



THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Requirements Engineering

Slide Set - 4

**Organized & Presented By:
Software Engineering Team CSED
TIET, Patiala**

Software Requirements

Software Requirements

Descriptions and specifications of a system

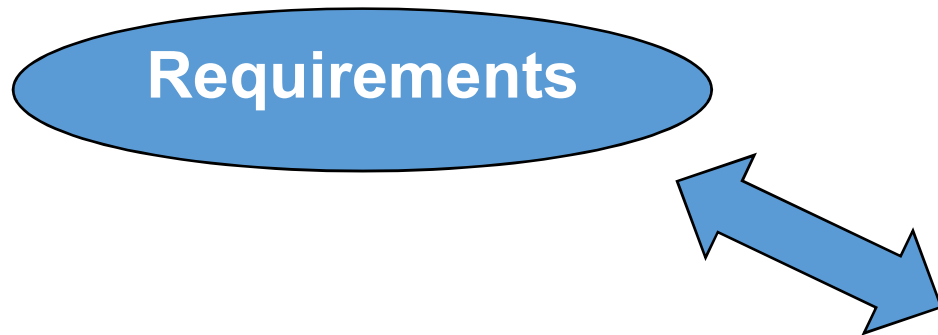
Objectives:

- To introduce the concepts of **user and system requirements**
- To describe **functional / non-functional requirements**
- To explain **two techniques** for describing system requirements
- To explain **how software requirements may be organised** in a requirements document

Requirements engineering

Requirements engineering is the process of establishing

- the services that the customer requires from a system
- the constraints under which it operates and is developed



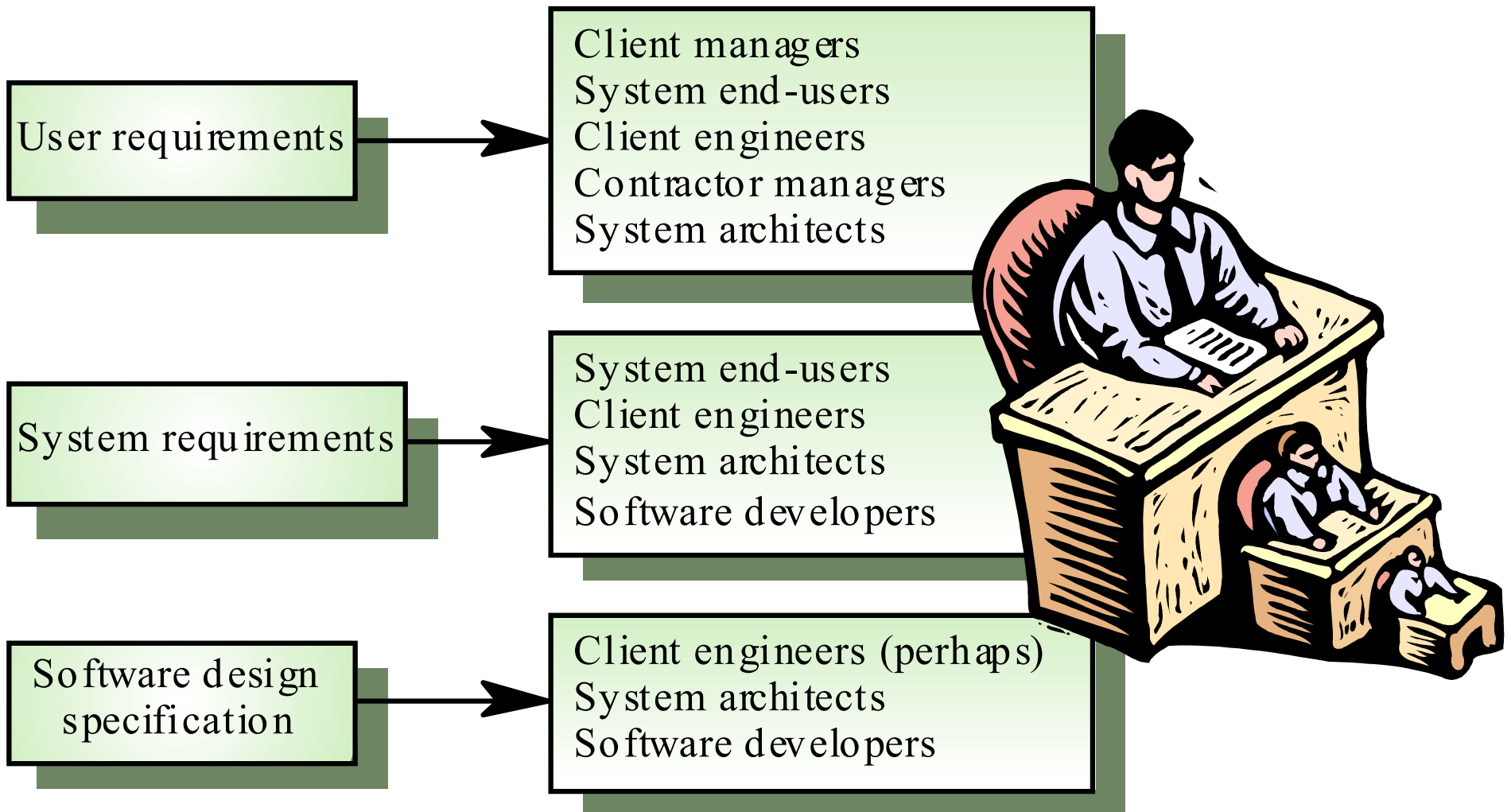
The descriptions of the system services and constraints

that are generated during the requirements engineering process

What is a requirement?

- It may range from a **high-level abstract statement** of a service or of a system constraint to a **detailed mathematical functional specification**
- **Types of Requirements:**
 - **User requirements**
 - Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers
 - **System requirements**
 - A structured document setting out detailed descriptions of the system services. Written as a contract between client and contractor
 - **Software specification**
 - A detailed software description which can serve as a basis for a design or implementation. Written for developers

Requirements readers



Functional and non-functional requirements

- **Functional requirements**

- Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.

- **Non-functional requirements**

- constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.

- **Domain requirements**

- Requirements that come from the application domain of the system and that reflect characteristics of that domain

Functional Requirements

Describe functionality or system services

- **Depend on the type of software**, expected users and the type of system where the software is used
- **Functional user requirements may be high-level statements of what the system should do**

Examples of functional requirements

- The user shall be able to search either all of the initial set of databases or select a subset from it.
- The system shall provide appropriate viewers for the user to read documents in the document store.
- Every order shall be allocated a unique identifier (ORDER_ID) which the user shall be able to copy to the account's permanent storage area.

Requirements imprecision

- Problems arise when **requirements are not precisely stated**
- **Ambiguous requirements** may be interpreted in different ways by developers and users

Requirements completeness and consistency

- In principle, requirements should be both complete and consistent

Complete

- They should include descriptions of all facilities required

Consistent

- There should be no conflicts or contradictions in the descriptions of the system facilities
- In practice, it is very difficult or impossible to produce a complete and consistent requirements document

Non-functional requirements

Define system properties and constraints e.g. reliability, response time and storage requirements.

Constraints are I/O device capability, system representations, etc.

- **Process requirements** may also be specified mandating a particular CASE system, programming language or development method
- **Non-functional requirements** may be more critical than functional requirements. If these are not met, the system is useless

Non-functional classifications

- **Product requirements**

- Requirements which specify that the **delivered product must behave in a particular way** e.g. execution speed, reliability, etc.

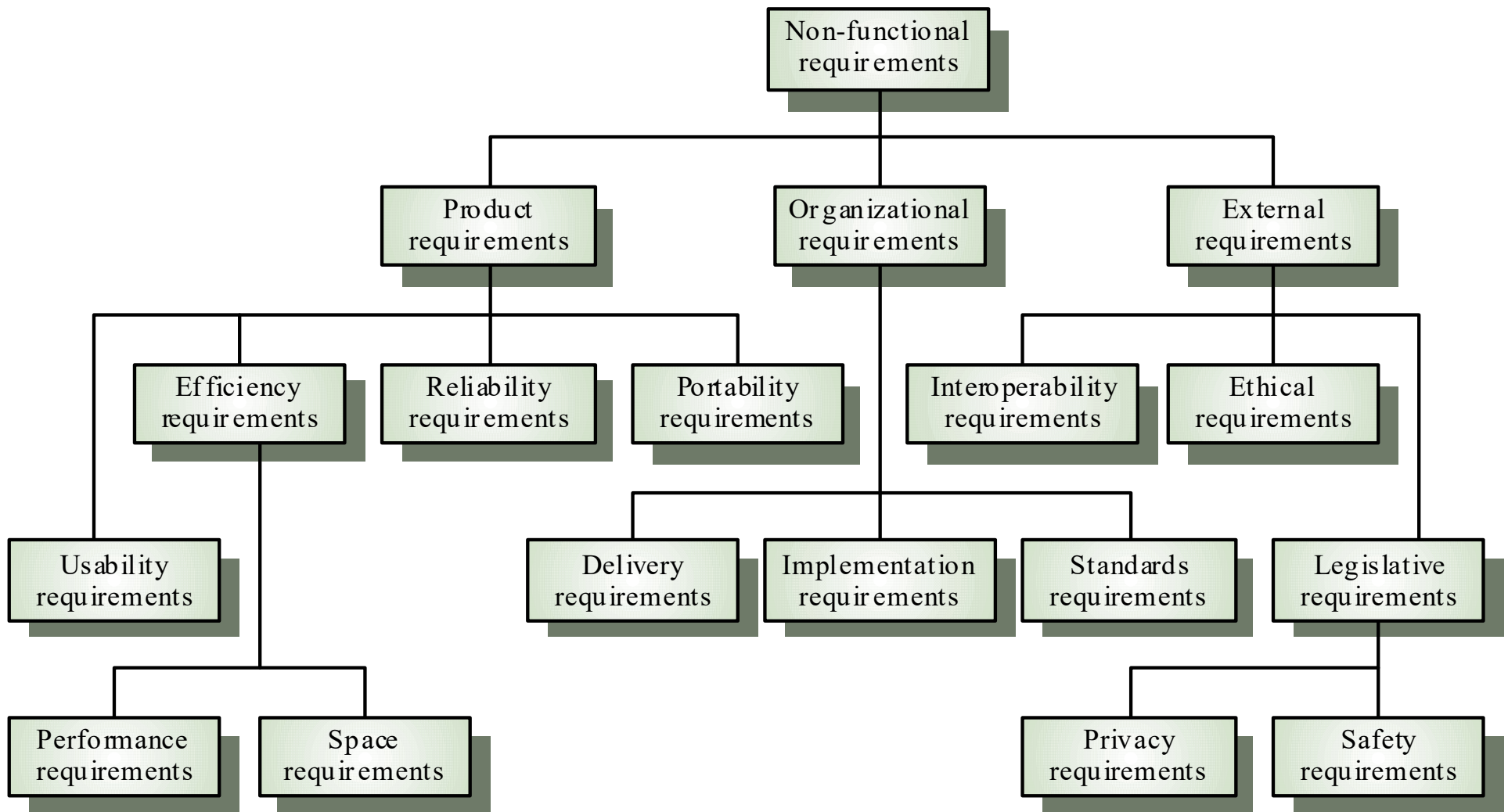
- **Organisational requirements**

- Requirements which are a **consequence of organisational policies and procedures** e.g. process standards used, implementation requirements, etc.

- **External requirements**

- Requirements which arise from factors which are **external to the system and its development process** e.g. interoperability requirements, legislative requirements, etc.

Non-functional requirement types



Goals and requirements

- **Non-functional requirements** may be very difficult to state precisely and imprecise requirements may be difficult to verify.
- **Goal**
 - A general intention of the user such as ease of use
- **Verifiable non-functional requirement**
 - A statement using some measure that can be objectively tested
- **Goals are helpful to developers** as they convey the intentions of the system users

Examples

- **A system goal**

- The system should be easy to use by experienced controllers and should be organised in such a way that user errors are minimised.

- **A verifiable non-functional requirement**

- Experienced controllers shall be able to use all the system functions after a total of two hours training.
- After this training, the average number of errors made by experienced users shall not exceed two per day.

Non Functional Requirements Measures

Property	Measure
Speed	Processed transactions/second User/Event response time Screen refresh time
Size	K Bytes Number of RAM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

Requirements interaction

- **Conflicts between different non-functional requirements are common in complex systems**
- Spacecraft system
 - **To minimise weight**, the number of separate chips in the system should be minimised
 - **To minimise power consumption**, lower power chips should be used
 - **However, using low power chips may mean that more chips have to be used.**

Which is the most critical requirement?



Domain requirements

- Derived from the application domain and describe system characteristics and features that reflect the domain
- May be new functional requirements, constraints on existing requirements or define specific computations
- If domain requirements are not satisfied, the system may be unworkable

Domain requirements problems

- Understandability

- Requirements are expressed in the language of the application domain
- This is often not understood by software engineers developing the system

- Implicitness (from the perspective of domain specialists)

- Domain specialists understand the area so well that they do not think of making the domain requirements explicit

User requirements

- **Should describe functional and non-functional requirements** so that they are understandable by system users who don't have detailed technical domain knowledge
- **User requirements are defined using natural language, tables and diagrams**

Problems with natural language

- **Lack of clarity**
 - Precision is difficult without making the document difficult to read
- **Requirements confusion**
 - Functional and non-functional requirements tend to be mixed-up
- **Requirements amalgamation**
 - Several different requirements may be expressed together

Guidelines for writing requirements

- Invent a standard format and use it for all requirements
- Use language in a consistent way. Use **shall** for mandatory requirements, **should** for desirable requirements
- Use **text highlighting** to identify key parts of the requirement

Avoid the use of computer jargon !!!

Requirements and design

- **In principle**, requirements should state **what the system should** do and the design should describe **how it does this**
- **In practice**, requirements and design are inseparable
 - A system architecture may be designed to structure the requirements
 - The system may inter-operate with other systems that generate design requirements
 - The use of a specific design may be a domain requirement

Problems with NL specification

- **Ambiguity**

- The readers and writers of the requirement must interpret the same words in the same way. NL is naturally ambiguous so this is very difficult

- **Over-flexibility**

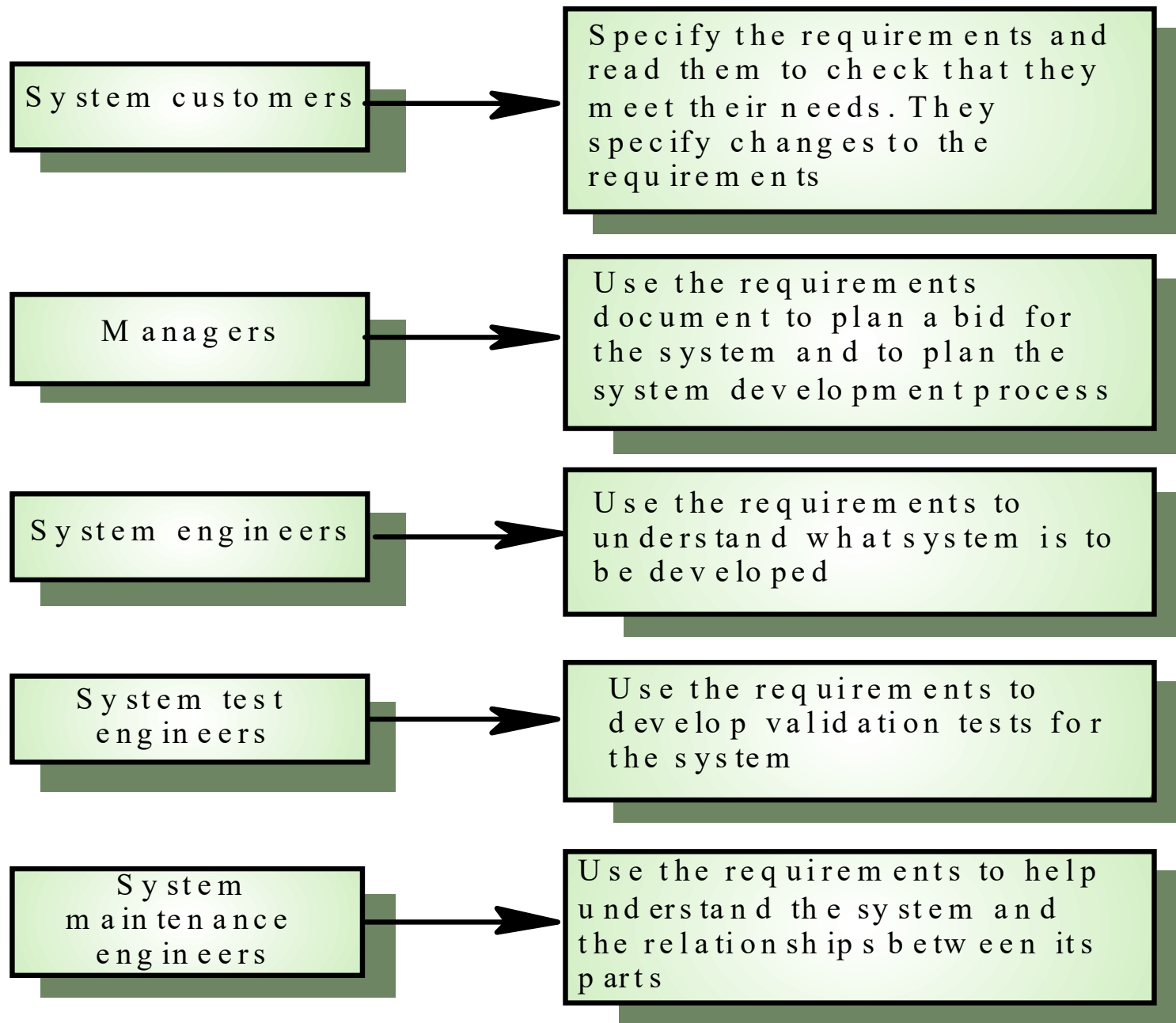
- The same thing may be said in a number of different ways in the specification

- **Lack of modularisation**

- NL structures are inadequate to structure system requirements

Alternatives in addition to NL specification

Notation	Description
Structured natural language	This approach depends on defining standard forms or templates to express the requirements specification.
Design description languages	This approach uses a language like a programming language but with more abstract features to specify the requirements by defining an operational model of the system.
Graphical notations	A graphical language, supplemented by text annotations is used to define the functional requirements for the system. An early example of such a graphical language was SADT (Ross, 1977; Schoman and Ross, 1977). More recently, use-case descriptions (Jacobsen, Christerson et al., 1993) have been used. I discuss these in the following chapter.
Mathematical specifications	These are notations based on mathematical concepts such as finite-state machines or sets. These unambiguous specifications reduce the arguments between customer and contractor about system functionality. However, most customers don't understand formal specifications and are reluctant to accept it as a system contract.



Users of a
requirements
document

Video Link – Functional vs Non-Functional Requirements

https://www.youtube.com/watch?v=NE1_cAWzQLM