SOFTWARE ENGINEERING

Prepared by:
B.Pranalini

Unit 2 Part 1: 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 **how software requirements may be organised** in a requirements document

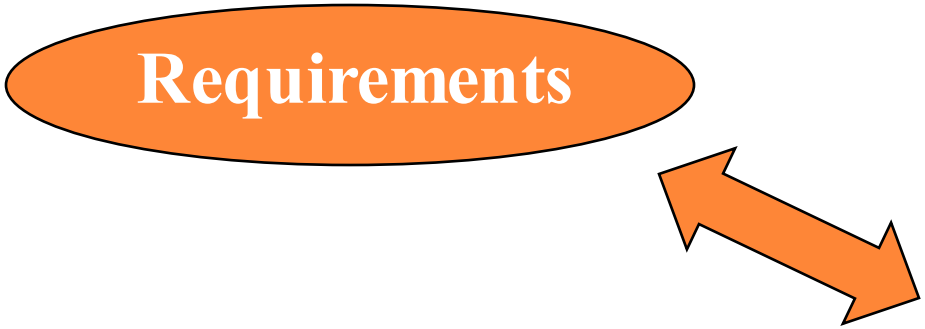
Topics covered

- **Functional and non-functional requirements**
- **User requirements**
- **Interface Specification**
- **The software 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

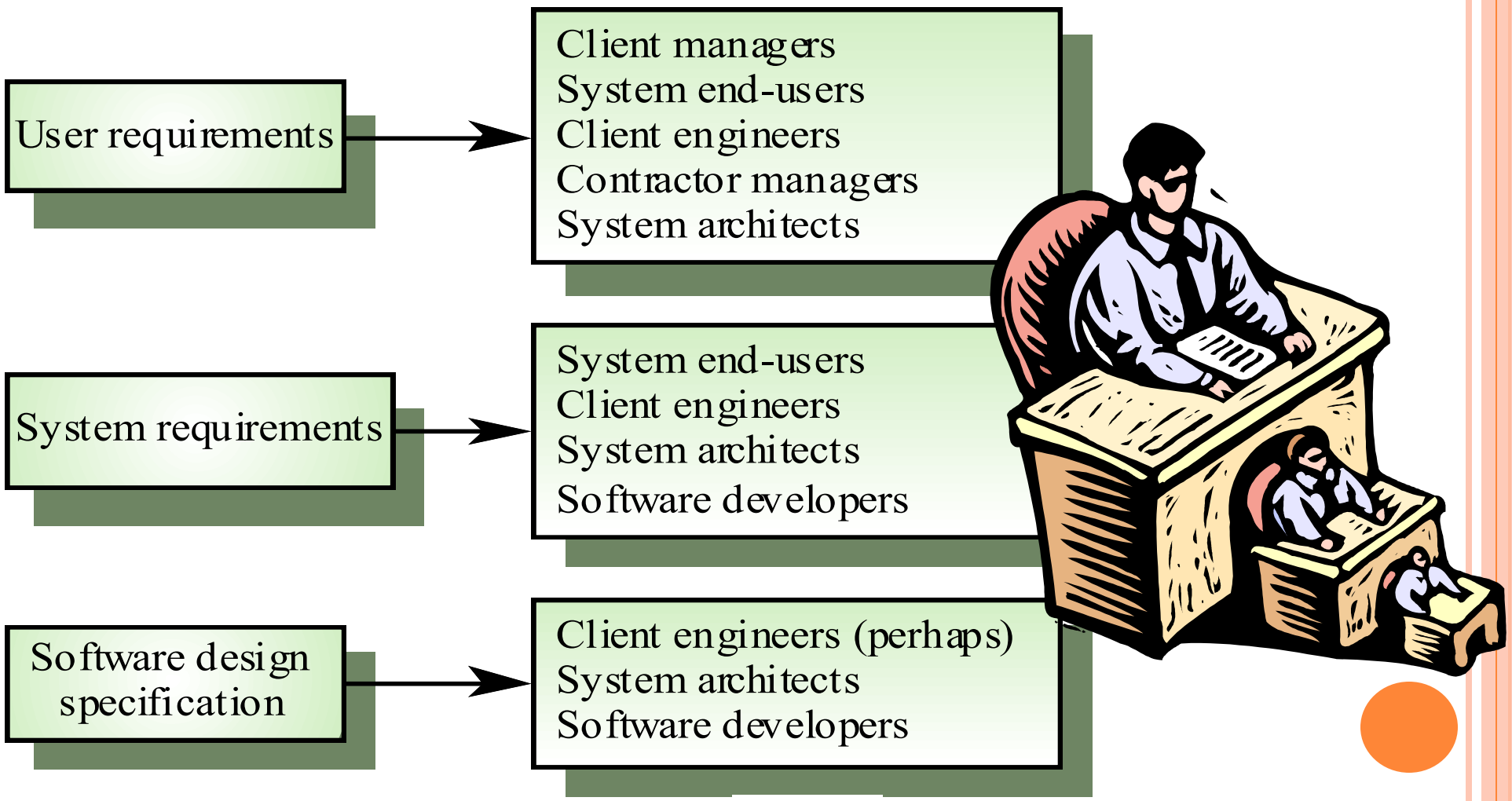
What is a requirement?

- It may range from a **high-level** abstract statement of a service or of a system constraint to a **detailed** mathematical functional specification
- This is inevitable as requirements may serve a **dual function**
  - May be the basis for a bid for a contract - therefore must be open to interpretation
  - May be the basis for the contract itself - therefore must be defined in detail
  - Both these statements may be called requirements

Types of requirement

- **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 BUT** functional system requirements should describe the system services in detail

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
- Consider the term ‘appropriate viewers’
  - **User intention** - special purpose viewer for each different document type
  - **Developer interpretation** - Provide a text viewer that shows the contents of the document

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

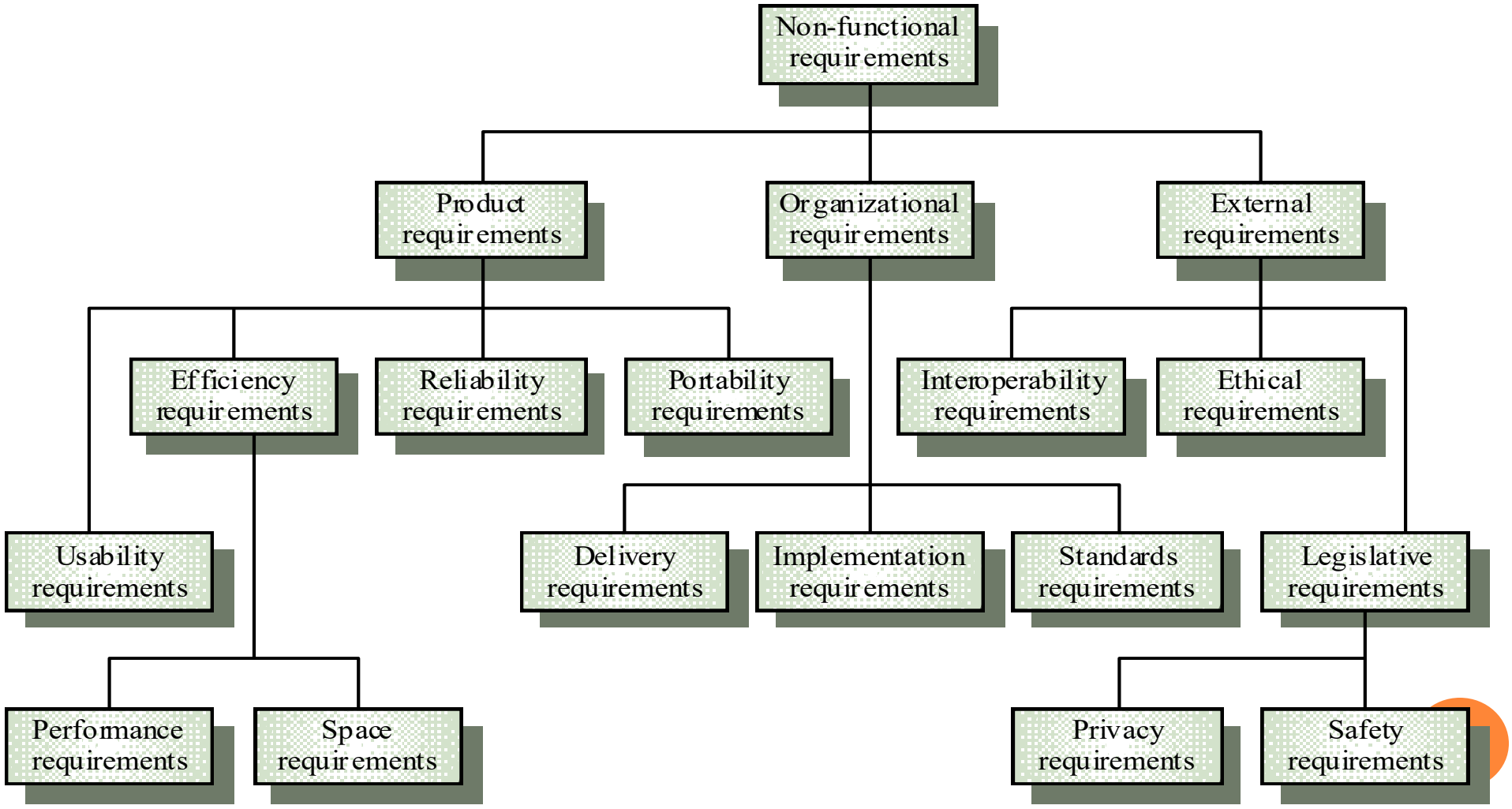
Define system properties and constraints e.g. reliability, response time and storage requirements. Constraints are I/O device capability, system representations, etc.

- **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



Non-functional requirements examples

- Product requirement
  - It shall be possible for all necessary communication between the APSE(Ada Programming Support Environment) and the user to be expressed in the standard Ada character set
- Organisational requirement
  - The system development process and deliverable documents shall conform to the process and deliverables defined in XYZCo-SP-STAN-95
- External requirement
  - The system shall not disclose any personal information about customers apart from their name and reference number to the operators of the system

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.

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.

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**
  - 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 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 !!!

Interface specification

- **Most systems must operate with other systems** and **the operating interfaces** must be specified as part of the requirements

- **Three types of interface** may have to be defined
  - **Procedural interfaces**
  - **Data structures that are exchanged**
  - **Data representations**

- Formal notations are an effective technique for interface specification

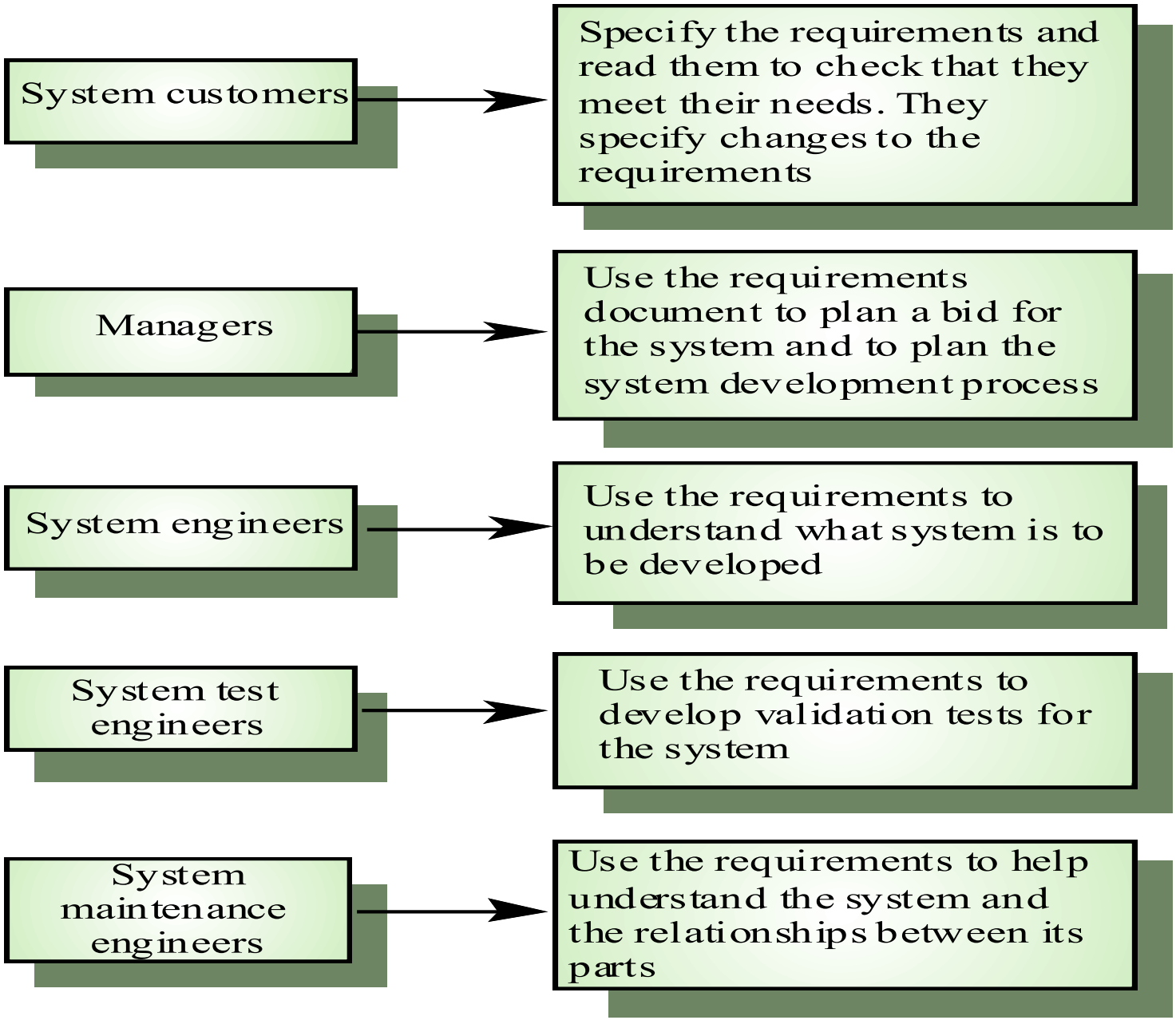
PDL INTERFACE DESCRIPTION

```
interface PrintServer {

// defines an abstract printer server
// requires: interface Printer, interface PrintDoc
// provides: initialize, print, displayPrintQueue, cancelPrintJob, switchPrinter

   void initialize ( Printer p ) ;
   void print ( Printer p, PrintDoc d ) ;
   void displayPrintQueue ( Printer p ) ;
   void cancelPrintJob (Printer p, PrintDoc d) ;
   void switchPrinter (Printer p1, Printer p2, PrintDoc d) ;
} //PrintServer
```

The requirements document

- The requirements document is the official statement of what is required of the system developers
- Should include both a definition and a specification of requirements
- It is NOT a design document. As far as possible, it should set of WHAT the system should do rather than HOW it should do it

USERS OF A REQUIREMENTS DOCUMENT



Requirements document requirements

- Specify external system behaviour
- Specify implementation constraints
- Easy to change
- Serve as reference tool for maintenance
- Record forethought about the life cycle of the system i.e. predict changes
- Characterise responses to unexpected events

IEEE requirements standard

- Introduction
- General description
- Specific requirements
- Appendices
- Index
- This is a generic structure that must be instantiated for specific systems

Chapter	Description
Preface	This should define the expected readership of the document and describe its version history, including a rationale for the creation of a new version and a summary of the changes made in each version.
Introduction	This should describe the need for the system. It should briefly describe the system's functions and explain how it will work with other systems. It should also describe how the system fits into the overall business or strategic objectives of the organization commissioning the software.
Glossary	This should define the technical terms used in the document. You should not make assumptions about the experience or expertise of the reader.
User requirements definition	Here, you describe the services provided for the user. The non-functional system requirements should also be described in this section. This description may use natural language, diagrams, or other notations that are understandable to customers. Product and process standards that must be followed should be specified.
System architecture	This chapter should present a high-level overview of the anticipated system architecture, showing the distribution of functions across system modules. Architectural components that are reused should be highlighted.
System requirements specification	This should describe the functional and non-functional requirements in more detail. If necessary, further detail may also be added to the non-functional requirements. Interfaces to other systems may be defined.
System models	This might include graphical system models showing the relationships between the system components, the system, and its environment. Examples of possible models are object models, data-flow models, or semantic data models.
System evolution	This should describe the fundamental assumptions on which the system is based, and any anticipated changes due to hardware evolution, changing user needs, and so on. This section is useful for system designers as it may help them avoid design decisions that would constrain likely future changes to the system.
Appendices	These should provide detailed, specific information that is related to the application being developed; for example, hardware and database descriptions. Hardware requirements define the minimal and optimal configurations for the system. Database requirements define the logical organization of the data used by the system and the relationships between data.
Index	Several indexes to the document may be included. As well as a normal alphabetic index, there may be an index of diagrams, an index of functions, and so on.

Requirements Document Structure

Key points

- Requirements set out what the system should do and define constraints on its operation and implementation
- Functional requirements set out services the system should provide
- Non-functional requirements constrain the system being developed or the development process
- User requirements are high-level statements of what the system should do

Key points

- User requirements should be written in natural language, tables and diagrams
- System requirements are intended to communicate the functions that the system should provide
- System requirements may be written in structured natural language, a PDL or in a formal language
- A software requirements document is an agreed statement of the system requirements

Thank You