Requirement Elicitation

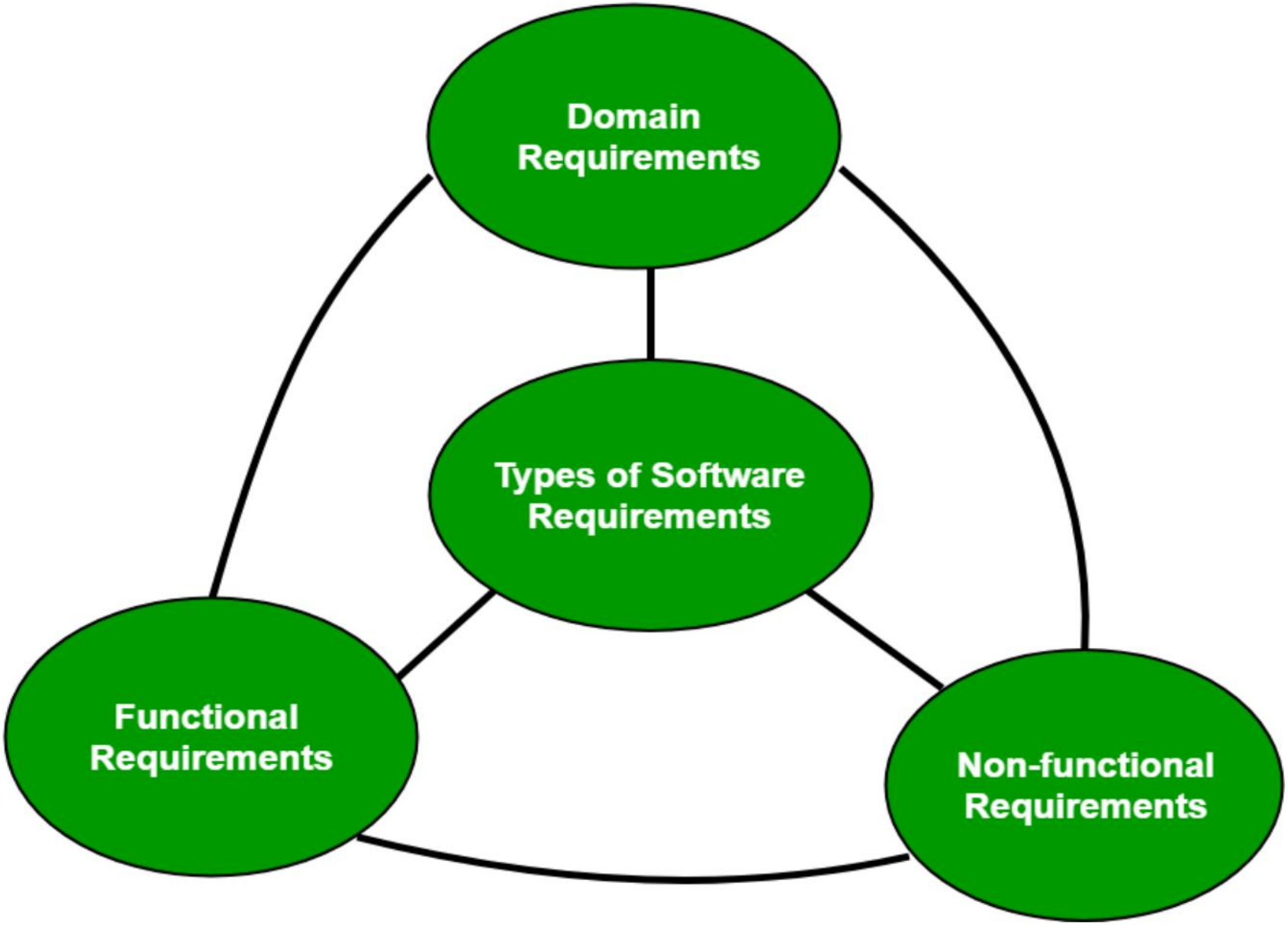
Lecture 3

Introduction

- Deciding precisely what to build is most important and most difficult
- Requirements are often buried under layers of assumptions, misconceptions, and politics
- Thorough understanding and constant communication with customers are essential

Classification

- **A software requirement can be of 3 types:**
- Functional requirements
- Non-functional requirements
- Domain requirements


Domain Requirements
Types of Software Requirements
Functional Requirements
Non-functional Requirements

Functional Requirements:

- These are the requirements that the end user specifically demands as basic facilities that the system should offer.
- All these functionalities need to be necessarily incorporated into the system as a part of the contract.
- These are represented or stated in the form of input to be given to the system, the operation performed and the output expected.
- They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

Non-functional requirements:

- These are basically the quality constraints that the system must satisfy according to the project contract.
- The priority or extent to which these factors are implemented varies from one project to other.
- They are also called non-behavioral requirements.

NFR

- They basically deal with issues like:
- Portability
- Security
- Maintainability
- Reliability
- Scalability
- Performance
- Reusability
- Flexibility

Domain requirements:

- Domain requirements are the requirements which are characteristic of a particular category or domain of projects.
- The basic functions that a system of a specific domain must necessarily exhibit come under this category.

Requirements Engineering

- Development, specification, and validation of requirements
- Elicitation and modeling
- Elicitation
  - Fact-finding, communication, and fact-validation
  - Output: requirements document Understood by customers unambiguously
- Modeling (based on requirements document)
- Requirements in a form understood by software engineers unambiguously

Requirements Elicitation

- The requirements elicitation process may appear simple: ask the customer, the users and others what the objectives for the system or product are, what is to be accomplished, how the system or product fits into the needs of business, and finally, how the system or product is to be used on a day-to-day basis. However, issues may arise that complicate the process. [1]

Challenges in Requirements Elicitation

- **Problems of scope'**. The boundary of the system is ill-defined or the customers/users specify unnecessary technical details that may confuse, rather than clarify, overall system objectives.
- **Problems of understanding**. The customers/users are not completely sure of what is needed, have a poor understanding of the capabilities and limitations of their computing environment, don't have a full understanding of the problem domain, have trouble communicating needs to the system engineer, omit information that is believed to be "**obvious**," specify requirements that conflict with the needs of other customers/users, or specify requirements that are ambiguous
- **Problems of volatility**. The requirements change over time. The rate of change is sometimes referred to as the level of requirement volatility

Requirements Quality

- **Visualization**. Using tools that promote better understanding of the desired end-product such as visualization and simulation.
- **Consistent language**. Using simple, consistent definitions for requirements described in natural language and use the business terminology that is prevalent in the enterprise.
- **Guidelines**. Following organizational guidelines that describe the collection techniques and the types of requirements to be collected. These guidelines are then used consistently across projects.
- **Consistent use of templates**. Producing a consistent set of models and templates to document the requirements.
- **Documenting dependencies**. Documenting dependencies and interrelationships among requirements.
- **Analysis of changes**. Performing root cause analysis of changes to requirements and making corrective actions.

Steps involved in Elicitation

- Identifying stakeholders
- Modeling goals
- Modeling context
- Discovering scenarios (or Use cases)
- Discovering "qualities and constraints" (Non-functional requirements)
- Modeling rationale and assumptions
- Writing definitions of terms
- Analyzing measurements (acceptance criteria)
- Analyzing priorities

Elicitation Techniques

- Interviews
- Brainstorming Sessions
- Facilitated Application Specification Technique (FAST)
- Quality Function Deployment (QFD)
- Use Case Approach

Interviews:

- Objective of conducting an interview is to understand the customer's expectations from the software.
- It is impossible to interview every stakeholder hence representatives from groups are selected based on their expertise and credibility.

Brainstorming Sessions:

- It is a group technique
- It is intended to generate lots of new ideas hence providing a platform to share views
- A highly trained facilitator is required to handle group bias and group conflicts.
- Every idea is documented so that everyone can see it.
- Finally a document is prepared which consists of the list of requirements and their priority if possible.

Facilitated Application Specification Technique:

- Each attendee is asked to make a list of objects that are-
  - Part of the environment that surrounds the system
  - Produced by the system
  - Used by the system
- Each participant prepares his/her list, different lists are then combined, redundant entries are eliminated, team is divided into smaller sub-teams to develop mini-specifications and finally a draft of specifications is written down using all the inputs from the meeting.

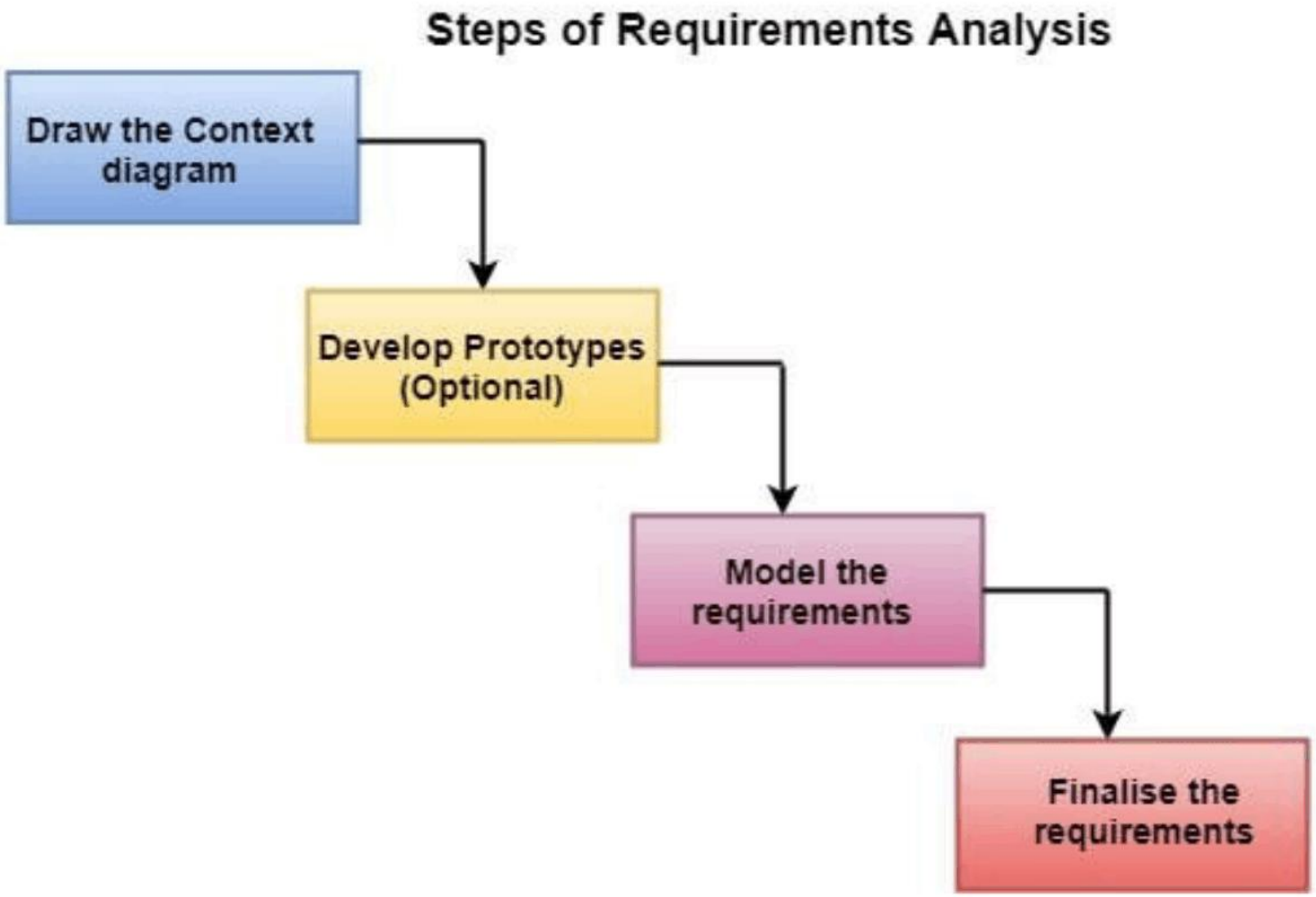
Quality Function Deployment:

- The major steps involved in this procedure are –
- Identify all the stakeholders, eg. Users, developers, customers etc
- List out all requirements from customer.
- A value indicating degree of importance is assigned to each requirement.
- In the end the final list of requirements is categorised as –
  - It is possible to achieve
  - It should be deferred and the reason for it
  - It is impossible to achieve and should be dropped off

Use Case Approach:

- **Actor –** It is the external agent that lies outside the system but interacts with it in some way. An actor maybe a person, machine etc. It is represented as a stick figure. Actors can be primary actors or secondary actors.
  - Primary actors – It requires assistance from the system to achieve a goal.
  - Secondary actor – It is an actor from which the system needs assistance.
- **Use cases –** They describe the sequence of interactions between actors and the system. They capture who(actors) do what(interaction) with the system. A complete set of use cases specifies all possible ways to use the system.
- **Use case diagram –** A use case diiagram graphically represents what happens when an actor interacts with a system. It captures the functional aspect of the system.
  - A stick figure is used to represent an actor.
  - An oval is used to represent a use case.
  - A line is used to represent a relationship between an actor and a use case.

Requirements Analysis



SRS

- The software requirements specification document lists sufficient and necessary requirements for the project development.
- To derive the requirements, the developer needs to have clear and thorough understanding of the products under development.
- This is achieved through detailed and continuous communications with the project team and customer throughout the software development process

References

- [1] https://en.wikipedia.org/wiki/Requirements_elicitation
- [2] https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=278253
- [3] http://www.cse.msu.edu/~chengb/RE-491/Papers/SRSExample-webapp.doc