Software requirements

- The descriptions of the services provided by the product and the constraints under which it operates and is developed
- Def. IEEE Std 610.12 (1990):
  - (A) A condition or capability needed by a user to solve a problem or achieve an objective
  - (B) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document
  - (C) A documented representation of a condition or capability as in definition (A) or (B)

Software requirements (2)

- Produced by applying a requirements engineering process
- ***Requirements abstraction*** (Davis, 1993)

  "*If a company wishes to let a contract for a large software development project, it must define its **needs** in a sufficiently abstract way that a solution is not pre-defined. The requirements must be written so that several contractors can bid for the contract, offering, perhaps, different ways of meeting the client organization's needs. Once a contract has been awarded, the contractor must write a **system definition** for the client in more detail so that the client understands and can validate what the software will do. Both of these documents may be called the **requirements document** for the system*"

Types of requirements

- **User requirements**
  - statements in natural language plus diagrams of the services the product provides and its operational constraints
  - written for (and with) customers
- **System requirements**
  - a structured document setting out detailed descriptions of the product 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

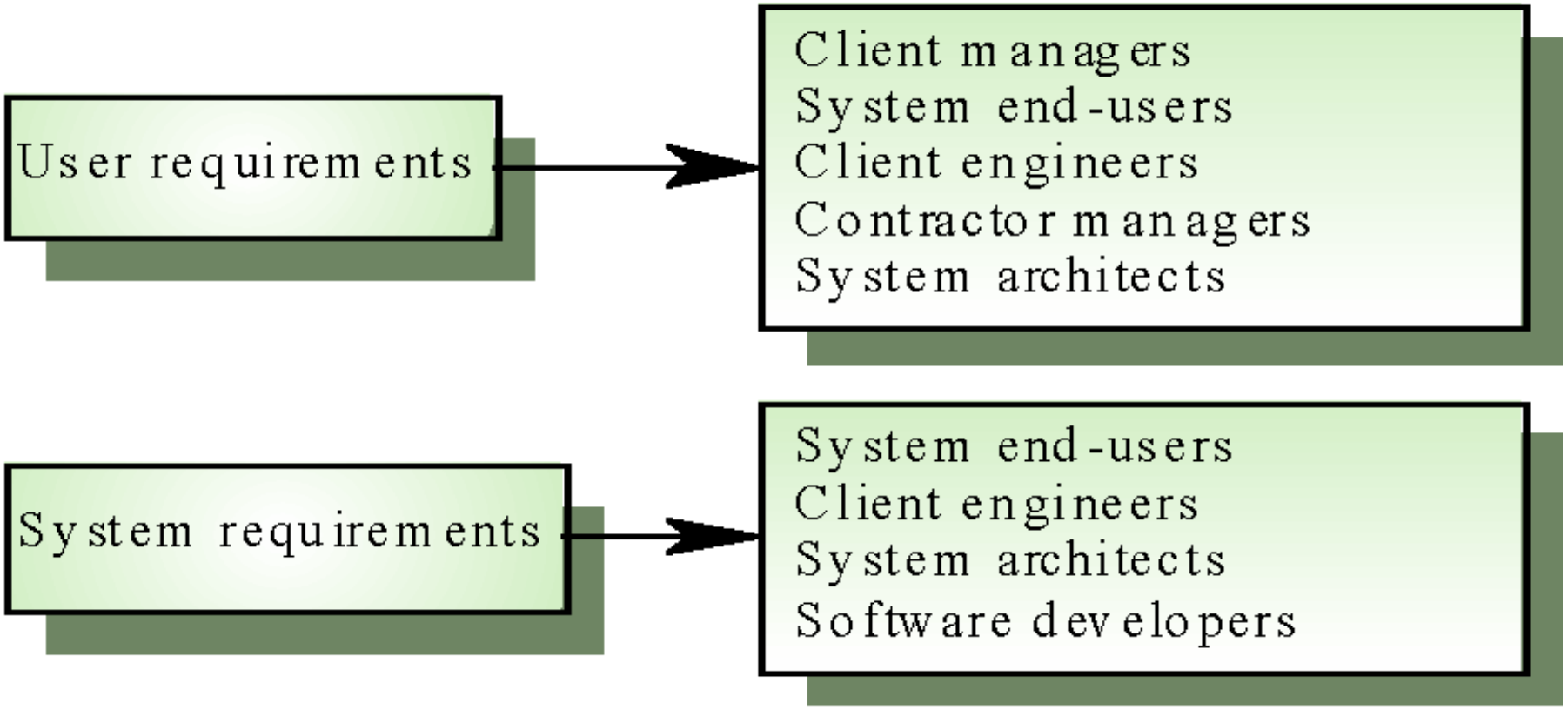
Terms definition

- *Customer* (*client*)
  the person or organization that places an order for the acquisition of a software product
- *Supplier* (*contractor*)
  the person or organization who builds (and often maintains) the software product for the customer
- *End-user*
  the person or organization that uses the delivered software product (may correspond to the customer)

Example requirements

- **User requirement**
  1. The product shall provide an appropriate viewer for the user to read and visualize external files (generated by use of other tools)
- **System requirement**
  - 1.1 The user shall have the possibility to define the type of the external file
  - 1.2 Each external file type shall be associated to the tool that is used to generate files of that type
  - 1.3 Each external file type shall be associated to an icon that is used to visualize files of that type
  - 1.4 The user shall have the possibility to define the icon of an external file type
  - 1.5 When a user selects a file icon the associated tool shall be run

Who reads the requirements?



Requirements *categories*

- ***Functional* requirements**

  statements of *services* the product should provide, how the product should *react* to particular inputs and how the product should *behave* in particular situations

  Ex.1 The user shall be able to search either all of the initial set of databases or select a subset from it

  Ex.2 The product shall provide appropriate viewers for the user to read documents in the document store

  Ex.3 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 *categories* (2)

- ***Non functional* requirements**

  describe *properties* and/or *constraints* on the services or functions offered by the product such as:

  - requirements which specify that the delivered product must behave in a particular way, (e.g., execution speed, reliability, etc.)
  - requirements which are a consequence of organizational policies and procedures (e.g., process standards used, implementation requirements, etc.)
  - requirements which arise from factors which are external to the product and its development process (e.g., interoperability requirements, legislative requirements, etc.)

Ex.1 The response time of the logon function shall not exceed 10 seconds

Ex.2 The product development process and deliverable documents shall conform to the process and deliverables defined in XYZCo-SP-STAN-95

Ex.3 The product shall not disclose any personal information about customers apart from their name and reference number to the operators

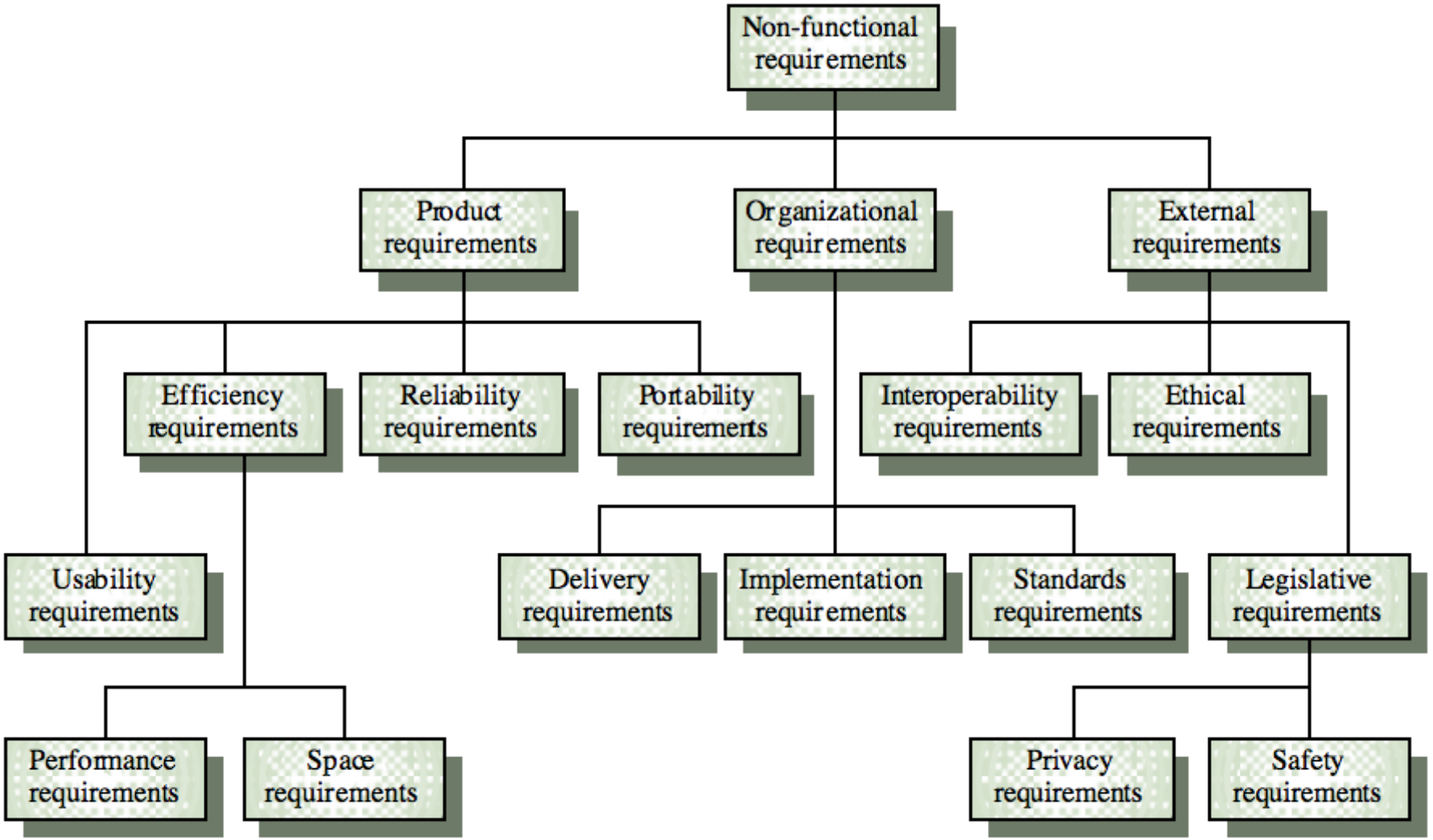
Requirements *categories* (3)

- **Domain requirements**
  - derived from the application domain and describe product characteristics and features that reflect the domain
  - may be new *functional* requirements, constraints on existing requirements (i.e., *non functional* requirements) or define specific computations

Ex.1 There shall be a standard user interface to all databases which shall be based on the Z39.50 standard

Ex.2 Because of copyright restrictions, some documents must be deleted immediately on arrival. Depending on the user's requirements, these documents will either be printed locally on the system server for manually forwarding to the user or routed to a network printer.

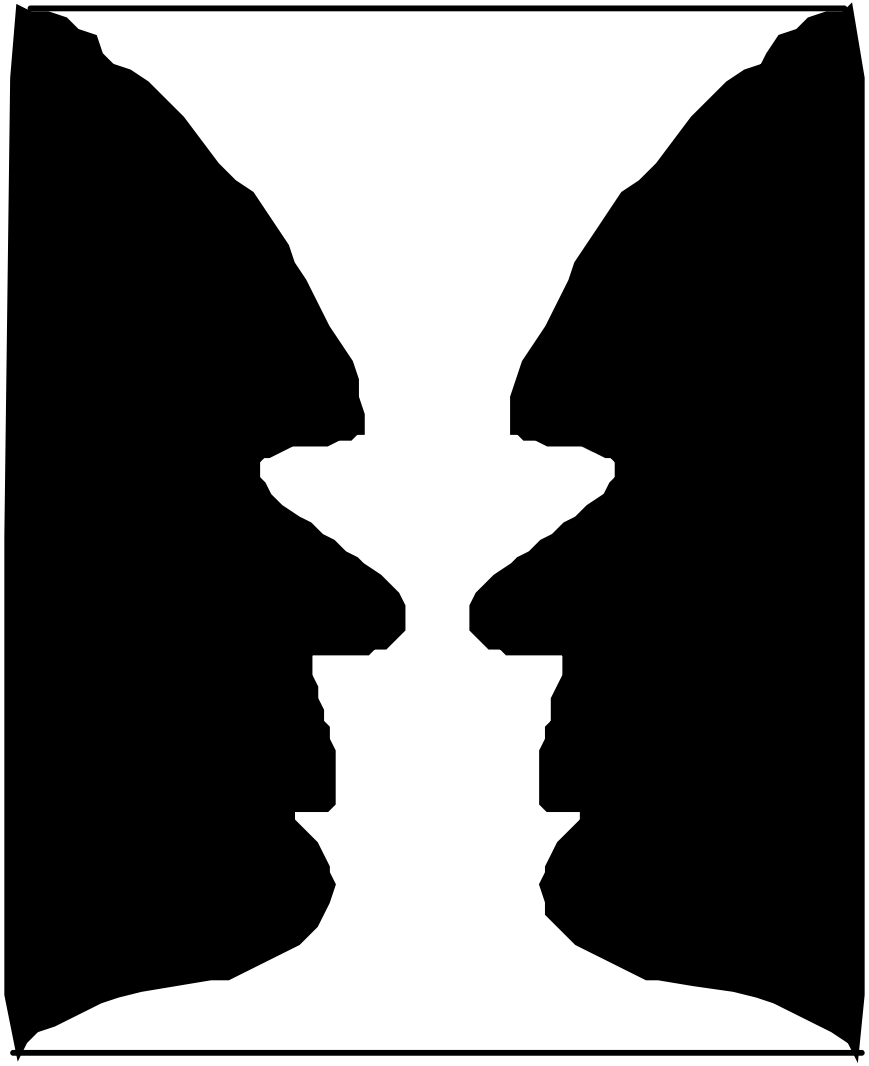
A partial classification of *non functional requirements*



Software requirements problems

Ambiguity

What do you see?



Software requirements problems (2)

- **Ambiguity**: requirements that may be understood in various ways

  Example 1: specify a time constraint without providing the time zone (in a product that is used to schedule and manage intercontinental calls)

  Example 2: meaning of "appropriate viewer"

  - *user interpretation*: a specific viewer for each file type
  - *developer interpretation*: a generic text viewer

- **Incompleteness**: the requirements do not include the whole set of important characteristics
- **Inconsistency**: clashes or contradictions among requirements

  Example

  - *Req. 1*: each data entry form shall contain no more than 5 editable fields
  - *Req. 2*: the personal details data entry form shall provide the following fields: first name, middle name, last name, date of birth, place of birth, slow mail address, telephone number, fax number and email address

Requirements verifiability

- Non functional requirements generically provided by the user (e.g., the product has to be *easy-to-use*) may turn to be not quantifiable and thus hard to verify

- It is mandatory to specify non functional requirements by use of a measure that eventually allows to quantitatively verify if the product meets or not those requirements

Example 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

User Requirements

- Should describe functional and non-functional requirements so that they are understandable by product users who don't have detailed technical knowledge
- User requirements are defined using *natural language*, tables and diagrams, by using the following guidelines:
  - use a standard format for all requirements
  - use language in a consistent way (e.g., 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

User requirement example

3.5.1 Adding nodes to a design

3.5.1.1 **The editor shall provide a facility for users to add nodes of a specified type to their design.**

3.5.1.2 The sequence of actions to add a node should be as follows:

1. The user should select the type of node to be added.
2. The user should move the cursor to the approximate node position in the diagram and indicate that the node symbol should be added at that point.
3. The user should then drag the node symbol to its final position.

Rationale: The user is the best person to decide where to position a node on the diagram. This approach gives the user direct control over node type selection and positioning.

Specification: ECLIPSE/WS/Tools/DE/FS/Section 3.5.1

System Requirements (Specifications)

- More detailed specifications of user requirements
- Serve as a basis for designing the product
- May be used as part of the contract
- May be expressed using various notations

Notation	Description
Structured natural language	This approach depends on defining standard forms or templates to express the requirements specification.
Program description languages (PDL)	This approach uses a language like a programming language (*PDL*, *Program Description 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. The graphical language is used to define *system models*
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.

System requirement example

- specified by use of a *form* in natural language

ECLIPSE/Workstation/Tools/DE/FS/3.5.1	
Function	Add node
Description	Adds a node to an existing design. The user selects the type of node, and its position. When added to the design, the node becomes the current selection. The user chooses the node position by moving the cursor to the area where the node is added.
Inputs	Node type, Node position, Design identifier.
Source	Node type and Node position are input by the user, Design identifier from the database.
Outputs	Design identifier.
Destination	The design database. The design is committed to the database on completion of the operation.
Requires	Design graph rooted at input design identifier.
Pre-condition	The design is open and displayed on the user's screen.
Post-condition	The design is unchanged apart from the addition of a node of the specified type at the given position.
Side-effects	None
Definition:	ECLIPSE/Workstation/Tools/DE/RD/3.5.1

System requirement example (2)

- specified by use of *PDL* (*Java-like*)

```
class ATM {
 // declarations here
 public static void main (String args[]) throws InvalidCard {
    try {
       thisCard.read () ;  // may throw InvalidCard exception
       pin = KeyPad.readPin () ; attempts = 1 ;
       while ( !thisCard.pin.equals (pin) & attempts < 4 )
          { pin = KeyPad.readPin () ;  attempts = attempts + 1 ;
          }
          if (!thisCard.pin.equals (pin))
            throw new InvalidCard ("Bad PIN");
       thisBalance = thisCard.getBalance () ;
       do {  Screen.prompt (" Please select a service ") ;
          service = Screen.touchKey () ;
          switch (service) {
            case Services.withdrawalWithReceipt:
                          receiptRequired = true ;

            .........
```

System requirement example (3)

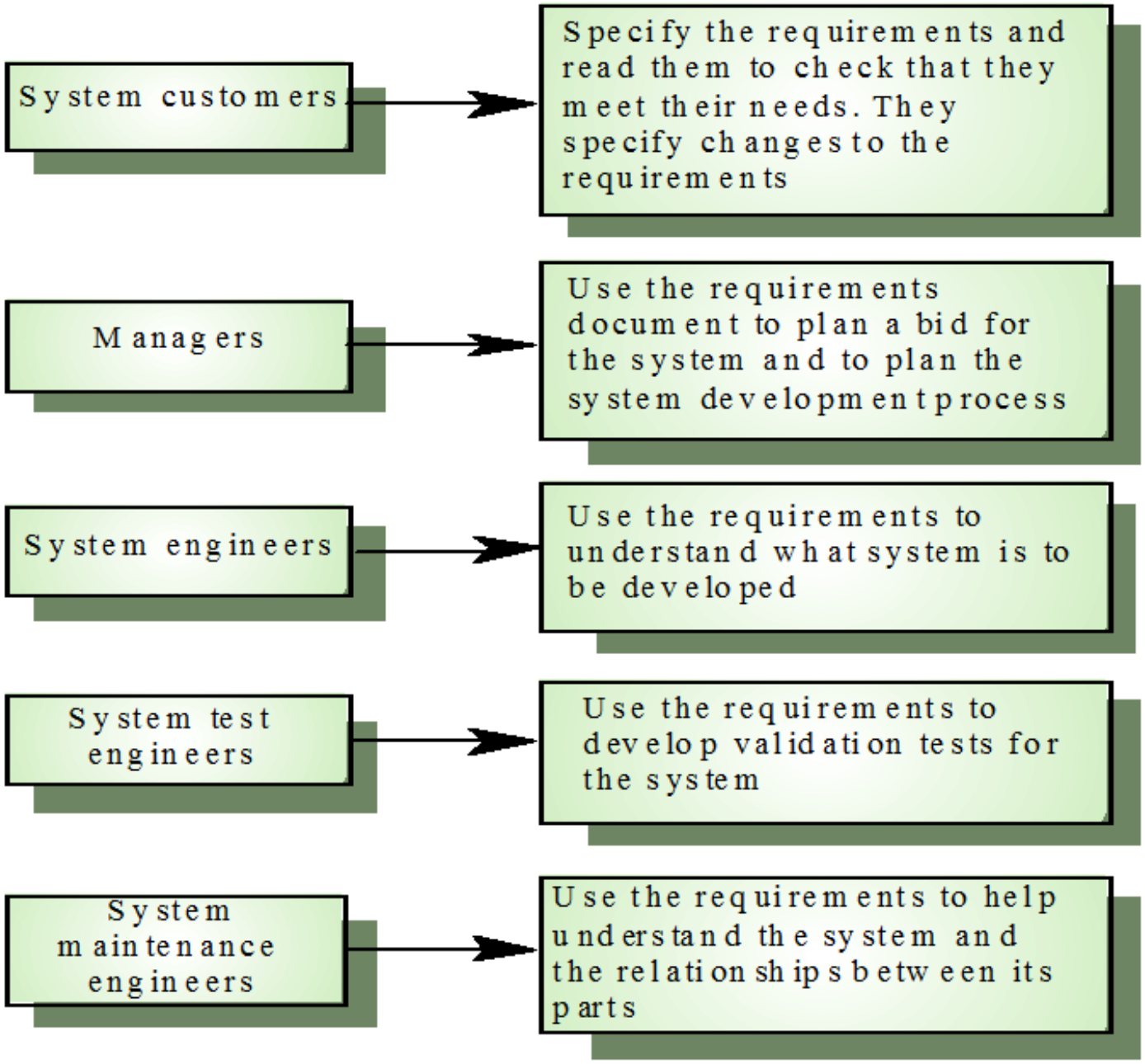
- interface specification in *PDL*

```
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 analysis document (or specification document)

- The requirements document is the official statement of what is required of the product developers
- Should include both a *definition* and a *specification* of requirements
- Should set of **WHAT** the product should do (*problem domain*) rather than **HOW** it should do it (*solution domain*)

Users of the requirements analysis document



A *template* for the req analysis doc

based on the **IEEE 830-1998** standard
(*IEEE Recommended Practice for Software Requirements Specifications*) page 1/2

Preface

expected readership, version history, changes summary

Introduction

purpose, brief description of the system, interaction with other systems, scope within the business context

Glossary

definition of technical terms used in the document

User requirements definition

functional and non-functional user requirements

System architecture

high-level overview of the system components

System requirements specification

functional and non-functional system requirements

A *template* for the req analysis doc

based on the **IEEE 830-1998** standard
(*IEEE Recommended Practice for Software Requirements Specifications*) page 2/2

System models

description of the relationships between the system components and the system and its environment

System evolution

assumptions on which the system is based and anticipated changes (hardware evolution, user needs changes, etc.)

Appendices

specific information related to the application which is being developed (ex. HW and DB descriptions)

Index

table of contents, alphabetic index, list of diagrams, etc.