

# Requirements Engineering

---

Instructor: Mehroze Khan


# Requirements Prioritization

- Different stakeholders have different set of requirements
  - potentially conflicting ideas
- Need to prioritize requirements to resolve conflicts
- Prioritization might separate requirements into three categories
  - ***essential***: absolutely must be met
  - ***desirable***: highly desirable but not necessary
  - ***optional***: possible but could be eliminated



# Characteristics of Requirements

The requirements must be:

- Correct
  - Consistent
  - Unambiguous
  - Complete
  - Feasible
  - Relevant
  - Testable
  - Traceable
- 

# Functional and Non-functional Requirements

- Software system requirements are often classified as functional or non-functional requirements:
- **Functional requirement**
  - Describes: interaction between the system and its environment, how should the system behave given certain stimuli, required behaviour in terms of required activities
  - Example: For a system of printing pay cheques, the functional requirements must answer the following:
    - When are pay cheques issued?
    - What input is necessary for a pay cheque to be printed?
- **Quality requirement or non-functional requirement**
  - Describes: some quality characteristic that the software must possess, a restriction on the system that limits our choices for constructing a solution
    - Constraints could be:
      - **Design constraint:** a design decision such as choice of platform or interface components
      - **Process constraint:** a restriction on the techniques or resources that can be used to build the system
  - Example: queries to the system must be answered within 3 seconds

# Functional and Non-functional Requirements

- To put it simply, functional requirements describe **what the product should do**, while non-functional requirements place constraints on **how the product should do it**.
- They can be expressed in the following form:
  - Functional requirement:** "The system must do [requirement]."
  - Non-functional requirement:** "The system shall be [requirement]."
- **Example:**
  - Functional requirement:** "The system must allow the user to submit feedback through a contact form in the app."
  - Non-functional requirement:** "When the submit button is pressed, the confirmation screen shall load within 2 seconds."

# Functional Requirements

## Functional Requirements

### Functionality

- What will the system do?
- When will the system do it?
- Are there several modes of operation?
- What kinds of computations or data transformations must be performed?
- What are the appropriate reactions to possible stimuli?

### Data

- For both input and output, what should be the format of the data?
- Must any data be retained for any period of time?

# Example

- **User story:** As an existing user, I want to be able to log into my account.
- **Functional requirement:**
  - Log In
    - The system must allow users to log into their account by entering their email and password.
    - The system must allow users to log in with their Google accounts.
    - The system must allow users to reset their password by clicking on "I forgot my password" and receiving a link to their verified email address.

# Non-functional Requirements

- Non-functional or Quality requirements, as the name suggests, are requirements that are not directly concerned with the specific services delivered by the system to its users.
- These non-functional requirements usually specify or constrain characteristics of the system as a whole.
- A *nonfunctional requirement* (NFR) can be described as a quality attribute, a performance attribute, a security attribute, or a general constraint on a system.



# Example

- Security
  - Users shall be forced to change their password the next time they log in if they have not changed it within the length of time established as “password expiration duration”.
  - Passwords shall never be visible at the point of entry or at any other time.
  - Any new login shall be allowed after two step verification.

# Non-functional Requirements

## Quality Requirements

### Performance

- Are there constraints on execution speed, response time, or throughput?
- What efficiency measures will apply to resource usage and response time?
- How much data will flow through the system?
- How often will data be received or sent?

### Usability and Human Factors

- What kind of training will be required for each type of user?
- How easy should it be for a user to understand and use the system?
- How difficult should it be for a user to misuse the system?

### Security

- Must access to the system or information be controlled?
- Should each user's data be isolated from the data of other users?
- Should user programs be isolated from other programs and from the operating system?
- Should precautions be taken against theft or vandalism?

## Reliability and Availability

- Must the system detect and isolate faults?
- What is the prescribed mean time between failures?
- Is there a maximum time allowed for restarting the system after a failure?
- How often will the system be backed up?
- Must backup copies be stored at a different location?
- Should precautions be taken against fire or water damage?

## Maintainability

- Will maintenance merely correct errors, or will it also include improving the system?
- When and in what ways might the system be changed in the future?
- How easy should it be to add features to the system?
- How easy should it be to port the system from one platform (computer, operating system) to another?

## Precision and Accuracy

- How accurate must data calculations be?
- To what degree of precision must calculations be made?

## Time to Delivery / Cost

- Is there a prescribed timetable for development?
- Is there a limit on the amount of money to be spent on development or on hardware or software?

# Testable Requirements

- Testable/Measurable Requirement:
  - A requirement which is unambiguous and clearly specifies the behaviour.
  - Objective description of the requirement's meanings.
  - All possible entities and activities can be examined and classified as Meet Requirements and Do Not Meet Requirements.
- Testable requirements are helpful in making good design.
- Requirements that are not testable are likely to be ambiguous, incomplete and incorrect.

# Testable Requirements

- 3 ways to help make requirements testable:
  - Specify a quantitative description for each adverb and adjective
  - Replace pronouns with specific names of entities
  - Make sure that every noun is defined in exactly one place in the requirements document

# Testable/Non-Testable Requirements

- Some examples:
  - **Not Testable:** Water quality information must be accessible immediately
  - **Testable:** Water quality information must be retrieved within five seconds of request
  - **Not Testable:** The system should handle a large number of users at a time
  - **Testable:** The system should handle 5000 users at a time
  - **Not Testable:** User should press the Save button when writing text in the system. This prevents it from being lost.
  - **Testable:** User should press the Save button when writing a note in the system. Pressing the Save button prevents the text from being lost.

# Requirements Documentation

- No matter what method we choose for defining requirements, we must keep a set of documents recording the result.
- We and our customers will refer to these documents throughout development and maintenance.
- Clear and precise illustrations and diagrams accompanying the documentation should be consistent with the text.

# Requirements Definition

- The requirements definition is a record of the requirements expressed in the customer's terms. Working with the customer, we document what the customer can expect of the delivered system:
- Outline the general purpose and scope of the system, including relevant benefits, objectives, and goals
- Describe the background and the rationale behind proposal for new system
- Describe the essential characteristics of an acceptable solution
- Describe the environment in which the system will operate
- Outline a description of the proposal, if the customer has a proposal for solving the problem
- List any assumptions we make about how the environment behaves

# Requirements Specification

- The requirements specification covers exactly the same ground as the requirements definition, but from the perspective of the developers.
- Where the requirements definition is written in terms of the customer's vocabulary, referring to objects, states, events, and activities in the customer's world, the requirements specification is written in terms of the system's interface.
- We accomplish this by rewriting the requirements so that they refer only to those real-world objects (states, events, actions) that are sensed or actuated by the proposed system.



# Requirements Specification

- Describe all inputs and outputs in detail, including
  - the sources of inputs
  - the destinations of outputs
  - the value ranges
  - data format of inputs and output data
  - data protocols
  - window formats and organizations
  - timing constraint
- Restate the required functionality in terms of the interfaces' inputs and outputs
- Devise fit criteria for each of the customer's quality requirements

# IEEE Standard for SRS Document

1. Introduction to the Document
  - 1.1 Purpose of the Product
  - 1.2 Scope of the 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 Functions
  - 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 Communications Interfaces
  - 3.2 Functional Requirements
    - 3.2.1 Requirement 1
    - 3.2.2 Requirement 2
    - ...
  - 3.3 Performance Requirements
  - 3.4 Design Constraints
  - 3.5 Other Quality Requirements
  - 3.6 Other Requirements
4. Appendices

# Capturing Requirements

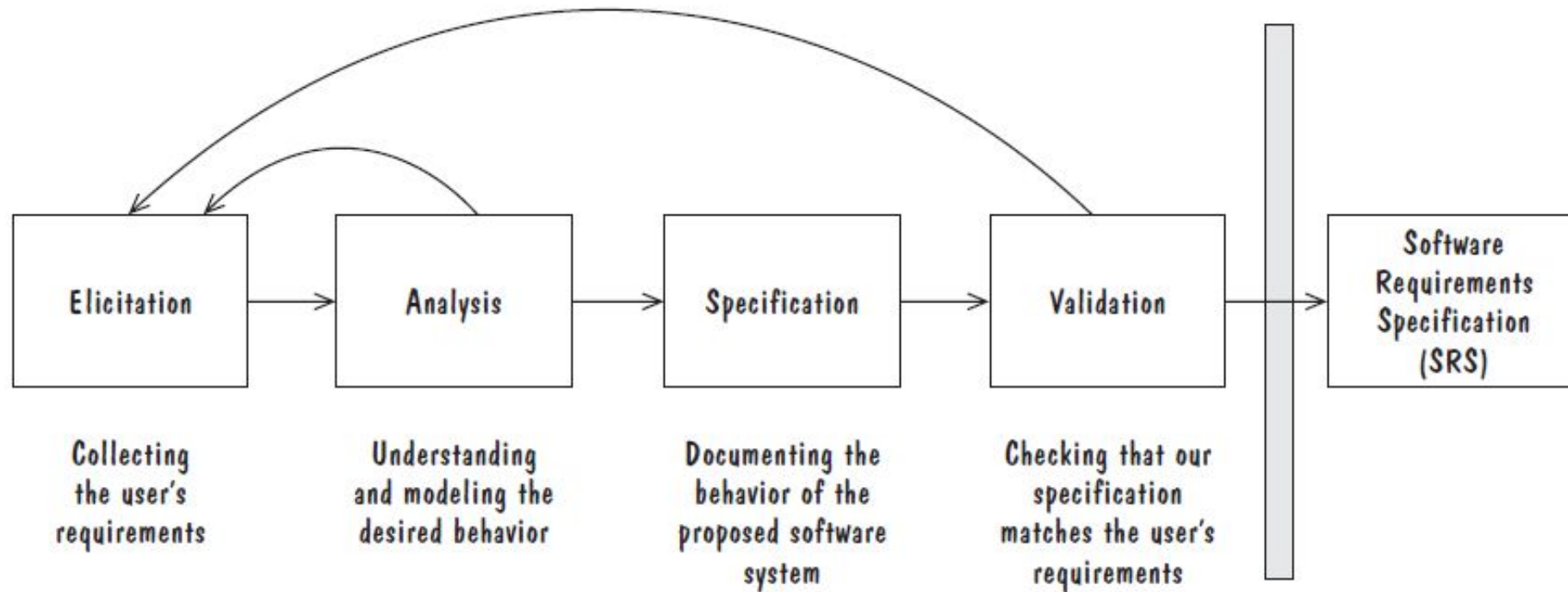





FIGURE 4.1 Process for capturing the requirements.

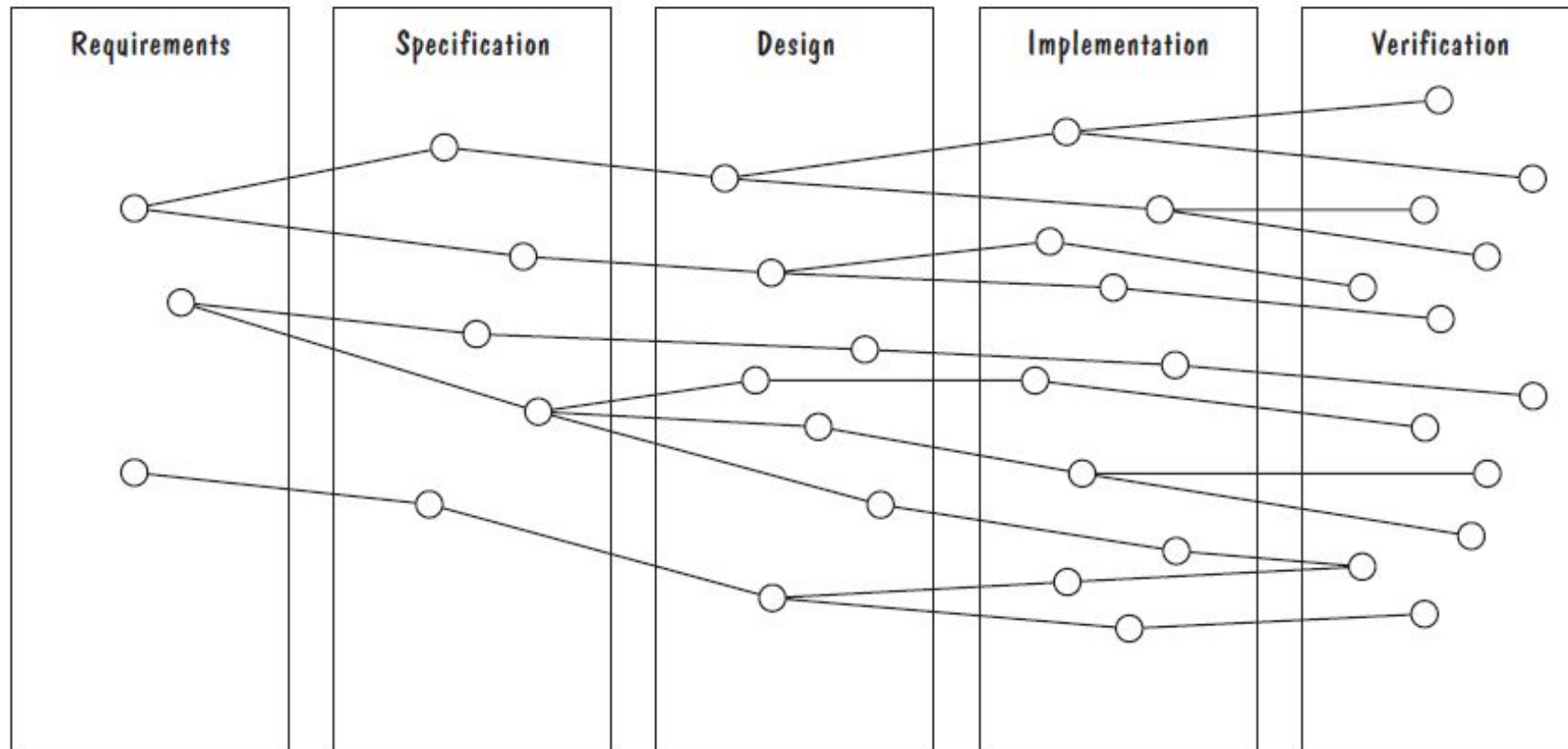


# Process Management and Requirements Traceability

- Process management is a set of procedures that track
    - the requirements that define what the system should do
    - the design modules that are generated from the requirement
    - the program code that implements the design
    - the tests that verify the functionality of the system
    - the documents that describe the system
  - It provides the threads that tie the system parts together
- 


# Process Management and Requirements Traceability

- Horizontal threads show the coordination between development activities



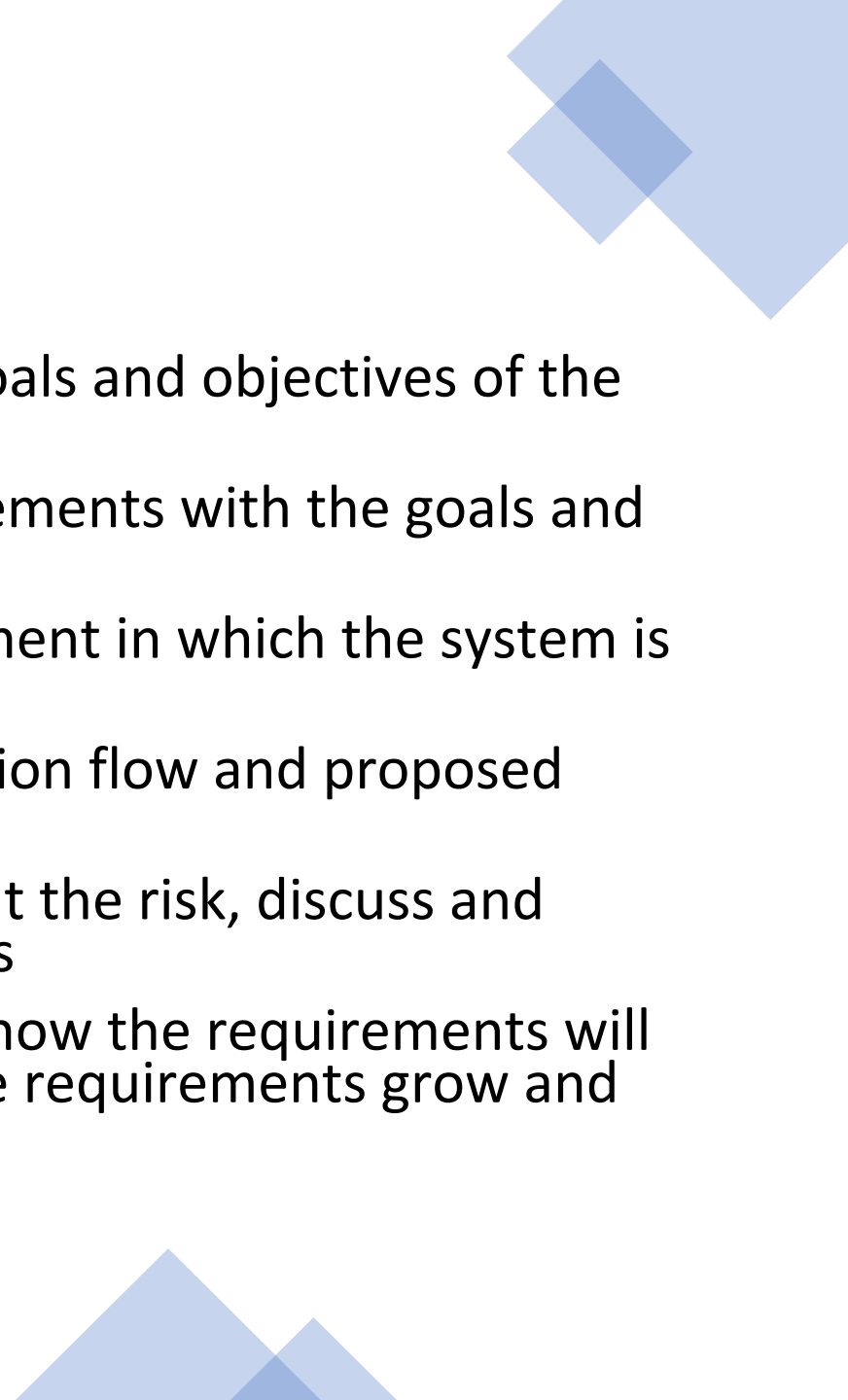


# Validation and Verification

- In **Requirements Validation**, we check that our requirements definition accurately reflects the customer's—actually, all of the stakeholders'—needs.
  - In **Verification**, we check that one document or artifact conforms to another. Thus, we verify that our code conforms to our design, and that our design conforms to our requirements specification; at the requirements level, we verify that our requirements specification conforms to the requirements definition.
  - To summarize, **verification ensures that we *build the system right***, whereas **validation ensures that we *build the right system!***
- 



# Requirements Review

- Review the stated goals and objectives of the system
  - Compare the requirements with the goals and objectives
  - Review the environment in which the system is to operate
  - Review the information flow and proposed functions
  - Assess and document the risk, discuss and compare alternatives
  - Testing the system: how the requirements will be revalidated as the requirements grow and change
- 

# Measuring Requirements

- Measurements focus on three areas
  - product
  - process
  - resources
- Number of requirements can give us a sense of the size of the developed system
- Number of changes to requirements
  - Many changes indicate some instability or uncertainty in our understanding of the system
- Requirement-size and change measurements should be recorded by requirements type



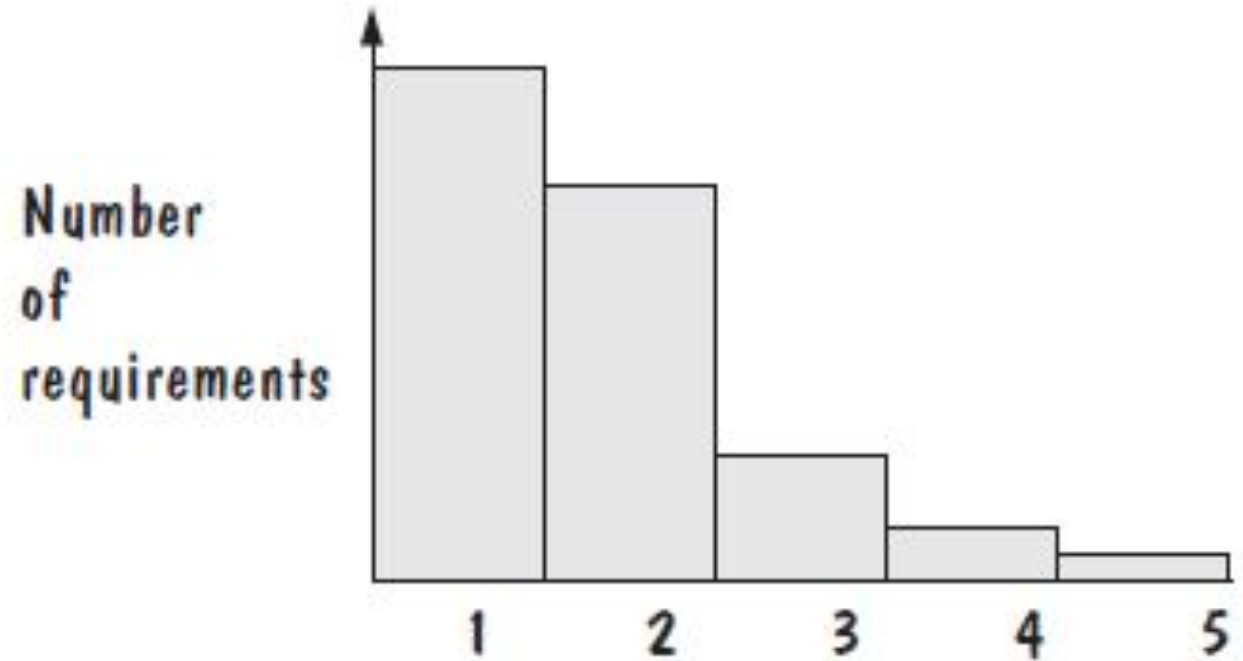
# Measuring Requirements

## Rating Scheme on Scale from 1 to 5

1. You understand this requirement completely, have designed systems from similar requirements, and have no trouble developing a design from this requirement
2. Some elements of this requirement are new, but they are not radically different from requirements that have been successfully designed in the past
3. Some elements of this requirement are very different from requirements in the past, but you understand the requirement and can develop a good design from it
4. You cannot understand some parts of this requirement, and are not sure that you can develop a good design
5. You do not understand this requirement at all, and can not develop a design

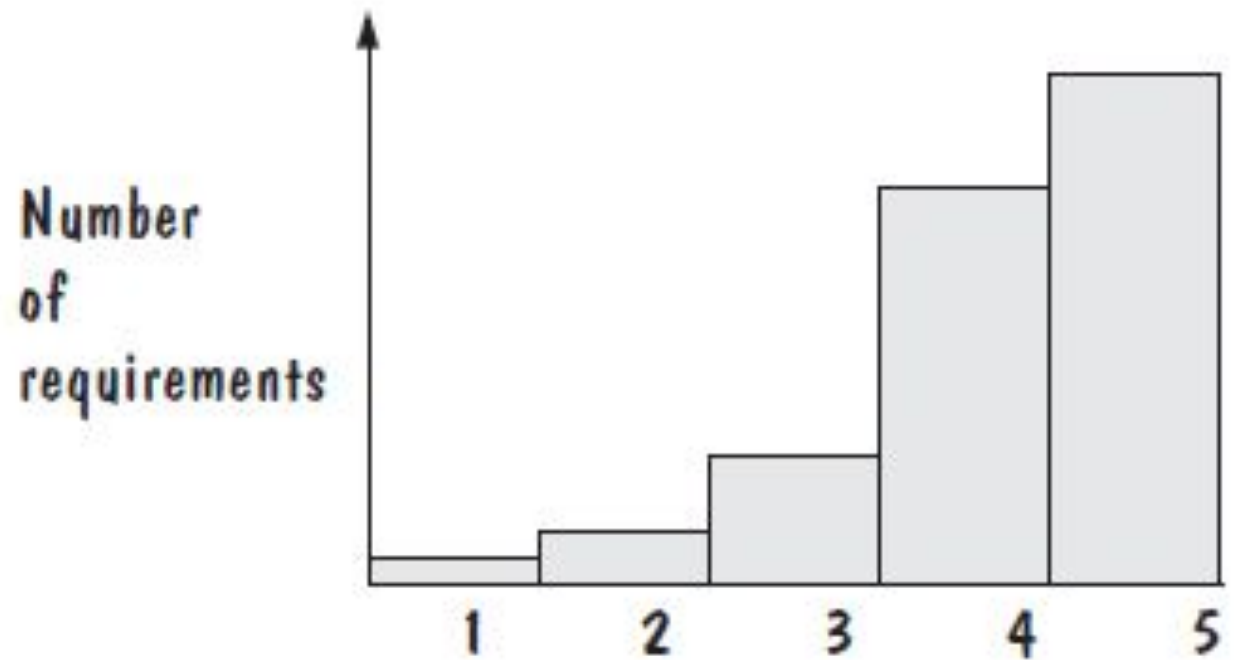
# Measuring Requirements

- If the designers and testers yield profiles with mostly 1s and 2s, as shown in Figure, then the requirements are in good shape and can be passed on to the design team.



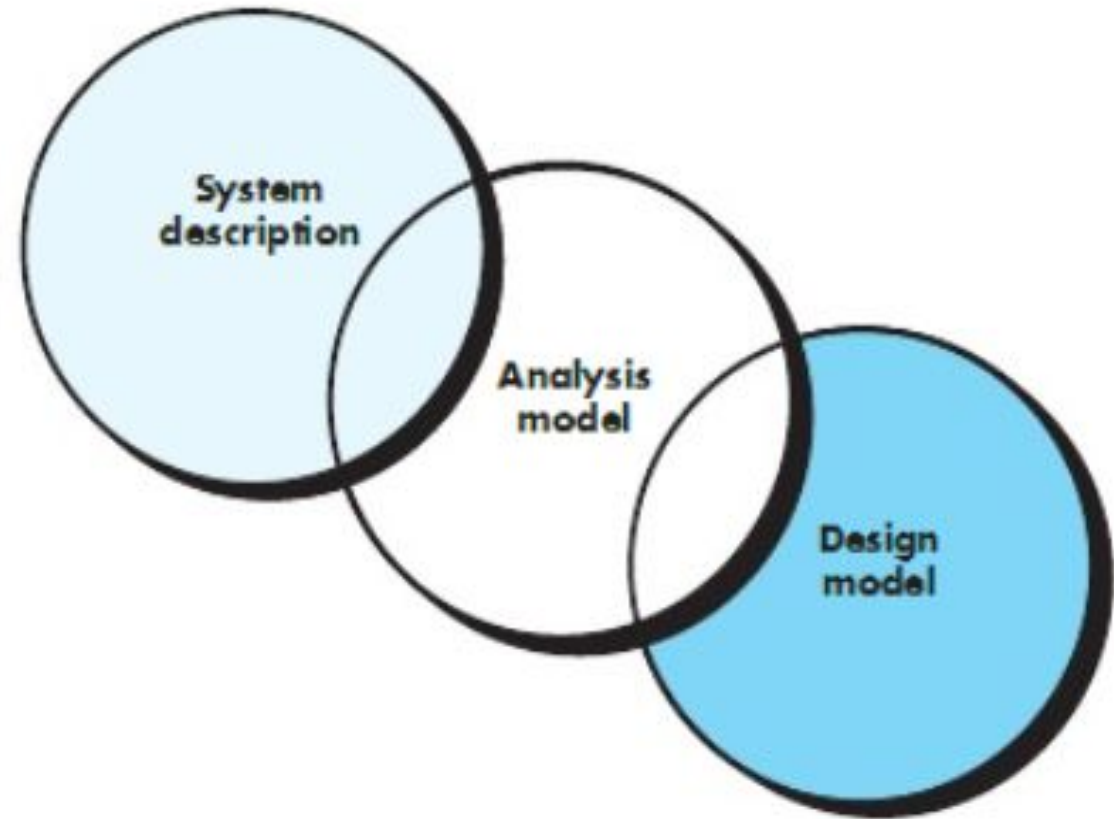
# Measuring Requirements

- If there are many 4s and 5s, then the requirements should be revised, and the revisions reassessed to have better profiles, before we proceed to design.



# Analysis

- Focus on requirements
- Each element should improve understanding of requirements
- Delay consideration of infrastructure till design
- Requirements model provides value to all stakeholders
- Keep the models simple



# References

1. Roger S. Pressman, Software Engineering A Practitioner's Approach, 9<sup>th</sup> Edition. McGrawHill
2. Shari PFleeger, Joanne Atlee, Software Engineering: Theory and Practice, 4<sup>th</sup> Edition
3. Roger S. Pressman, Software Engineering A Practitioner's Approach, 5<sup>th</sup> Edition. McGrawHill