Module 2

Requirements Analysis & Modelling

Domain Analysis

The process by which a software engineer learns about the domain to better understand the problem:

- The *domain* is the general field of business or technology in which the clients will use the software
- A *domain expert* is a person who has a deep knowledge of the domain

Benefits of performing domain analysis:

- Faster development
- Better system
- Anticipation of extensions

Domain Analysis document

A. **Introduction** (name and motivation)

B. **Glossary** (meaning of terms)

C. **General knowledge about the domain** (summary of rules or facts)

D. **Customers and users** (people who will use or buy software)

E. **The environment** (equipments and systems used)

F. **Tasks and procedures currently performed** (remove unnecessary details)

G. **Competing software** (other software in market)

H. **Similarities to other domains** (generic vs specific)

Defining the Problem and the Scope

A problem can be expressed as:

- A *difficulty* the users or customers are facing,
- Or as an *opportunity* that will result in some benefit such as improved productivity or sales.

The solution to the problem normally will entail developing software

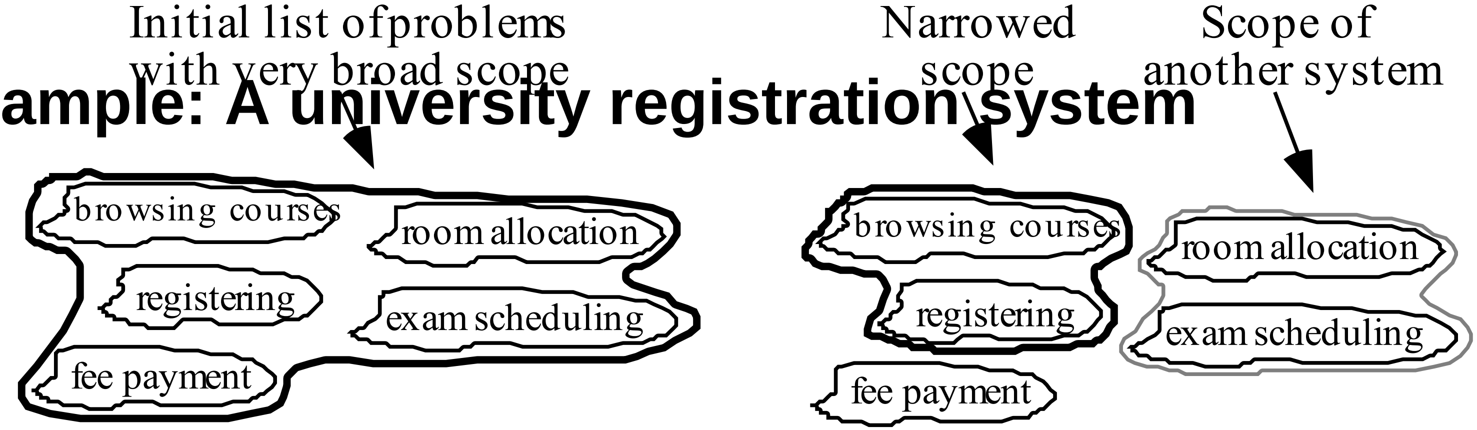
A good problem statement is short and succinct

Defining the Scope

Narrow the *scope* by defining a more precise problem

- List all the things you might imagine the system doing
  - —Exclude some of these things if too broad
  - —Determine high-level goals if too narrow

Example: A university registration system



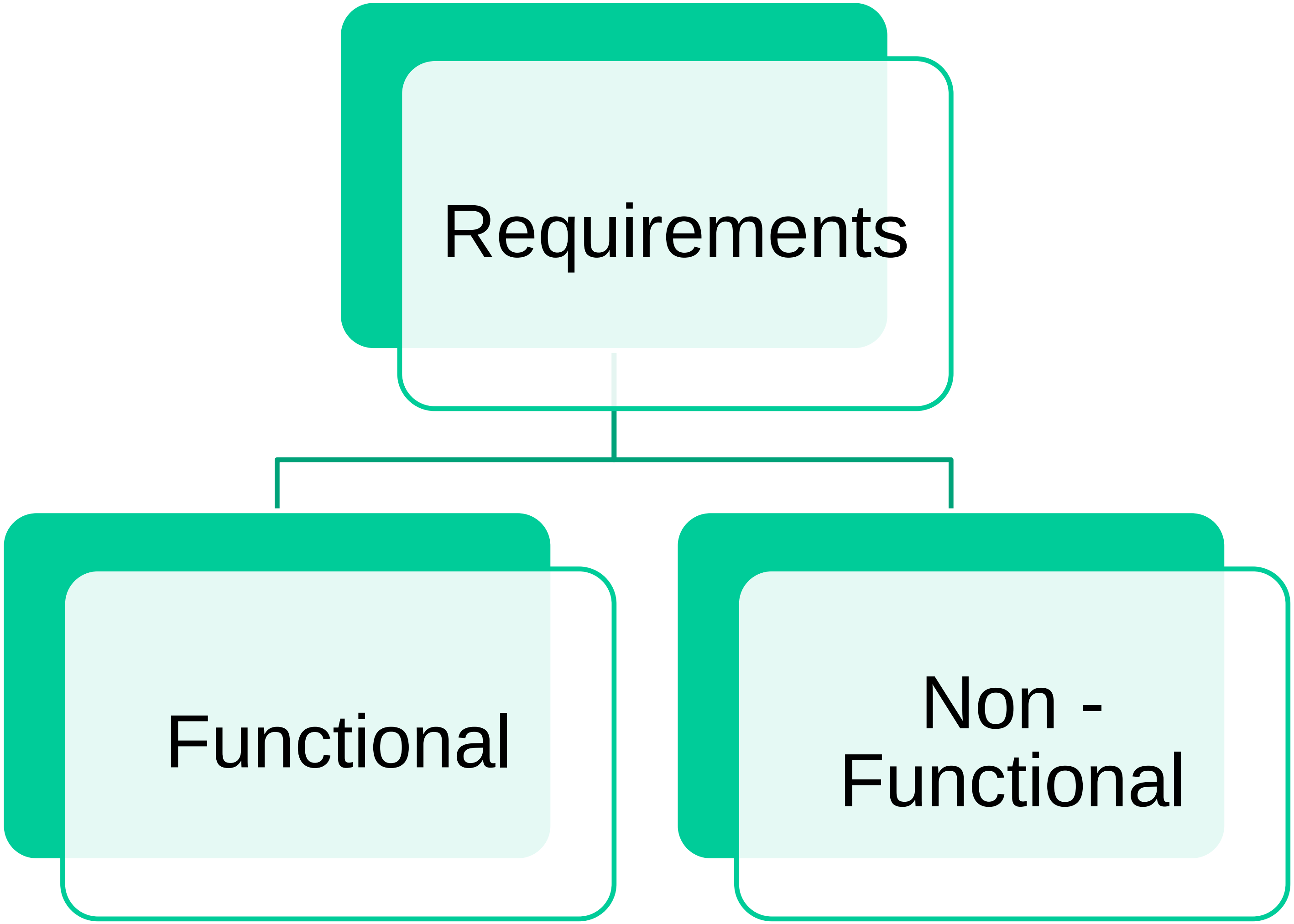
What is a Requirement ?

It is a statement describing either

- 1) an aspect of what the proposed system must do,
- or 2) a constraint on the system’s development.
- In either case it must contribute in some way towards adequately solving the customer’s problem;
- the set of requirements as a whole represents a negotiated agreement among the stakeholders.

A collection of requirements is a *requirements document*.

Types of Requirements



Requirements

Functional

Non - Functional

Types of Requirements –Functional

- Describe *what* the system should do

  - i.e. the services provided for the user and for other systems

- Should include

  - everything that a user of the system would need to know

  - everything that would concern any other system that has to interface to this system.

Functional Requirements -categories

- What *inputs* the system should accept (user commands and inputs)

- What *outputs* the system should produce (screen or printed form)

- What data the system should *store* that other systems might use (usually a database or backup)

- What *computations* the system should perform (e.g. Sorting process)

- The *timing and synchronization* of the above (generally for hard real –time systems)

Non –Functional Requirements

- Are **constraints** that must be adhered to during development.
- They limit the use of resources and set bounds on aspects of software quality.
- All requirements should be **verifiable.**

Non- Functional Requirements

Quality requirements

- C*onstraints* on the design to meet specified levels of quality

Platform requirements

- C*onstraints* on the environment and technology of the system

Process requirements

- C*onstraints* on the project plan and development methods

Quality Requirementts

All must be verifiable

Examples: Constraints on

- Response time (Google Search)
- Throughput (No. of transactions per minute)
- Resource usage (RAM, bandwidth, CPU time)
- Reliability (avg. amount of time between failures or probability of a failure in a given time)
- Availability (amount of time that the server is up and running)
- Recovery from failure (maximum allowed impact of a failure)
- Allowances for maintainability and enhancement
- Allowances for reusability

Non – Functional 2nd Category

- Platform (differs from resource usage)

  - what OS and hardware the software must to able work on.

  - e.g. Ms Windows 2003, 512 MB RAM, 1GB free disk space.

- Technology to be used

  - programming language

  - database system

  - same technology should be used for all the system to reduce training costs

Non – Functional 3rd Category

- Development process (methodology) to be used

  - certain processes should be used, testing approach to be used.

  - process model to be applied.

- Cost and delivery date

  - Generally specified in project plan document.

Classify into F or NF or X for Airline reservation system

- how information about flights, passengers and bookings are entered.
- What information appears on tickets and reports.
- How fares are calculated.
- What information should be stored in DB that travel agents and others access.
- System should be designed so it can be extended to handle a frequent flier plan.
- System must be available at all times.only 2 mins downtime a week is allowed.
- A merge sort algorithm must be used to sort the flights by departure time.

Questions..

- **What is requirement? Explain its types. [M-2011][D-2011]**
- **For hospital management system, write functional and non-functional requirements [M-2012]**
- **What do you mean by requirements? Explain functional and non- functional requirements in detail [D-2012]**

Some Techniques for Gathering and Analysing Requirements (Requirements Elicitation)

Observation

- Read documents and discuss requirements with users
- Shadowing important potential users as they do their work
  —ask the user to explain everything he or she is doing
- Session videotaping

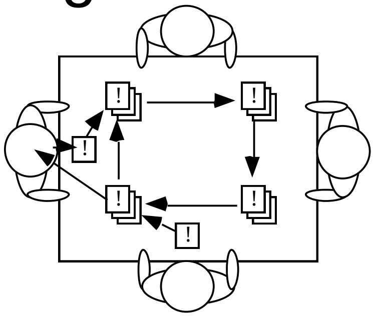
Interviewing

- Conduct a series of interviews
  —Ask about specific details
  —Ask about the stakeholder’s vision for the future
  —Ask if they have alternative ideas
  —Ask for other sources of information
  —Ask them to draw diagrams

Gathering and Analysing Requirements...

Brainstorming

- Appoint an experienced moderator
- Arrange the attendees around a table
- Decide on a ‘trigger question’
- Ask each participant to write an answer and pass the paper to its neighbour



Joint Application Development (JAD) is a technique based on intensive brainstorming sessions

Gathering and Analysing Requirements...

Prototyping

- The simplest kind: *paper prototype*.
  - —a set of pictures of the system that are shown to users in sequence to explain what would happen
- The most common: a mock-up of the system's UI
  - —Written in a rapid prototyping language
  - —Does *not* normally perform any computations, access any databases or interact with any other systems
  - —May prototype a particular aspect of the system

Gathering and Analysing Requirements...

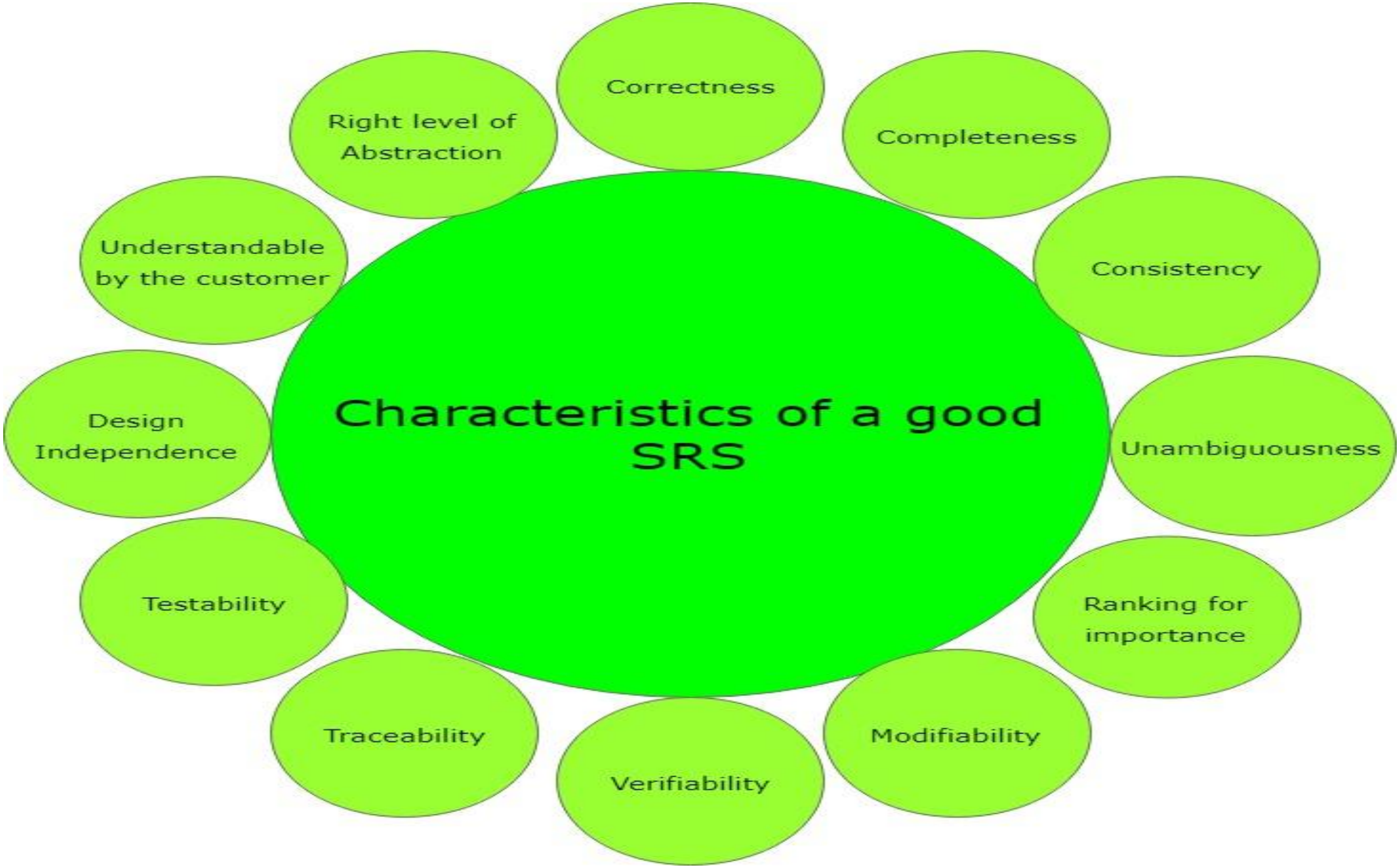
Use case analysis

- Determine the classes of users that will use the facilities of this system (actors)
- Determine the tasks that each actor will need to do with the system

Software Requirement Specification (SRS)

- SRS is a document created by system analyst after the requirements are collected from various stakeholders.
- SRS defines how the intended software will interact with hardware, external interfaces, speed of operation, response time of system, portability of software across various platforms, maintainability, speed of recovery after crashing, Security, Quality, Limitations etc.
- The requirements received from client are written in natural language.
- It is the responsibility of system analyst to document the requirements in technical language so that they can be comprehended and useful by the software development team.

Features of a Good SRS


Characteristics of a good SRS
Correctness
Completeness
Consistency
Unambiguousness
Ranking for importance
Modifiability
Verifiability
Traceability
Testability
Design Independence
Understandable by the customer
Right level of Abstraction