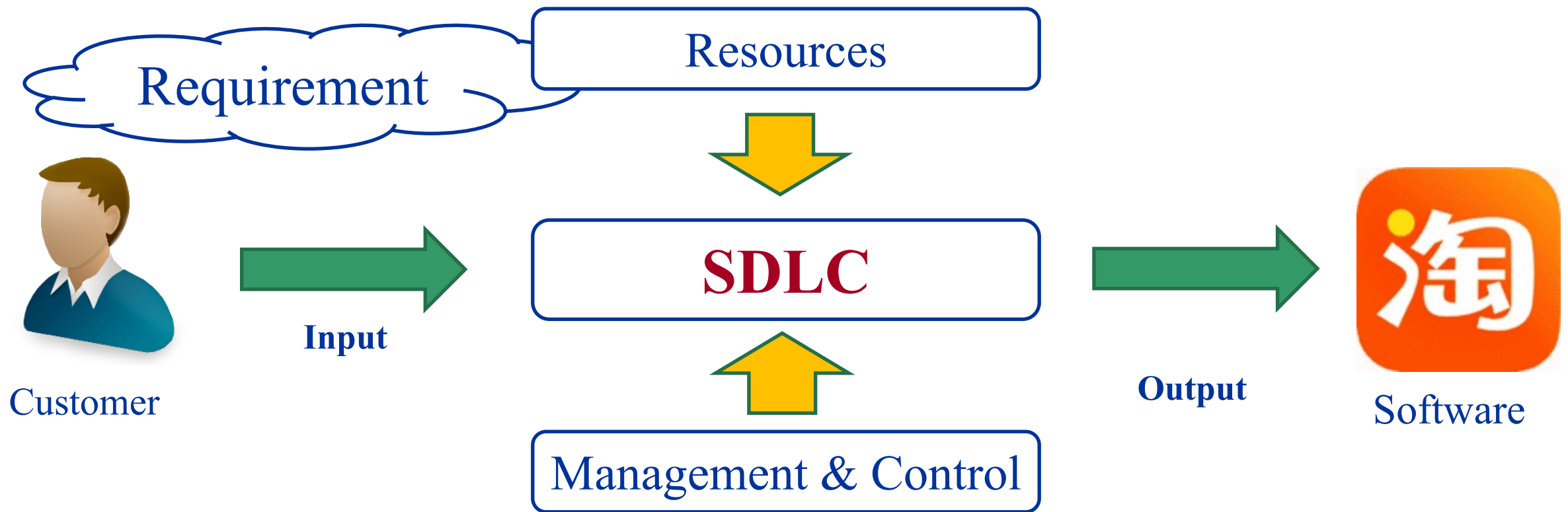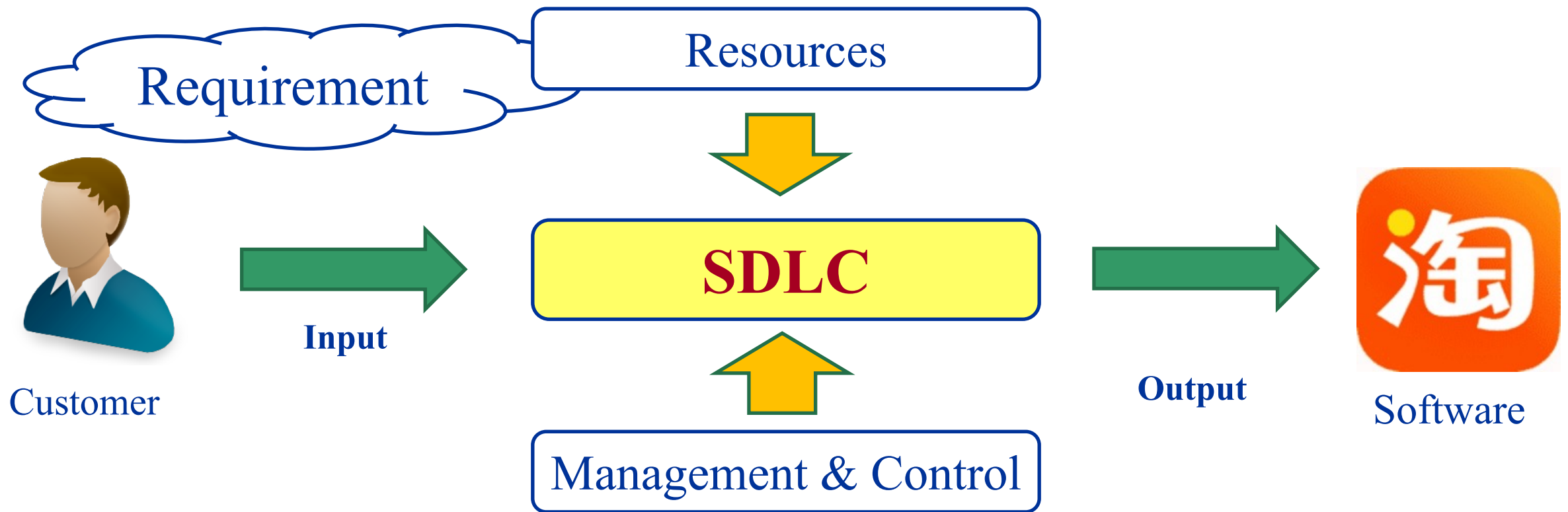# Logistics

- Homework 1 DUE today; We donnot accept late submission;

- Weekly Report starts from this Week;

- Weekly Report Template is available on BB;

- Office Hour: (from this Week)
  – Time: 7:00 – 8:00 PM, Wed (W5-W16)
  – Loc: 1-106A @SIST

Software Development Life Cycle (SDLC)

Requirement

Resources

Customer

Input

SDLC

Management & Control

Output

Software

Software Development Life Cycle (SDLC)

Requirement

Resources

Customer

Input

SDLC

Output

Software

Management & Control

Lecture 7: Requirement Engineering

Yutian Tang

SIST@ShanghaiTech

Software Development Life Cycle

| Requirement Analysis | Design | Implementation | Testing | Maintenance |

Software Development Life Cycle (2)

| Requirement Analysis | Design | Implementation | Testing | Maintenance |

Requirement

Customer

Input

Requirement Analysis

Output

Software Requirements Specification v. 1.1
for: Software product
by: Customer

1 Introduction
Purpose, scope, …

2 Overall description
Functions, users, constraints, …

3 Requirements
functional, non-functional, interfaces, …

Requirement Specification

Stakeholders for software

Requirements

Engineers

Domain Experts

Regulation

Customer

Regulators
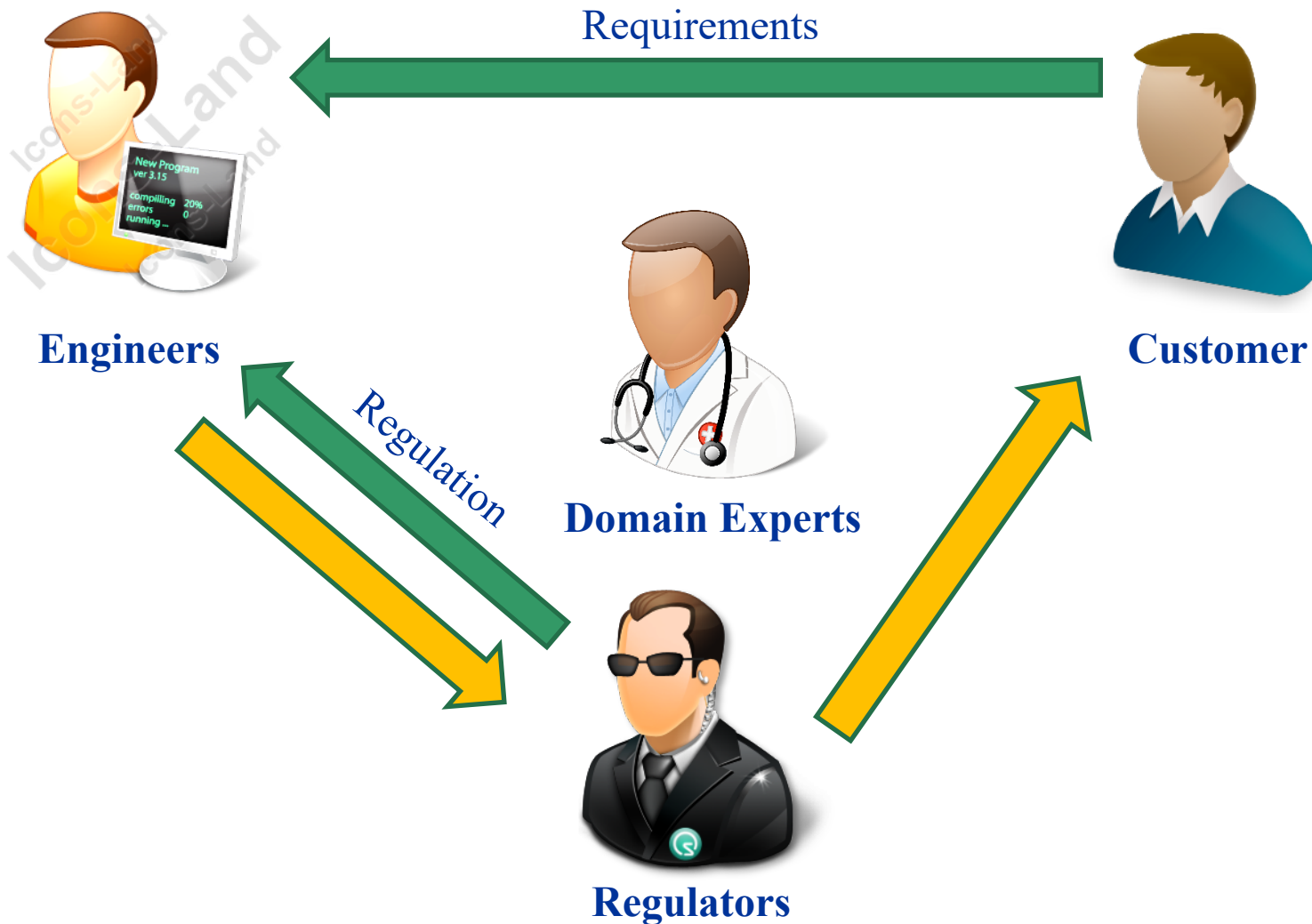
Software Requirement

- The requirements for a system are the descriptions of the services that a system should provide and the constraints on its operation.

- The process of finding out, analyzing, documenting and checking these services and constraints is called requirements engineering.

- Requirements can be categorized into (levels of descriptions):
  - User requirements (high-level abstraction)
  - System requirements (detailed description)

- Requirements can be categorized into (type):
  - Functional requirement
  - Non-functional requirement
  - Domain requirement

Software Requirement (2)

(high-level abstraction)

User requirements

Customer

快！

System requirements

(detailed description)

Engineers

Response time < 1.5s

User Requirement

- User requirements are statements, in a natural language plus diagrams, of what services the system is expected to provide to system users and constraints under which it must operate.

- The user requirements may vary from broad statements of system features required to detailed, precision descriptions of the system functionality

System Requirement

- System requirements are more detailed descriptions of
  - the software system's functions,
  - services, and
  - operational constraints.

- The system requirements document should define exactly what is to be implemented.

User requirements vs System requirements

User requirement

1. The Mentcare system shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month.
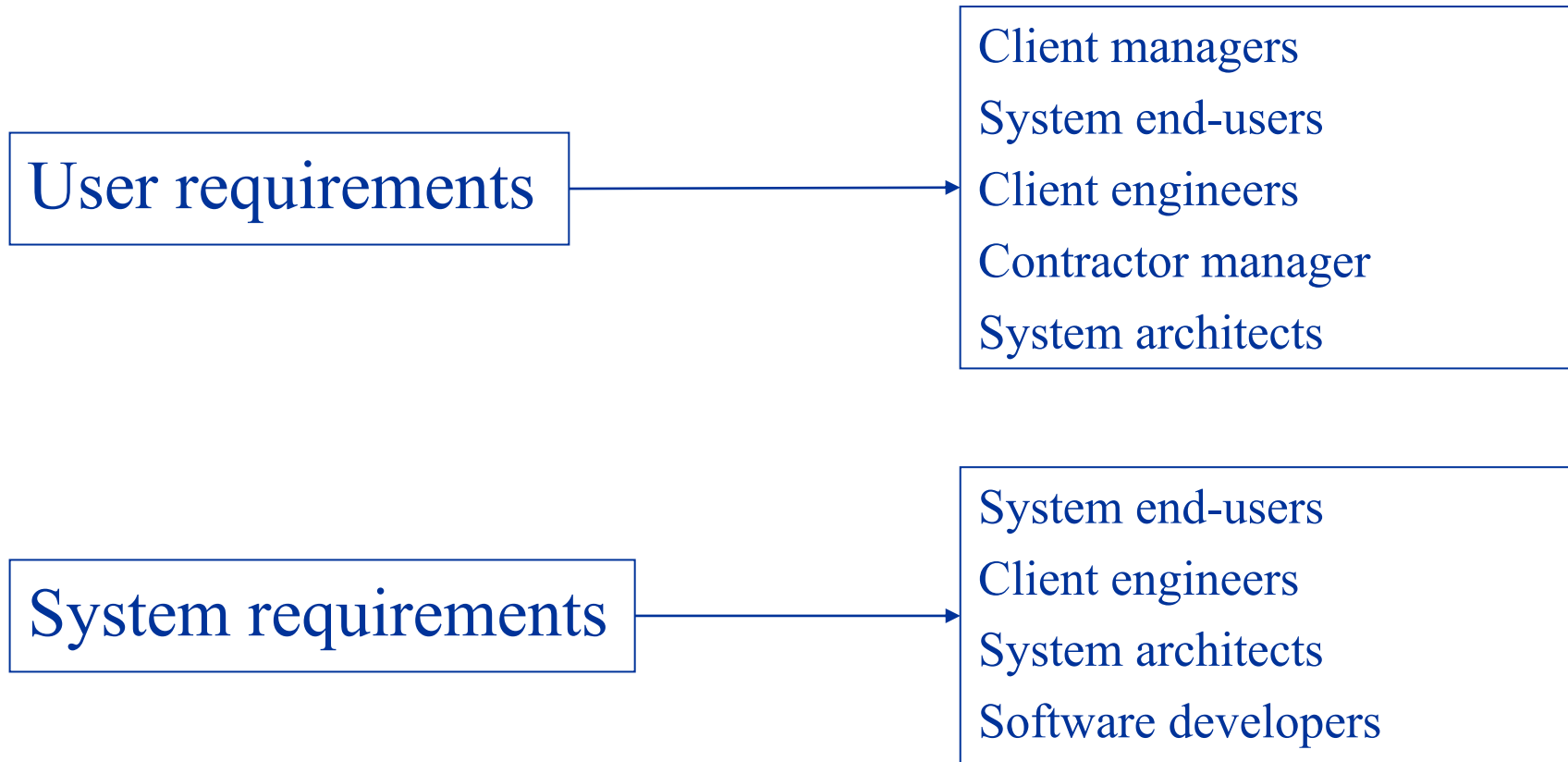
System requirement

1.1. On the last working day of each month, a summary of the drugs prescribed, their cost and the prescribing clinics shall be generated.

1.2 The system shall generate the report for printing after 17.30 on the last working day of the month.

1.3 A report shall be crated for each clinic and shall list the individual drug names, the total number of prescriptions, the number of doses prescribed and the total cost of the prescribed drugs.

1.4 If drugs are available in different does units (e.g. 10mg, 20mg, etc) separate reports shall be created for each does unit.

1.5 Access to drug cost reports shall be restricted to authorized users as listed on a management access control list.

User requirements vs System requirements (2)

- You need to write requirements at different levels of details because different type of readers use them in different ways.

- The readers of the user requirements are not usually concerned with how the system will be implemented and may be managers who are not interested in the detailed facilities of the system.

- The readers of the system requirements need to know more precisely what the system will do because they are concerned with how it will support the business processes or because they are involved in the system implementation.

User requirements vs System requirements (3)

User requirements → Client managers
System end-users
Client engineers
Contractor manager
System architects

System requirements → System end-users
Client engineers
System architects
Software developers

System stakeholders

- Any person or organization who is affected by the system in some way and so who has a legitimate interest

- Stakeholder types
  - End users
  - System managers
  - System owners
  - External stakeholders

Functional and Non-functional Requirement

Functional and Non-functional Requirement

- 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.
  - May state what the system should not do.
- Non-functional requirements
  - Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
  - Often apply to the system as a whole rather than individual features or services.
- Domain requirements
  - Constraints on the system from the domain of operation

Functional Requirement

- *Functional Requirement* These are statements of services the system should provide, how the system should react to particular inputs, and how the system should behave in particular situations.

- In some cases, the functional requirements may also explicitly state what the system should not do.

Functional Requirement (2)

- 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.

- Functional system requirements should describe the system services in detail.

Requirements imprecision

- Problems arise when functional requirements are not precisely stated.

- Ambiguous requirements may be interpreted in different ways by developers and users.

- Consider the term 'search' in requirement 1
  - User intention – search for a patient name across all appointments in all clinics;
  - Developer interpretation – search for a patient name in an individual clinic. User chooses clinic then search.

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, because of system and environmental complexity, it is impossible to produce a complete and consistent requirements document.
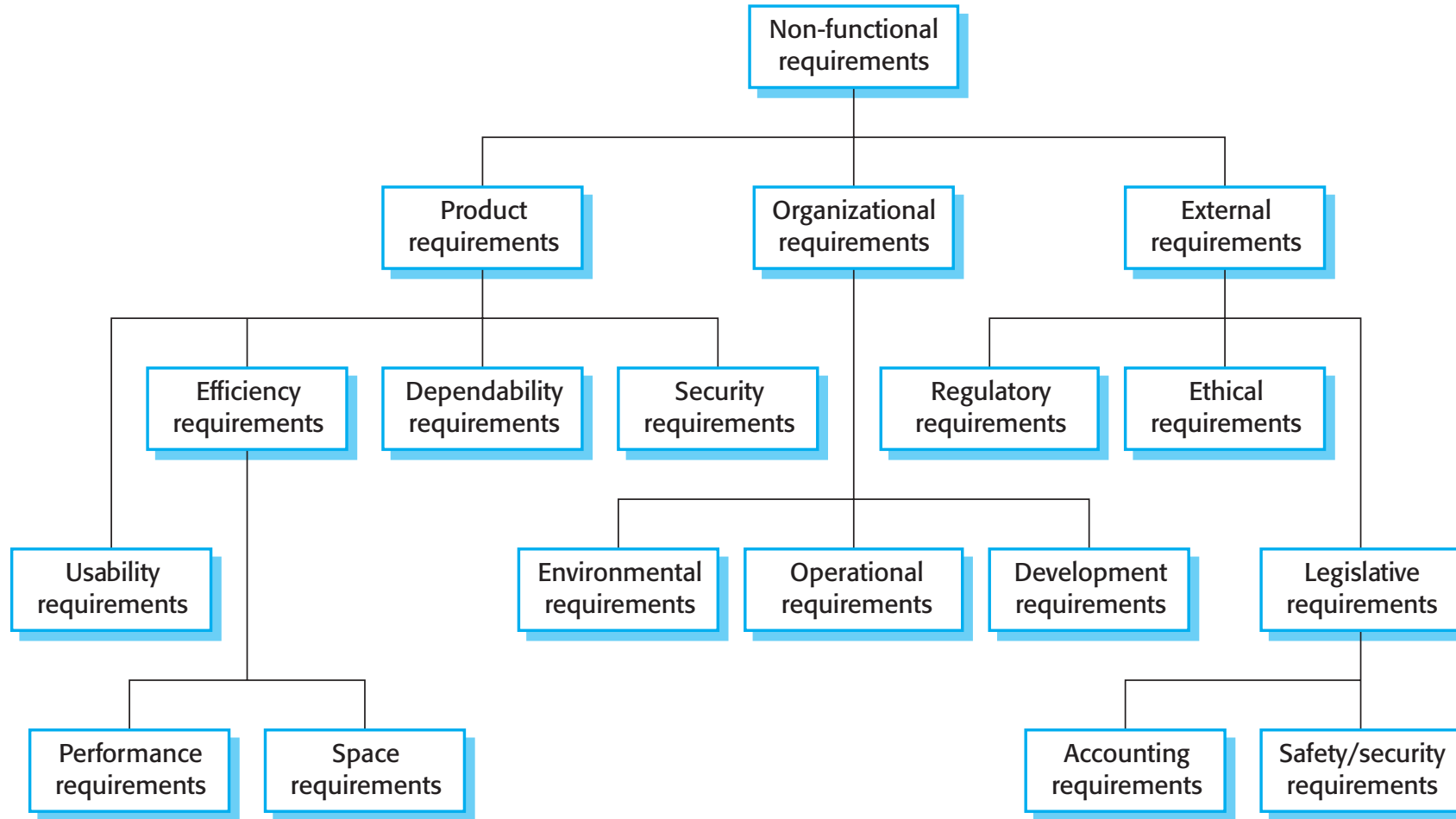
Recap: Functional Requirements

- Functions, tasks, or behaviors the system must fully support.
  - How user of the system use the system

- The "skeleton" of the system requirements
  - Should be captured in early iterations

- Need to distinguish "core functions" from "features"

Non-Functional Requirements

- These 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 IDE, programming language or development method.

- Non-functional requirements may be more critical than functional requirements. If these are not met, the system may be useless.

Non-Functional Requirements (2)

Non-functional requirements implementation

- Non-functional requirements may affect the overall architecture of a system rather than the individual components.
  - For example, to ensure that performance requirements are met, you may have to organize the system to minimize communications between components.
- A single non-functional requirement, such as a security requirement, may generate a number of related functional requirements that define system services that are required.
  - It may also generate requirements that restrict existing requirements.

Examples of Non-Functional Requirements

- User interface and human factors:
  - What type of user will be using the system?
  - Will more than one type of user be using the system?
  - What sort of training will be required for each type of user?
  - Is it particularly important that the system be easy to learn?
  - Is it particularly important that users be protected from making errors?
  - What sort of input/output devices for the human interface are available, and what are their characteristics?

Examples of Non-Functional Requirements

- Performance characteristics
  - Are there any speed, throughput, or response time constraints on the system?
  - Are there size or capacity constraints on the data to be processed by the system?
- Error handling and extreme conditions
  - How should the system respond to input errors?
  - How should the system respond to extreme conditions?

Examples of Non-Functional Requirements

- Quality issues
  - What are the requirements for reliability?
  - Must the system trap faults?
  - What is the maximum time for restarting the system after a failure?
  - Is it important that the system be portable (able to move to different hardware or operating system environments)?

- System Modifications
  - What parts of the system are likely candidates for later modification?
  - What sorts of modifications are expected (levels of adaptation)?
  - Might unwary adaptations lead to unsafe system states?

Identifying Non-functional Requirements

- Certain constraints are related to the design solution that are unknown at the requirements stage.

- Certain constraints are highly subjective and can only be determined through complex, empirical evaluations.

- Non-functional requirements tend to conflict and contradict.

- There is no 'universal' set of rules and guidelines for determining when nonfunctional requirements are optimally met.

Recap: Non-Functional Requirements

- Constraints placed on various attributes of system functions or tasks

- Equally important compared to functional requirements
  - Separate software products from software practices
- Sources
  - Domain: i.e. Human can tolerate up to 150ms delay in voice communication
  - Legacy: i.e. QWERTY keyboard
  - User: i.e. User want to operate the interface with one hand
  - Regulation: The system should switch to backup and resume within 1ms after the primary program crashes

Requirement Elicitation

- Step 1: (Business analyst) develops common understanding of the problem domain with (customers) and (domain experts)

- Step 2: (Business analyst) explains the problem to (the development team) and develop a design strategy

- Step 3: (Business analyst) presents the design strategy to the customer, and agree on technical solutions

Requirement Elicitation (2)

- Eliciting and understanding requirements from stakeholders can be difficult for several reasons:

- 1. Stakeholders often don't know what they want from a computer system except in the most general terms;

  – They may make unrealistic demands because they don't know what is and isn't feasible.

- 2. Stakeholders in a system naturally express requirements in their own terms and with implicit knowledge of their own work.

  – Requirements engineers, without experience in the customer's domain, may not understand these requirements.

Requirement Elicitation (3)

- 3. Different stakeholders, with diverse requirements, may express their requirements in different ways.

    – Requirement engineers, without experience in the customer's domain, may not understand these requirements.

- 4. Political factors may influence the requirements of a system. Managers may demand specific system requirements because these will allow them to increase their influence in the organization.

- 5. The economic and business environment in which the analysis take place is dynamic.

Requirement Elicitation – Interviewing

- Formal or informal interviews with stakeholders are part of most RE processes.

- Types of interview
  - Closed interviews based on pre-determined list of questions
  - Open interviews where various issues are explored with stakeholders.

- Effective interviewing
  - Be open-minded, avoid pre-conceived ideas about the requirements and are willing to listen to stakeholders.
  - Prompt the interviewee to get discussions going using a springboard question, a requirements proposal, or by working together on a prototype system.

Problems with Interviews

- Application specialists may use language to describe their work that isn't easy for the requirements engineer to understand.

- Interviews are not good for understanding domain requirements
  - Requirements engineers cannot understand specific domain terminology;
  - Some domain knowledge is so familiar that people find it hard to articulate or think that it isn't worth articulating.

Requirement Elicitation – Ethnography

- Ethnography n.人种学/民族志
- Ethnography is an observational technique that can be used to understand operational processes and help derive requirements for software to support processes.
- An analyst immerses himself/herself in the working environment where the system will be used.
- The day-to-day work is observed, and notes are made of the actual tasks in which participants are involved.
- The value of ethnography is that it helps discover implicit system requirements that reflect the actual ways that people work.

Business analysts

- Need to be familiar with the problem domain and development techniques

- The bridge between the customers and the development team
  - To the customers:
    - Explain in domain language what can/cannot be achieved with existing constraints
    - Hide technical details when explaining the technical solution to the customers
    - Create user manual
  - To the development team:
   - Reformulate the domain problem as mathematical problems

Common Problems During Requirement Elicitation

- Problem of scope
  - What environmental condition the system will operate in?

- Problem of understanding

- Problem of volatility
  - User needs evolve over time

Problem of Understanding

- The customer fails to explain their needs well.

  – Need a common language

- The analyst may not understand the customer's need.

  – Need to study the problem domain

- The customer may not know what he/she wants

  – The team should identify customer needs from the problem domain

- The analyst may not clearly convey the requirements to the development team

  – Problem abstraction

Natural Languages Are Prone to Ambiguities
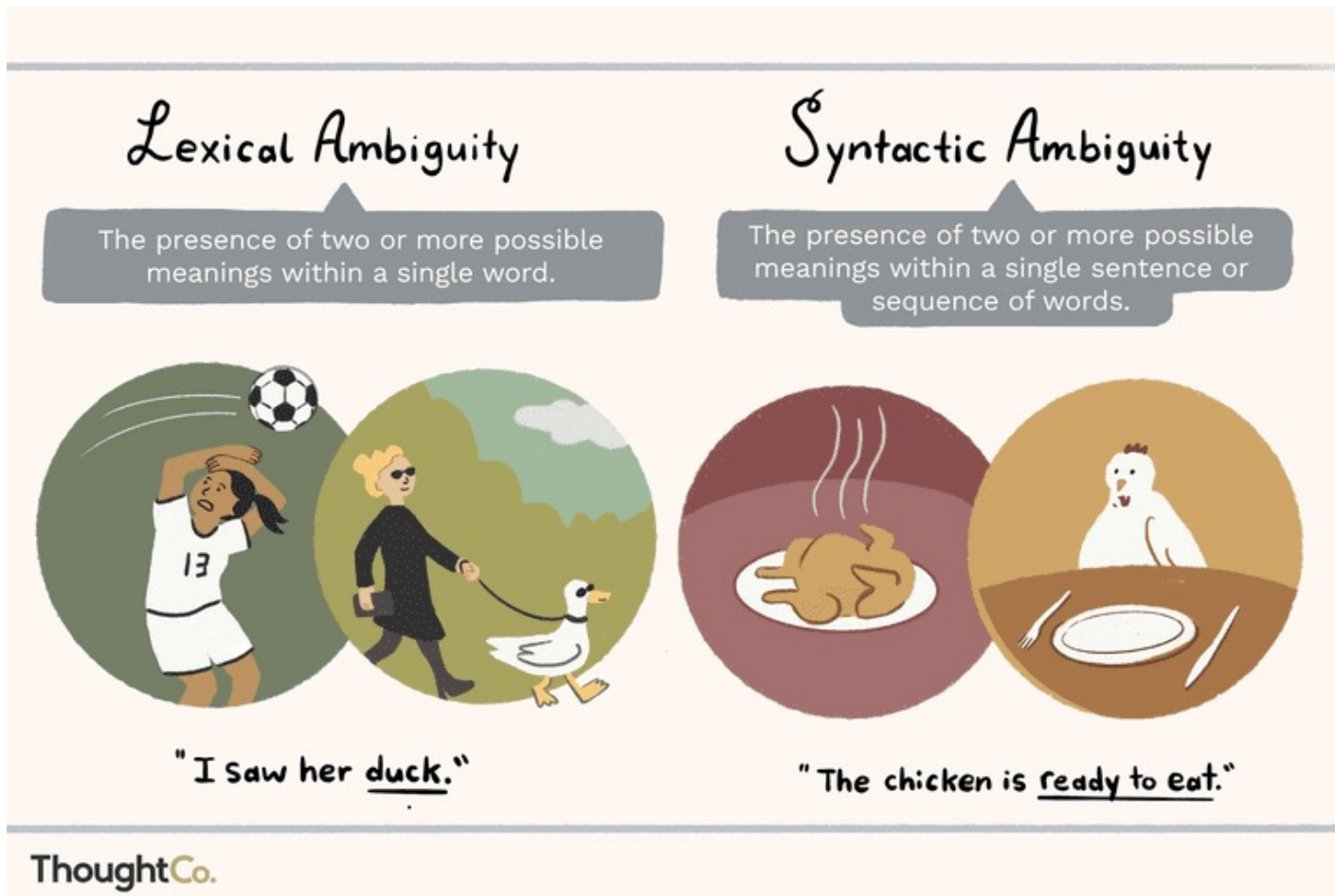
We need a widely used formal language

Communications among various stakeholders

- Need a common language for communication

- Unified Modeling Language (UML)

- Recognized as an international standard
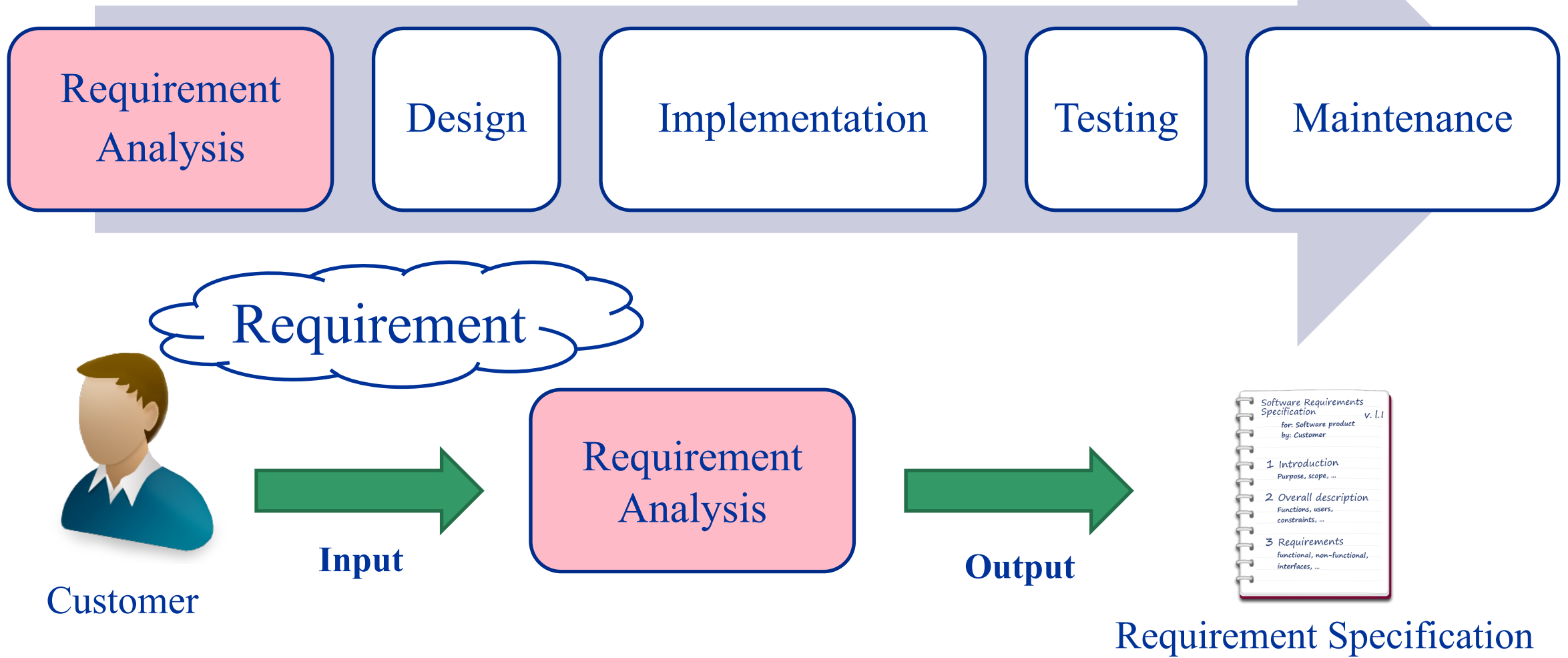
- It's just a tool, not a solution

Stories and Scenarios

- Scenarios and user stories are real-life examples of how a system can be used.

- Stories and scenarios are a description of how a system may be used for a particular task.

- Because they are based on a practical situation, stakeholders can relate to them and can comment on their situation with respect to the story.

Stories and Scenarios

- A structured form of user story

- Scenarios should include
    - A description of the starting situation;
    - A description of the normal flow of events;
    - A description of what can go wrong;
    - Information about other concurrent activities;
    - A description of the state when the scenario finishes

Software Development Life Cycle (2)

Requirement Analysis	Design	Implementation	Testing	Maintenance

Requirement

Customer → **Input** → Requirement Analysis → **Output** → Requirement Specification

Software Requirements Specification v. l.l
for: Software product
by: Customer

1 Introduction
Purpose, scope, ...

2 Overall description
Functions, users, constraints, ...

3 Requirements
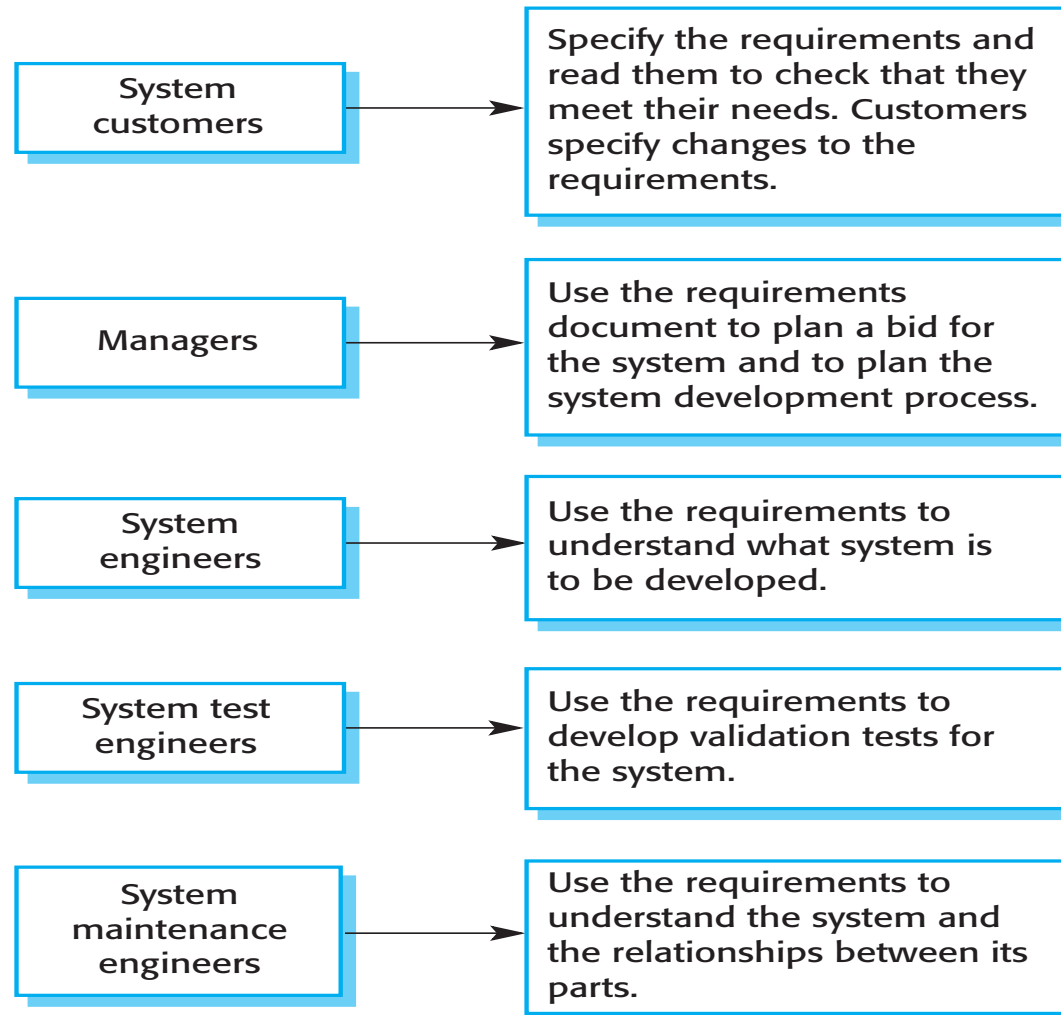functional, non-functional, interfaces, ...

Requirement Specification

- The software requirements document is the official statement of what is required of the system developers.

- Should include both a definition of user requirements and a specification of the system 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.

Guidelines for Writing Requirement Specification

- 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.

- Include an explanation (rationale) of why a requirement is necessary.

Use of Requirement Specification

System customers	Specify the requirements and read them to check that they meet their needs. Customers specify changes to the requirements.
Managers	Use the requirements document to plan a bid for the system and to plan the system development process.
System engineers	Use the requirements to understand what system is to be developed.
System test engineers	Use the requirements to develop validation tests for the system.
System maintenance engineers	Use the requirements to understand the system and the relationships between its parts.

Reference

- 830-1998 - IEEE Recommended Practice for Software Requirements Specifications
- https://ieeexplore.ieee.org/document/720574