SOFTWARE PROCESS MODELING

Requirements Engineering

Session Outcomes

- Why we need Requirements Engineering
- Requirement levels and types
- Requirements Engineering process
  - Requirements elicitation and analysis
  - Requirements Specification
  - Requirements Validation

Requirements Engineering

- Addresses two main problems
  - What do we want to build?
  - How do we write this down?

Library Management System

- What do you have to build?
- What do you want to build?
- How does it differ from the existing system?

What is RE?

- Requirements engineering is a process of establishing
  - the functions and attributes
    - that a customer requires from a system
  - the constraints
    - under which it operates and is developed.

Why RE?

- Trouble in understanding what customer really wants
- Record requirements in a disorganized manner
- Spend far too little time verifying what we do record
- Fail to establish a solid foundation for the system or software that the user wants built

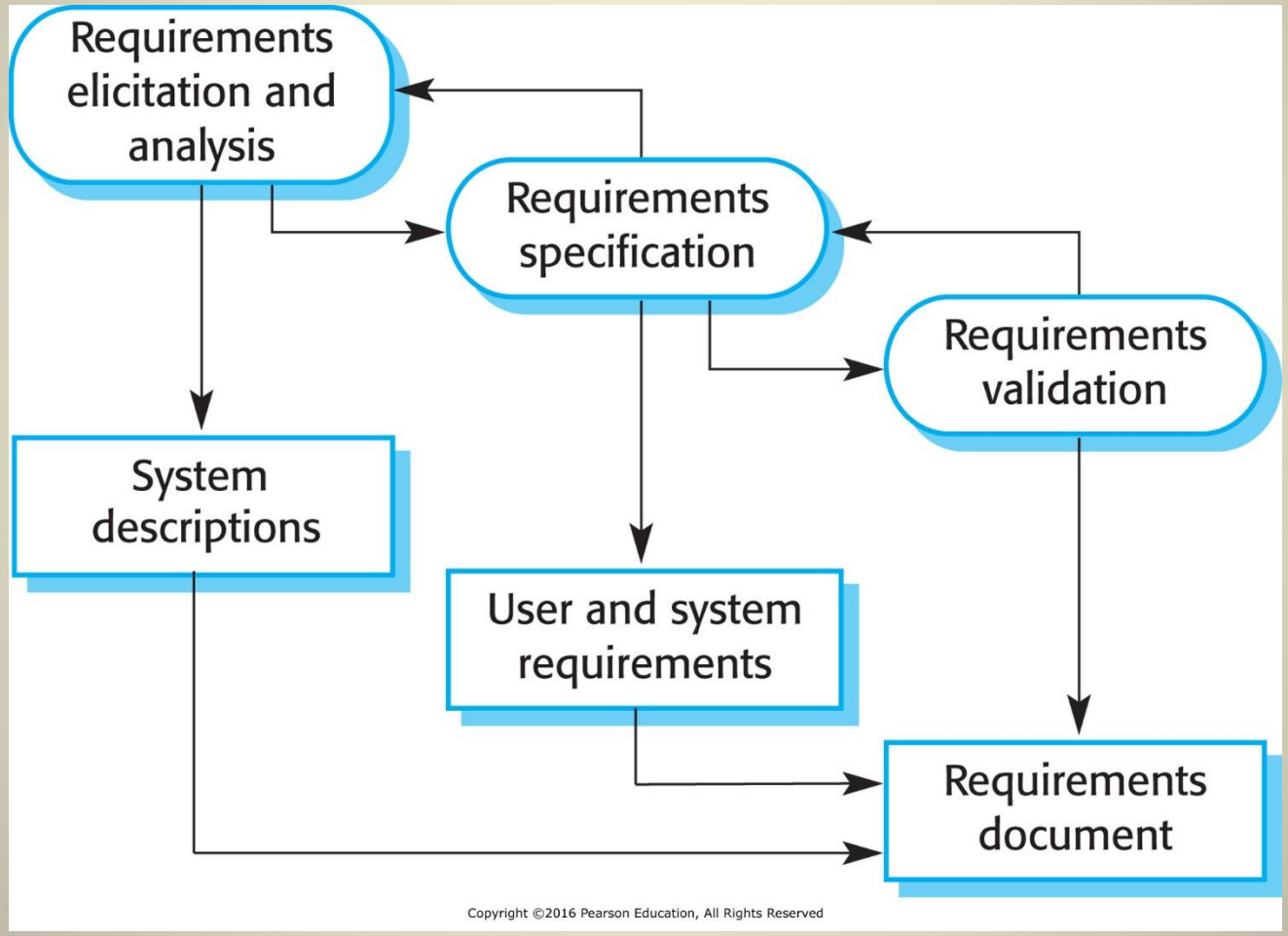
System stakeholders

- 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

Activity

- Identify Stakeholders in the Library Management System

Requirements Engineering process



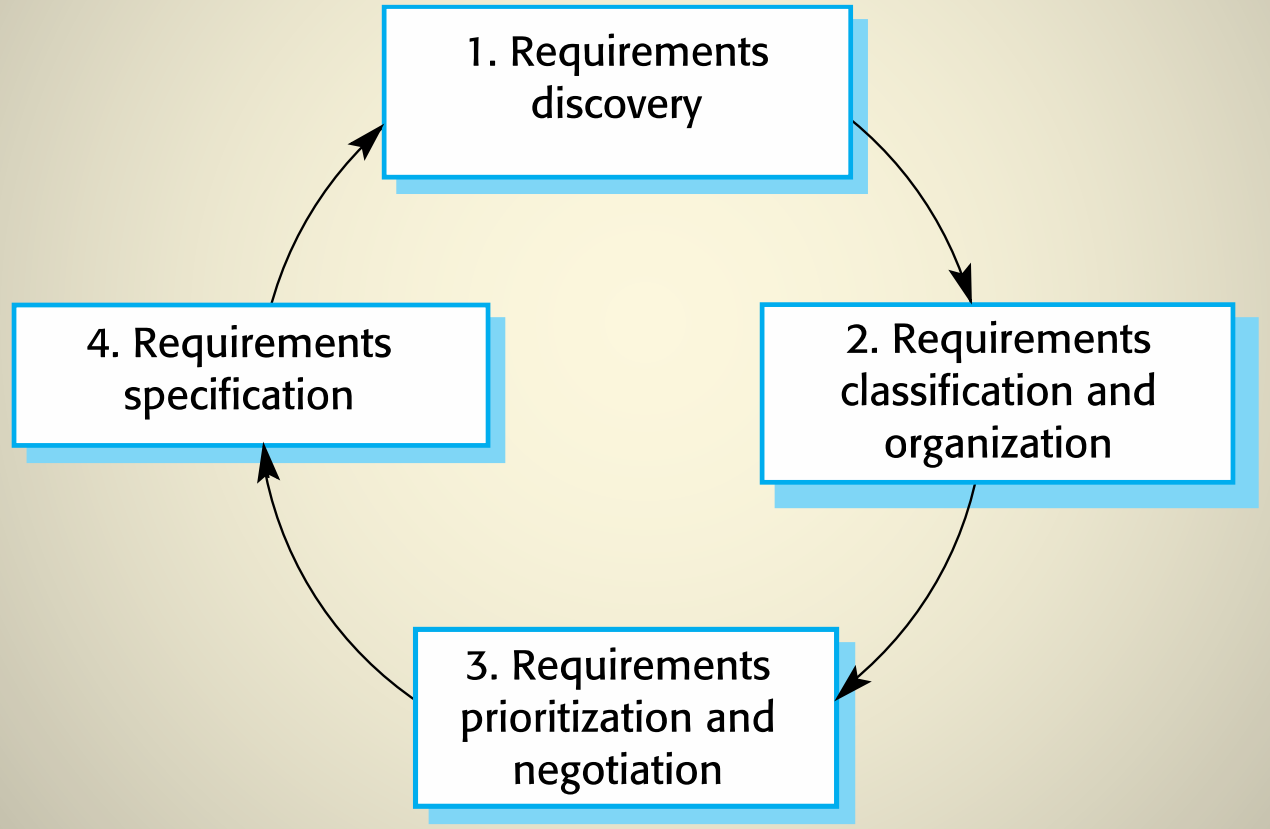
Requirements elicitation and analysis

Requirements elicitation

- Aim – Understand the work stakeholders do and how a new system would support that work.
- Stages include:
  - Requirements discovery,
  - Requirements classification and organization,
  - Requirements prioritization and negotiation,
  - Requirements specification.

Ref: Software Engineering, I. Sommerville, 10th Edition

The requirements elicitation and analysis process



Ref: Software Engineering, I. Sommerville, 10th Edition

Requirements Elicitation and Analysis

1. Requirements discovery
   - Interacting with stakeholders to discover their requirements.
2. Requirements classification and organisation
   - Groups related requirements and organises them into coherent clusters.
3. Prioritisation and negotiation
   - Prioritising requirements and resolving requirements conflicts.
4. Requirements specification
   - Requirements are documented and input into the next round.

Ref: Software Engineering, I. Sommerville, 10th Edition

1. Requirements discovery

- The process of gathering information about the required and existing systems and distilling the user and system requirements from this information.
- Requirements elicitation techniques
  - Interviewing
    - Closed or Open
  - Observation/Ethnography
    - Observing and analysing how people actually work.

Ref: Software Engineering, I. Sommerville, 10th Edition

"I'll go talk to the stakeholders and find out their requirements... in the meantime, you guys start coding."

Activity

- Visit a library where a Library System is not used. Observe their behavior.
  - Note the differences in the case study and actual system.
  - Note down the additional requirements you identified

2. Requirements Classification

- 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
  - Requirements that are not directly concerned with specific functionality
- Constraints

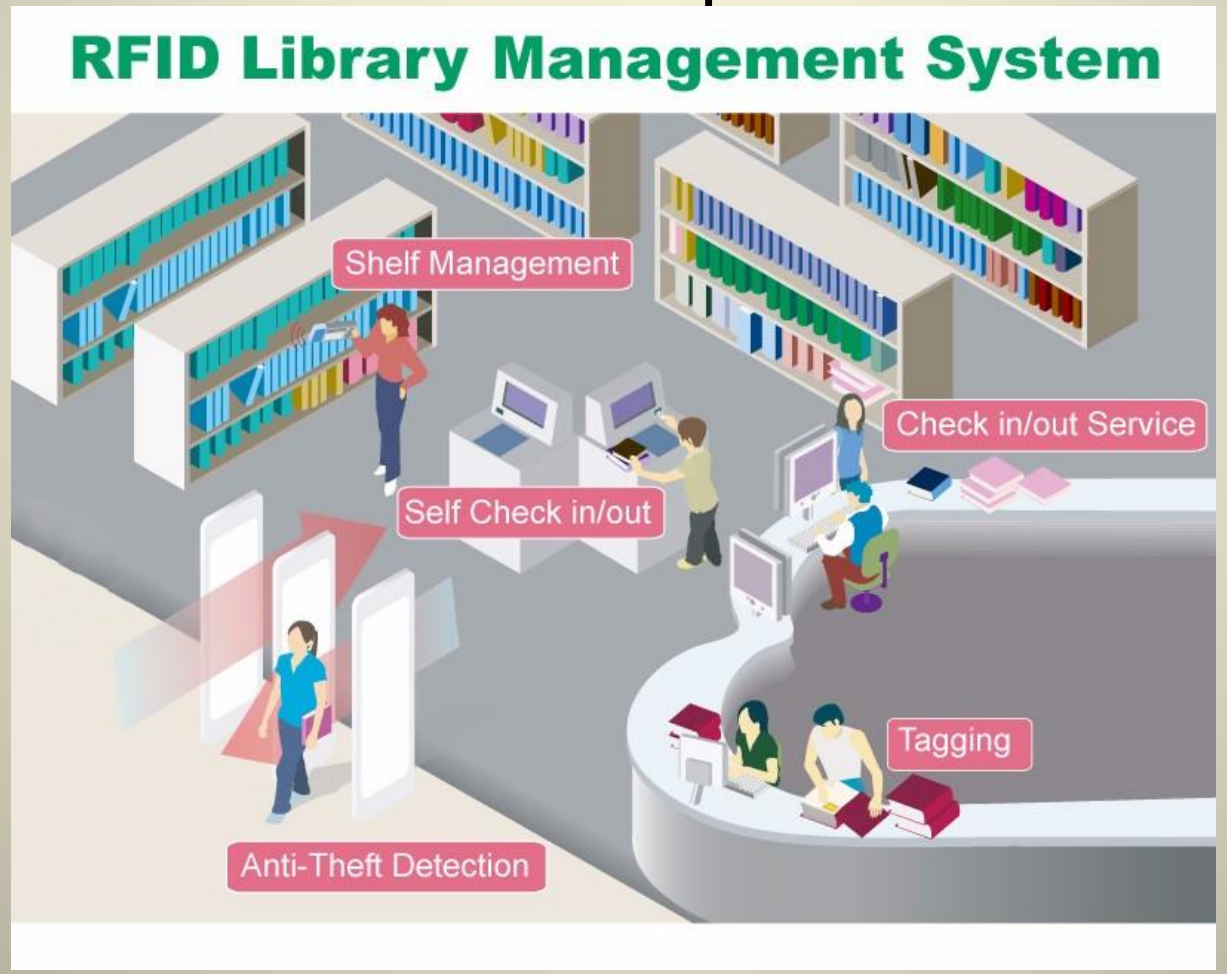
2.1 Functional requirements

- 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

Ref: Software Engineering, I. Sommerville, 10th Edition

Activity

- Write down two Functional Requirements for the Library System



2.2 Non-functional requirements

- Often apply to the system as a whole rather than individual features or services.
- These define system properties
- Are also called Quality Attributes
- Non-functional requirements may be more critical than functional requirements. If these are not met, the system may be useless.

Ref: Software Engineering, I. Sommerville, 10th Edition

Non-functional - examples

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

Activity

- Write down three non-functional Requirements for the Library System
- Choose the most important among them

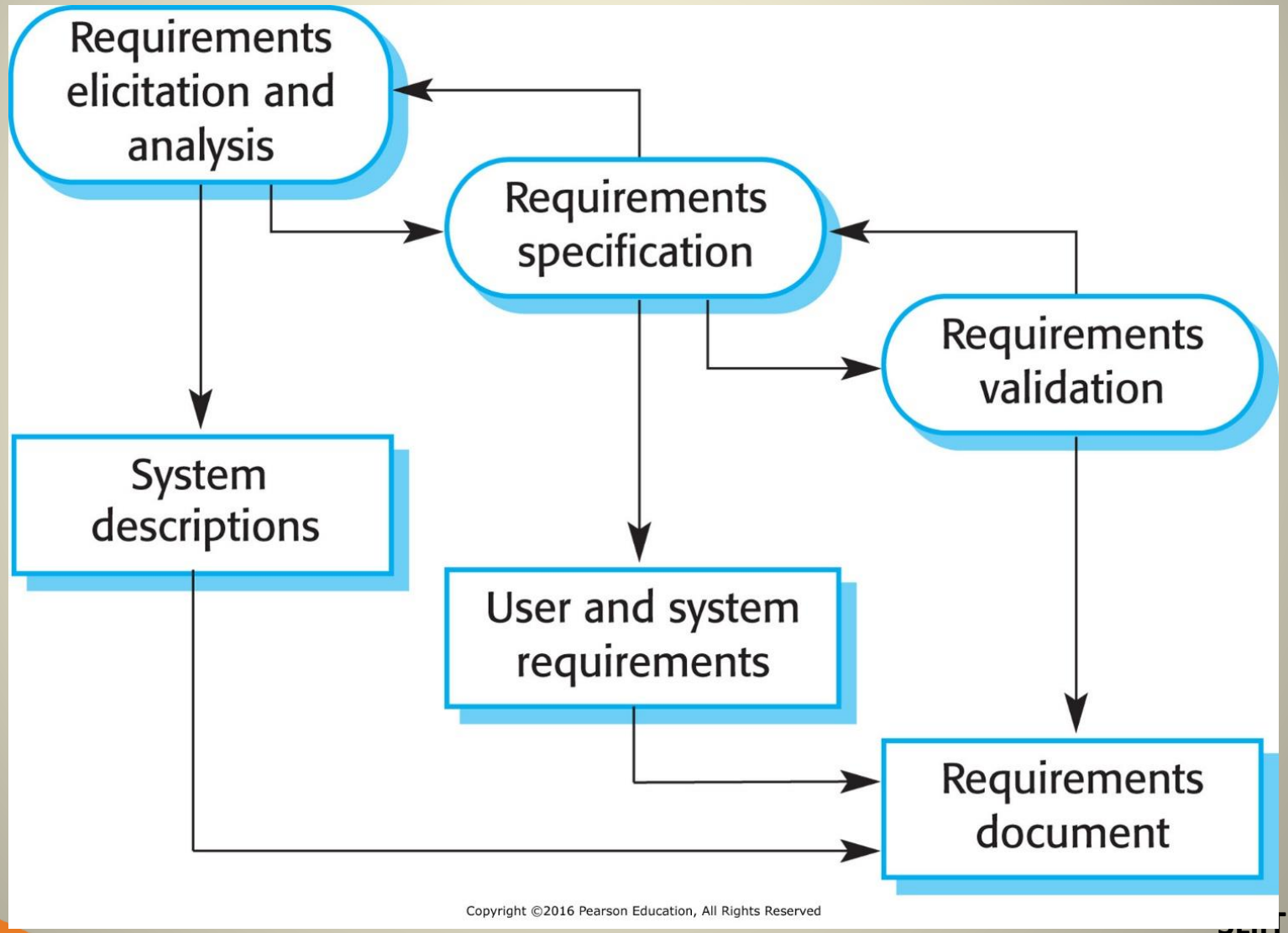
Requirements imprecision

- inconsistent Requirements
  - –the customer cannot decide what problem they really want solved.
- ambiguous Requirements
  - –it is not possible to determine what the requirements mean.
- Incomplete Requirements
  - –there is insufficient information to allow a system to be built.

Activity

- Write down two imprecise Requirements for the Library System.

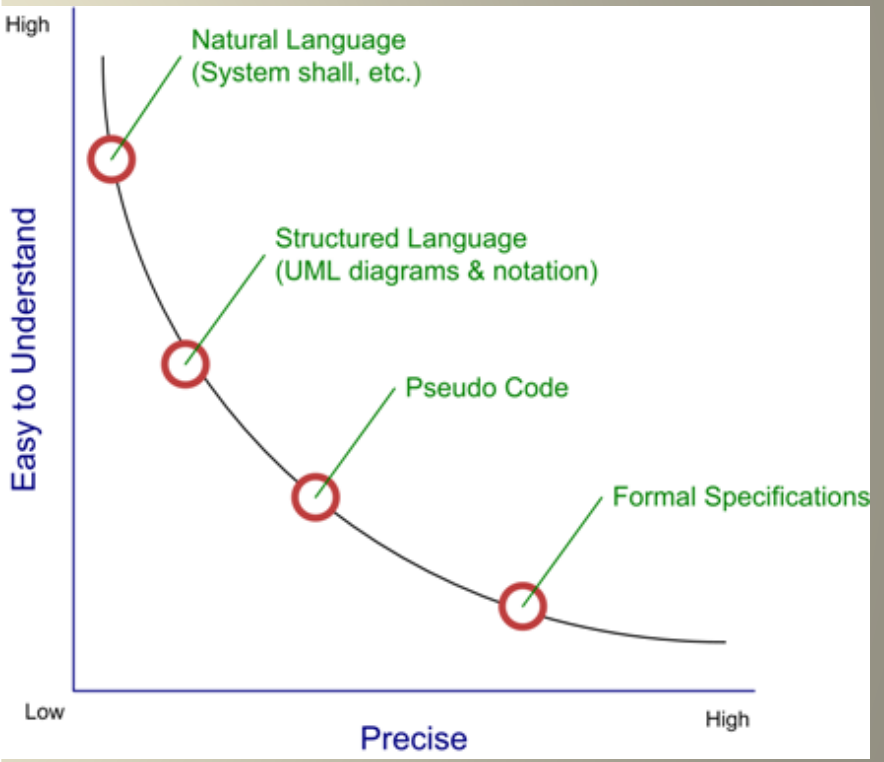
Requirements Engineering process



Requirements Specification

System requirements specification

Notation	Description
Natural language	The requirements are written using numbered sentences in natural language. Each sentence should express one requirement.
Structured natural language	The requirements are written in natural language on a standard form or template. Each field provides information about an aspect of the requirement.
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. This approach is now rarely used although it can be useful for interface specifications.
Graphical notations	Graphical models, supplemented by text annotations, are used to define the functional requirements for the system; UML use case and sequence diagrams are commonly used.
Mathematical specifications	These notations are based on mathematical concepts such as finite-state machines or sets. Although these unambiguous specifications can reduce the ambiguity in a requirements document, most customers don't understand a formal specification. They cannot check that it represents what they want and are reluctant to accept it as a system contract



Ref: Software Engineering, I. Sommerville, 10th Edition

Graphical Notations

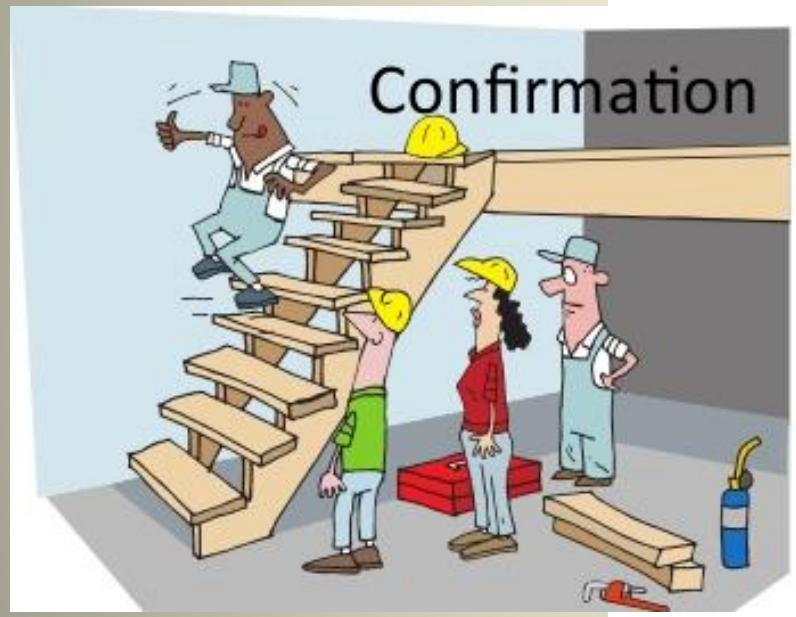
- UML
  - Use case Diagrams
  - Activity Diagrams
- User Stories
  - https://www.youtube.com/watch?v=LGeDZmrWwsw (Agile)
  - https://www.youtube.com/watch?v=6q5-cVeNjCE (Agile)
  - https://youtu.be/502ILHjX9EE(*Agile)
  - http://www.agileacademy.com.au/agile/knowledgehub


User
Story

A User Story

Is a concise description of a functionality that will be valuable to a user of a system.

3 C's



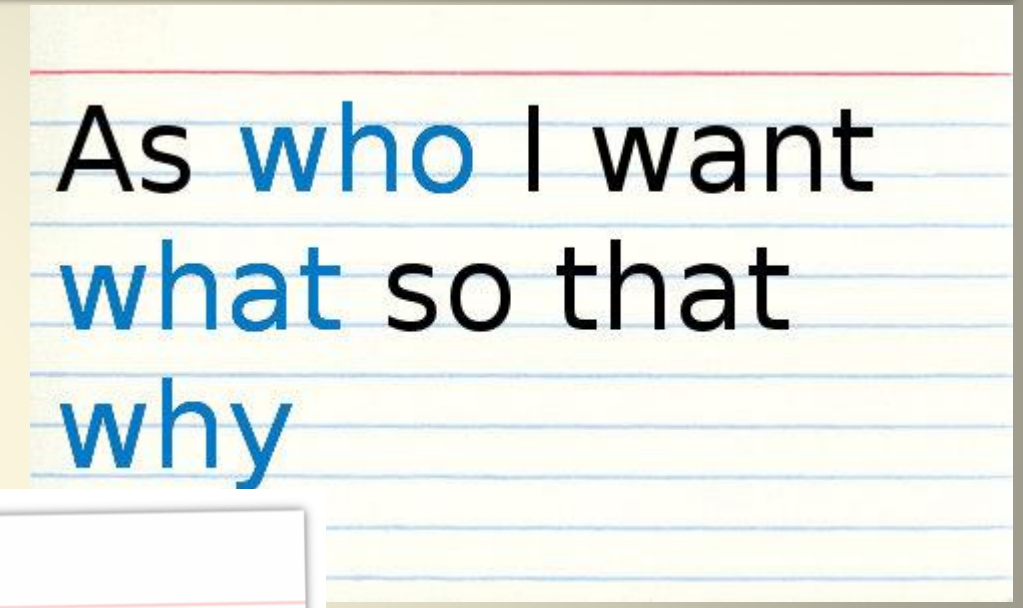
Confirmation

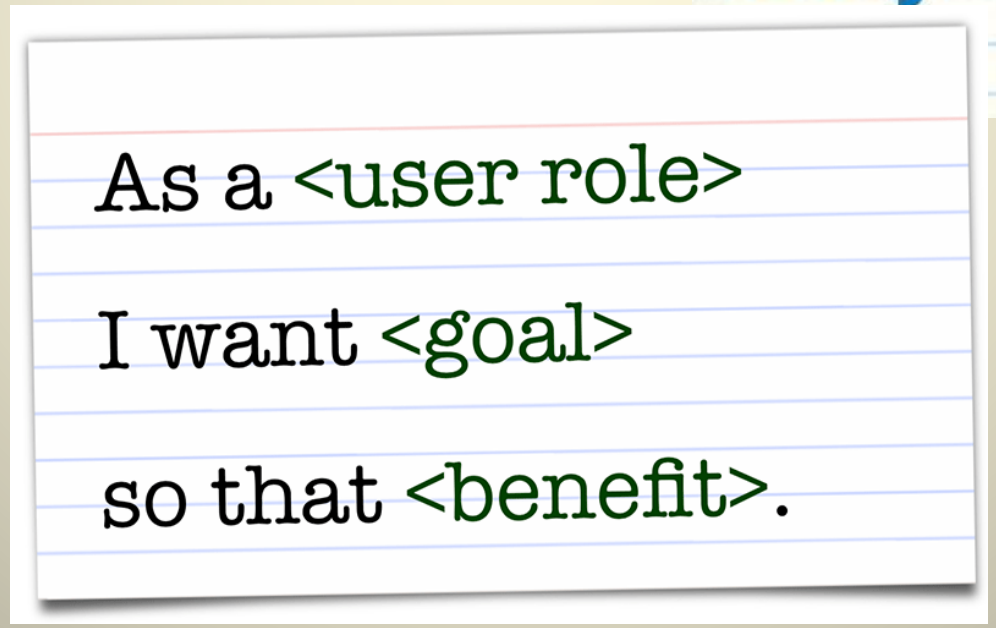
Card

Conversation

Card

- Size
- Format





Example – Online Banking System

- As a Customer I want to view account summary online so that I do not have to wait till the month end to view the statement.
- As an Employee I want to add new customers online so that it saves my time.
- As a User I want to update profile details so that my details are up-to-date

Activity

- Write two user stories for the Library System
  - Be creative
  - Think about the role

Compare

A	B
As a recruiter I want to review resumes from applicants to one of her ads.	As a recruiter I want to manage the ads she has placed.

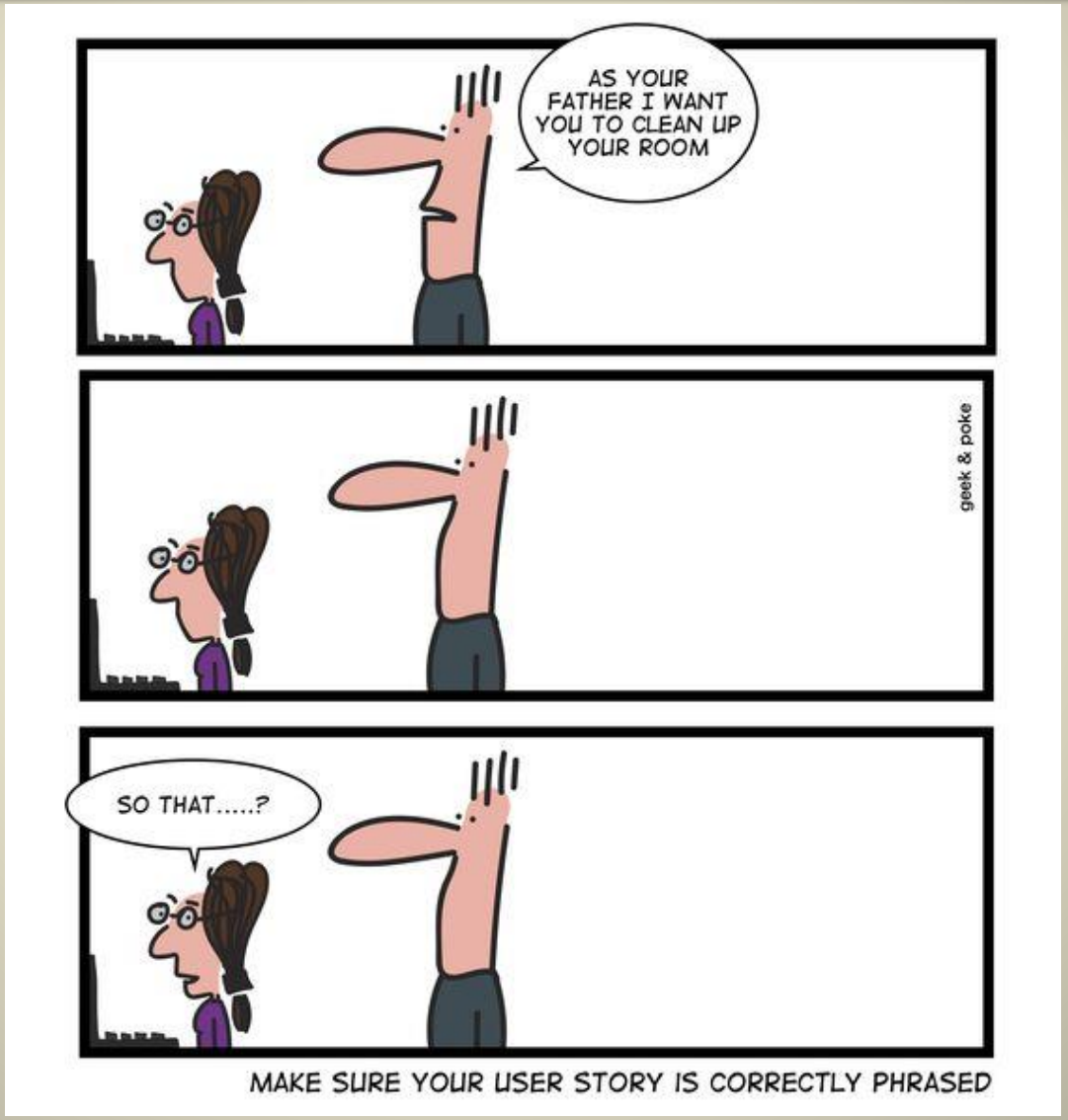
Compare

A	B
As a driver I want to find the store with the shortest drive time so I can get there quickly.	As a driver I want to find directions to a store in Google Maps so I can get there quickly.

Compare

A	B
As a user I want to have my previous orders stored in the database so they will be there permanently	As a repeat customer I want to access old orders so that I can quickly purchase the same order again.

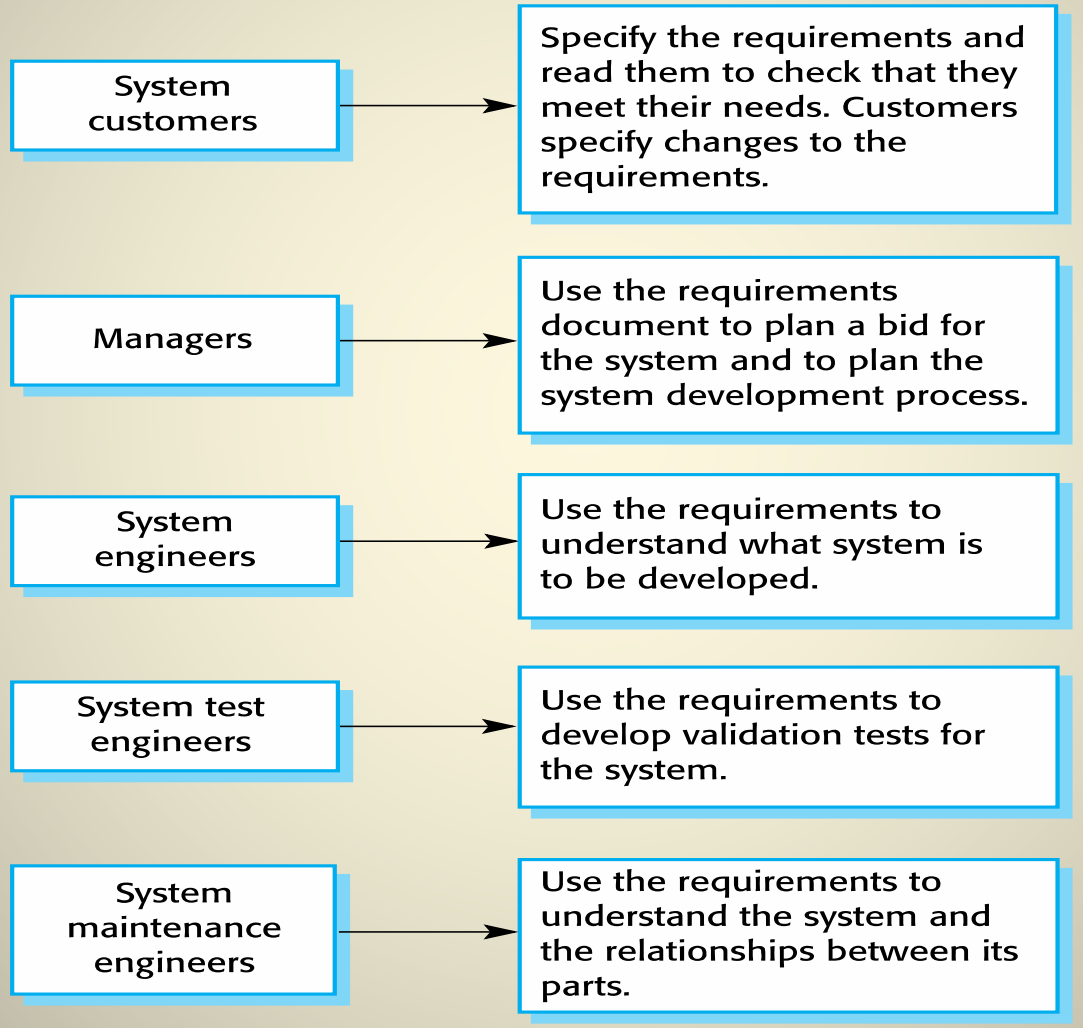
Everyday User Stories



SRS

- Software Requirements 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.

Users of SRS



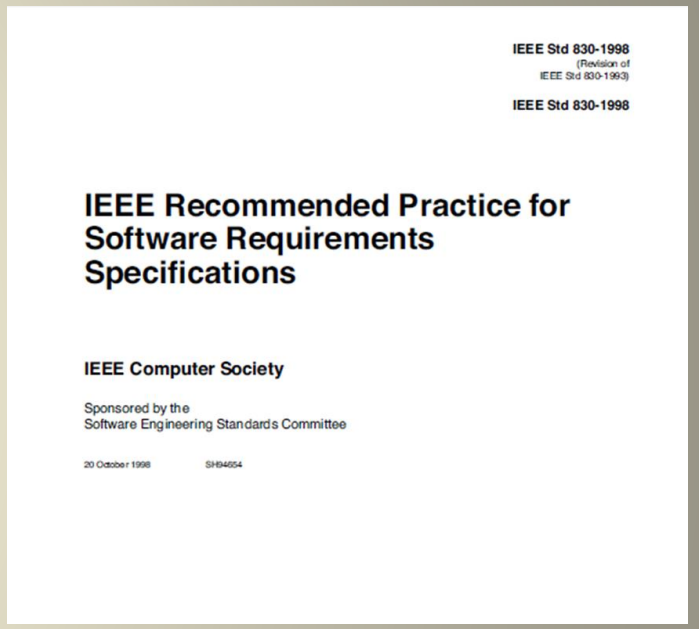
Ref: Software Engineering, I. Sommerville, 10th Edition

IEEE 830

IEEE_SRS_Template.pdf

Role of SRS

1. Should correctly define all of the software requirements.
2. Should not describe any design or implementation details.
3. Should not impose additional constraints on the software.


IEEE Std 830-1998
(Revision of
IEEE Std 830-1993)

IEEE Std 830-1998

IEEE Recommended Practice for Software Requirements Specifications

IEEE Computer Society

Sponsored by the
Software Engineering Standards Committee

20 October 1998 SH94654

SRS Template

- Introduction
  - Purpose, Scope , Overview
- General Description
  - Product Perspective, User Characteristics
- Specific Requirements
  - Functional Requirements
  - Non-functional
  - Constraints

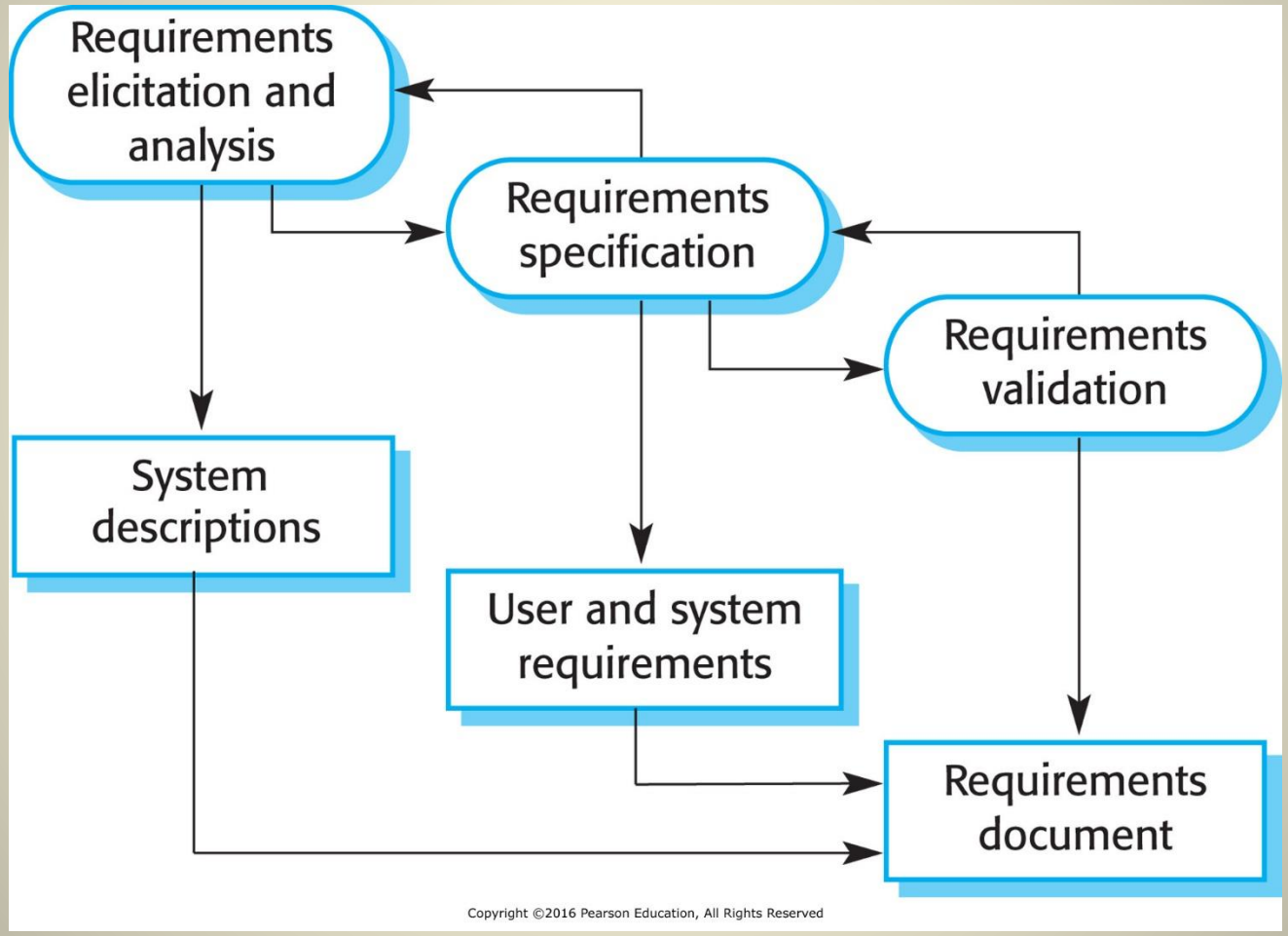
SRS for Library Management System

- Introduction
  - Library Management System (LMS) project aims to develop a computerized system to maintain all the daily work of the SLIIT library. This system will help the users to manage the library daily activities in electronic format.
- The General Description
  - LMS is an online system which would give SLIIT staff members as well as students easy access to the Library contents.

SRS for Library Management System

- Functional Requirements
  - Story cards, Use Case Diagram, Activity Diagrams
- Non-functional Requirements
- Constraints
  - Budget constraints
  - Technology Constraints

Requirements Engineering process



Requirements Validation

Requirements validation

- Validate whether the elicited requirements define the system that the customer really wants.
- Requirements error costs are high so validation is very important.
- Requirements checking
  - Validity
  - Precise requirements
  - Realism
  - Verifiability

Requirements validation techniques

- Requirements reviews
  - Systematic manual analysis of the requirements.
- Prototyping
  - Using an executable model of the system to check requirements.
- Test-case generation
  - Developing tests for requirements to check testability.
  - Will be discussed later

References

- Software Engineering – 10th Edition by Ian Sommerville, Chapter 4
- http://iansommerville.com/systems-software-and-technology/
- http://iansommerville.com/software-engineering-book/
- IEEE Recommended Practice for Software Requirements Specifications - IEEE Std 830-1998