Stakeholder survey

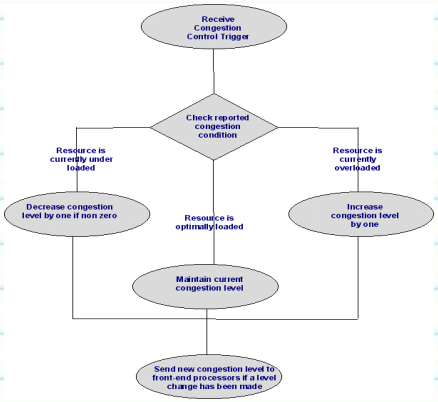
User interviewing

Writing user stories

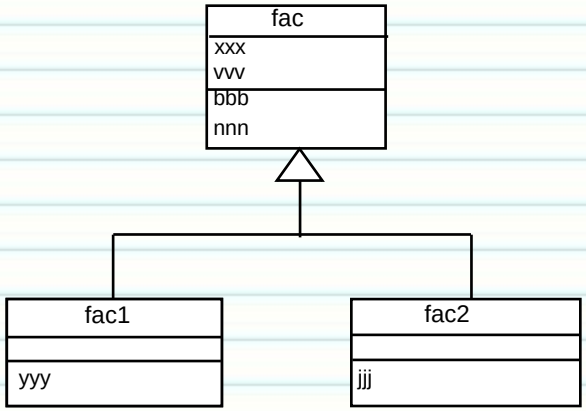
Focuses of Information Collection Activities

- What is the business, the current business situation, and how does it operate?
- What is the system's environment or context?
- What are existing business processes, their input and output, and how do they relate to each other?
- What are the problems with the current system?
- What are the business or product goals?
- Who are the users of the current and future systems, respectively?
- What do the customer and users want, and what are their business priorities?
- What are the quality, performance, and security considerations?

Constructing Analysis Models



Intuitive models



Purpose: to aid understanding of the application, requirements, and constraints.

Formal or informal models

Businesses in Different Domains

Finance

Insurance

Health Care

Transportation

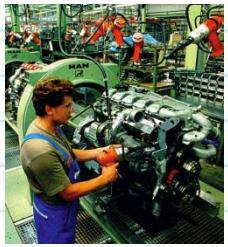
Defense

Telecommunication

Retailing

Energy

government



Manufacturing

Education

and more

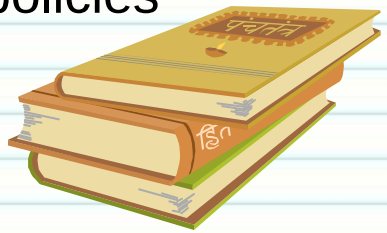
Deriving Requirements and Constraints

Current Situation

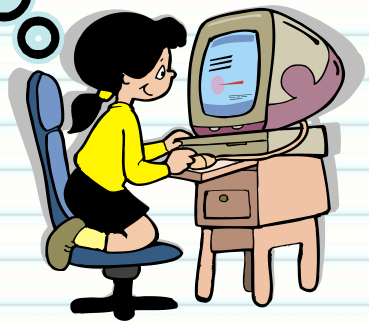
Business Goals

Discrepancies/ Problems

Industry standards and government policies



Needs & wish list



Requirements & constraints

Requirements Specification Details

Requirements Specification

1. Introduction to Document
   - 1.1 Purpose of Product
   - 1.2 Scope of Product
   - 1.3 Acronyms, Abbreviations, Definitions
   - 1.4 References
   - 1.5 Outline of the Rest of the SRS
2. General Description of Product
   - 2.1 Context of Product
   - 2.2 Product Function
   - 2.3 User Characteristics
   - 2.4 Constraints
   - 2.5 Assumptions and Dependencies
3. Specific Requirements
   - 3.1 External Interface Requirements
     - 3.1.1 User Interfaces
     - 3.1.2 Hardware Interfaces
     - 3.1.3 Software Interfaces
     - 3.1.4 Communication Interfaces
   - 3.2 Functional Requirements
     - 3.2.1 Class 1
     - 3.2.2 Class 2
     - 3.2.3 ...
   - 3.3 Performance Requirements
   - 3.4 Design Constraints
   - 3.5 Quality Requirements
   - 3.6 Other Requirements
4. Appendices

IEEE SRS Standard by Objects, 1998

Feasibility Study

- Not all projects are practically doable with technology, time, and resource constraints.
- Feasibility study aims at determining if the project is doable under the given constraints.
- Feasibility study in RE is concerned with
  - the feasibility of the functional, performance, nonfunctional, and quality constraints
  - adequacy of the technology
  - timing and cost constraints
  - constraints imposed by the customer, industry and government agencies

Requirements Verification and Validation

- Verification: Are we building the product right?
  - Are we building the product in the right way?
  - It concerns the product building process.
- Validation: Are we building the right product?
  - Are we building the product that the customer wants?
  - It concerns the correctness of the product being built.

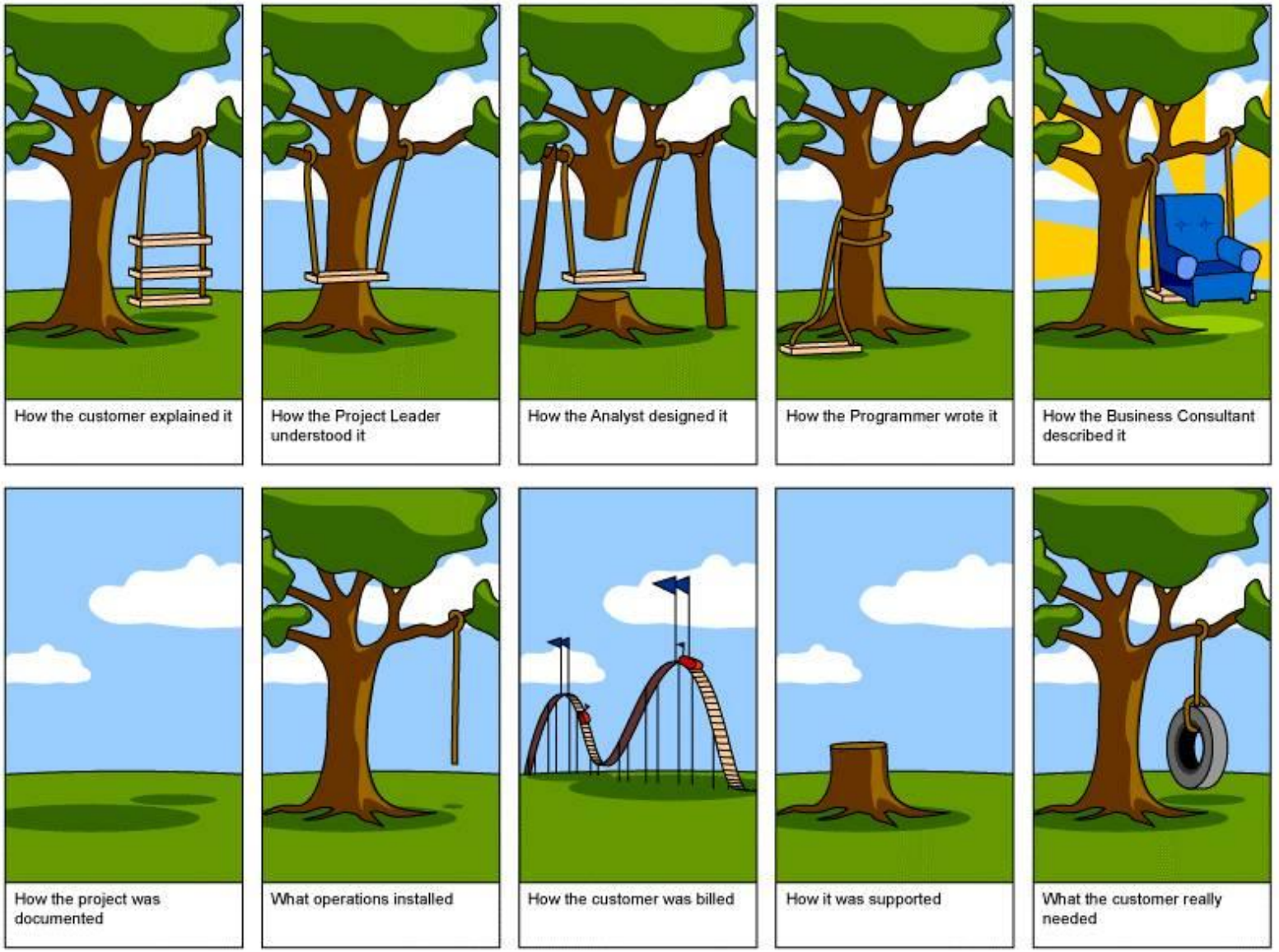
Three Types of Requirements Review

- Technical review is an internal review performed by the technical team. Techniques include:
  - **peer review -** peers perform informal "desktop reviews" sometimes guided by a review questionnaire
  - **walkthrough -** the analyst explains each requirement while the reviewers examine it and raise doubts
  - **inspection** - inspector is guided by a checklist of commonly encountered problems in SRS (e.g., incompleteness, duplicate definition, inconsistency, etc.)
- In every case there is a requirement for period of time between which the review materials are available and the review is held – to allow for the pre-review (this is where most errors are found)

Three Types of Requirements Review

- Additional reviews (these are validation activities)
  - Expert review means review of the requirements specification by domain experts.
  - Customer/user reviews are performed by involving the customer and/or users of the system.

What is the right system to build ?



Quality Requirement Characteristics

- Correct
- Feasible
- Necessary
- Prioritized
- Unambiguous
- Verifiable

Correct Requirement

- Accurately describes functionality to be delivered
- Reference for correctness – source of requirement
  - Actual customer
  - Higher-level system requirements specification
- Need user representatives to determine correctness of user requirements
  - Inspection of requirements
  - Prevent developers from “guessing”

Feasible Requirement

- Implement within known capabilities and limitations of system environment
- Developer should work with requirements analysts or marketing personnel throughout elicitation
  - Provides reality check on technical feasibility of requirement
    - What can and cannot be done
    - What can be done only at excessive cost
    - What can be done only with other tradeoffs
- Example: *"The product shall switch between displaying and hiding non-printing characters instantaneously."*
  - Not feasible – computers cannot do anything instantaneously

Necessary Requirement

- Something the customer really needs
- Something required for conformance to external requirement, external interface, or standard
- Should be traceable back to its source
  - Source should be someone authorized to specify requirements
  - Untraceable requirements may not really be necessary

- Estimates are that over 30% of most software requirements are capabilities that the user never asked for
  - "Gold plating"

Prioritized Requirement

- Assigns implementation priority for particular product release
  - Value provided to customer
  - Relative cost of implementation
  - Relative technical risk associated with implementation
- Assigned by customers or their surrogates
- Gives project manager more control
  - New requirements added during development
  - Budget cuts
  - Schedule overruns
  - Team member departure
- Requirement priority levels example
  - High priority – must be in next product release
  - Medium priority – necessary, but can be deferred to later release
  - Low priority – nice to have, but may be dropped if insufficient time or resources

Unambiguous Requirement

- Reader should be able to draw only one interpretation
- Multiple readers should arrive at same interpretation
- Should be void of subjective words (*easy*, *simple*, *rapid*, *efficient*, etc.)
- Should be written in simple, straightforward language of user domain
- Reveal ambiguity
  - Formal inspections of requirements specification
  - Write test cases from requirements
  - Create user scenarios showing expected behavior
- Example: *"The HTML Parser shall produce an HTML markup error report which allows quick resolution of errors when used by HTML novices."*
  - Ambiguous – the word *"quick"*

Complete SRS

- No requirements or necessary information should be missing
- Hard to spot missing requirements
- Ways to achieve completeness
  - Use case method – focus on user tasks rather than system functions during elicitation
  - Graphical analysis models – represent different views of requirements
- Flagging missing requirements information
  - TBD ("to be determined") – resolved before product development
- Example: *"The product shall provide status messages at regular intervals not less than every 60 seconds."*
  - Incomplete – what are the status messages? how are they supposed to be displayed to the user?

Consistent SRS

- Different requirements should not conflict
- Disagreements among requirements must be resolved before development can proceed
- Ensure requirements modification does not introduce inconsistencies

Traceable SRS

- Link each requirement to its source
  - Higher-level system requirement
  - Use case
  - Voice-of-the-customer statement
- Link each requirement to its development
  - Design elements
  - Source code
  - Test cases
- Uniquely label each requirement
- Express each requirement in a structured, fine-grained way

Quality Requirements Guidelines

- Keep sentences and paragraphs short
- Use active voice
- Use proper grammar, spelling, and punctuation
- Use terms consistently and define in glossary or data dictionary
- Read requirements from developer's perspective (to check if well-defined)
- Use right level of granularity
  - Avoid long narrative paragraphs containing multiple requirements
  - Write individually testable requirements
- Use consistent level of detail throughout document
- Avoid stating requirements redundantly
  - Makes maintenance of document easier
  - Less vulnerable to inconsistencies

Test Driven Requirements

- This is cited as one of the best techniques for Agile, but many teams are not using it
- This is done before a single line of code is written – it helps to reduce requirements rework that impacts the design and code
- The technique works as follows:

1. Review the requirements for that iteration – look for initial areas of requirements specification problems.
2. Get feedback on the problems discovered – update requirements as necessary
3. Develop test cases from the requirements. The test case design will uncover other problems.
4. Get feedback on the problems discovered – update requirements as necessary

Module Summary: Requirements The Basics

- Requirements define the problem to be solved and establish the terms by which mission success will be measured.
- Requirements problems are the single biggest problem on development projects so care in creating good requirements always pays off.
- The later a problem is discovered the more costly it is to recover from.
- Requirements are distributed within the system architecture via flow-down, allocation and derivation.
- Requirements traceability is a technique of tracking the source and connections between requirements. It is used to assess the consequences of potential requirements changes.
- When a system is decomposed into smaller segments, interfaces are created that must be defined and managed.