Requirement Engineering & Modelling [week-6 theory]

Overview

- The process of collecting the software requirement from the client then understand, evaluate and document it is called as requirement engineering.
- Requirement engineering constructs a bridge for design and construction.

What is requirement?

The requirements are the high-level descriptions about a particular system service, constraints or to a detailed specification that are generated during the requirements gathering process.

The goal of requirement engineering is to develop and maintain sophisticated and descriptive "System Requirement Specification" document.

Importance

They establish a foundation for product vision, scope, cost, and schedule and they ultimately must target finished product quality and performance.

Requirement types

1. **User Requirements -** It is a detailed description in natural language along with diagrams of the services the system provides and its operational constraints. It is usually developed by end users.

2. **System requirements -** It is a structured document detailing the descriptions of the system's functions, services and operational constraints.

3. **Functional Requirements -** It describes the services of the system, how the system should react to particular inputs and how the system should behave in definite situations.

4. **Non-functional Requirements -** It describes the attributes of the system.

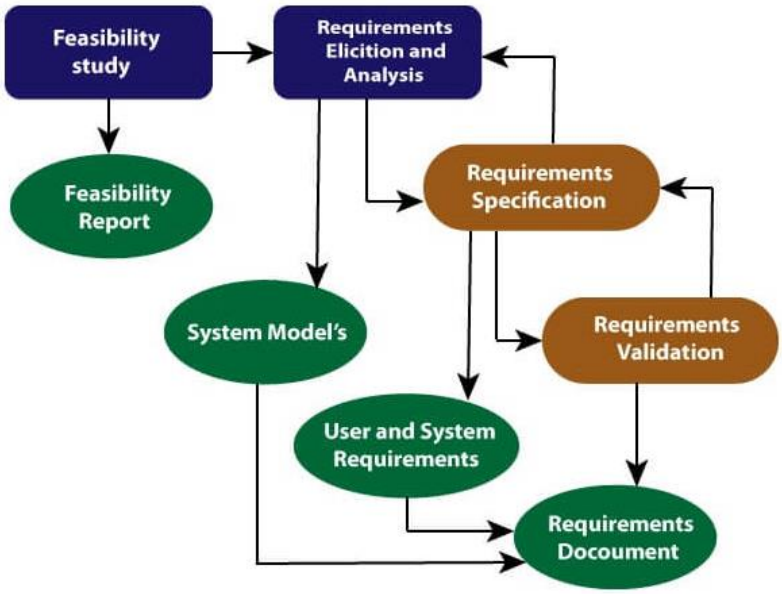
5. **Domain Requirements -** Requirements that arises from the domain of the application and that reflect characteristics of that domain. It can be either functional or non-functional specifications.

Sources of requirements

- Interviews with users and other stakeholders
- Observations of users performing tasks
- Business Case or Proposal
- Concept of Operation or Vision document
- Procedure manuals and user task lists
- Enhancement Requests for the existing system

- Marketing material and product definitions
- Analysis of a market leader or competitor's products

Requirement engineering Process



Requirement Engineering Process

It is a four-step process, which includes

1. Feasibility Study
2. Requirement Elicitation and Analysis
3. Software Requirement Specification
4. Software Requirement Validation

1. Feasibility study

- When the client approaches the organization for getting the desired product developed, it comes up with rough idea about what all functions the software must perform and which all features are expected from the software.
- Referencing to this information, the analysts do a detailed study about whether the desired system and its functionality are feasible to develop.
- This feasibility study is focused towards goal of the organization. This study analyses whether the software product can be practically materialized in terms of implementation, contribution of project to organization, cost constraints and as per values and objectives of the organization. It explores technical aspects of the project and product such as usability, maintainability, productivity and integration ability.
- The output of this phase should be a feasibility study report that should contain adequate comments and recommendations for management about whether or not the project should be undertaken.
- The objective behind the feasibility study is to create the reasons for developing the software that is acceptable to users, flexible to change and conformable to established standards.

Types of Feasibility

1) **Technical Feasibility -** Technical feasibility evaluates the current technologies, which are needed to accomplish customer requirements within the time and budget.
2) **Operational Feasibility -** Operational feasibility assesses the range in which the required software performs a series of levels to solve business problems and customer requirements.
3) **Economic Feasibility -** Economic feasibility decides whether the necessary software can generate financial profits for an organization.

2. Requirement Elicitation and Analysis

- This is also known as the gathering of requirements. Here, requirements are identified with the help of customers and existing systems processes, if available.
- Analysis of requirements starts with requirement elicitation. The requirements are analysed to identify inconsistencies, defects, omission, etc. We describe requirements in terms of relationships and also resolve conflicts if any.

3. Software Requirement Specification

- Software requirement specification is a kind of document which is created by a software analyst after the requirements collected from the various sources - the requirement received by the customer written in ordinary language.
- It is the job of the analyst to write the requirement in technical language so that they can be understood and beneficial by the development team.
- The models used at this stage include ER diagrams, data flow diagrams (DFDs), function decomposition diagrams (FDDs), data dictionaries, etc.

4. Software Requirement Validation

- After requirement specifications developed, the requirements discussed in this document are validated. The user might demand illegal, impossible solution or experts may misinterpret the needs. Requirements can be the check against the following conditions -

  ✓ If they can practically implement
  ✓ If they are correct and as per the functionality and specially of software
  ✓ If there are any ambiguities
  ✓ If they are full
  ✓ If they can describe

Typical Requirements Engineering Problems

1. **Undocumented processes**
   - In many organisations there is often no or very poor documentation available about existing processes. In this situation, requirements gathering becomes a two-step process. Firstly, back-engineering of existing process, and then identifying areas for improvement and optimisation.
   - To ensure requirements are full and correct, it's critical to identify key stakeholders and subject matter experts and engage with them directly. This helps eliminate any

assumptions and provides a full picture. Drawing business process maps and visualising workflows are effective techniques that can be used in this situation.

2. **Conflicting requirements**
   - Uncertainty about exiting process or different priorities for different stakeholders, often leads to conflicting requirements. If this is the case, the role of a business analyst is to document all requirements, identify contradictory requests and let stakeholders decide on priorities.
   - As a business analyst you may have some recommendations about what should be prioritised, but it's still important to hear stakeholders' opinion. Setting up a poll can be one of the ways to get clarity about what is important to the majority of stakeholders.

3. **Lack of access to end users**
   - Unavailability of end users may occur due to a few reasons and requires appropriate resolution. Sometimes end users are too busy with their day-to-day work and unwilling to participate in requirements gathering activities.
   - In such situations the best a business analyst can do is to minimise the number and length of engagements. Doing as much research as possible prior to the engagement will help to make the conversation more structured and insightful. It is almost like turning requirements gathering into requirements validation sessions. Defining focus groups and finding the most suitable end-users in each group will also help.

4. **Stakeholder design**
   - This is the case when the stakeholders or end-user have the urge to dictate how the system should work rather than providing details about what the system should do. Listening to stakeholders about potential solutions can be insightful but may also divert from actual problems and better solution designs.
   - To prevent such situations, validate each potential 'false requirement' by asking 'why?', eventually it will reveal the 'true' requirement.

5. **Communication problems**
   - This category included language barriers, wrong assumptions, unclearly defined vocabulary, and excessive use of professional terminology that can lead to misunderstandings between stakeholders and a business analyst.
   - The best strategy to avoid such situation is to communicate often and establish two-way communication. Document gathered requirements and send them for review and feedback to multiple subject matter experts, create and share a glossary of terms, and always verify assumptions.

Requirement modelling strategies

Following are the requirement modelling strategies,

1. Flow Oriented Modeling
2. Class-based Modeling

1. **Flow Oriented Modeling:** It shows how data objects are transformed by processing the function.

The Flow oriented elements are:

a) Data flow model
b) Control flow model
c) Control Specification
d) Process Specification

2. **Class-based Modeling:** Class based modeling represents the object. The system manipulates the operations.

The elements of the class-based model are:

a) classes and object
b) attributes
c) operations
d) class – responsibility - collaborator (CRS) models.

Overview of UML

- UML stands for **Unified Modelling Language.**
- UML is used to model an application structure, behaviour and even business processes.
- It is a visual modeling language used for analysis, design, implementation of software-based systems, modeling business and similar processes.
- It is a standard modeling language, not a software development process.
- It is a standard notation for the modeling of real-world objects as a first step in developing an object-oriented methodology.

Advantages of UML

- UML provides a standard for software development.
- It reduces the development time and cost to develop diagrams of UML using supporting tools.
- It has a large visual element to construct.
- Communication with programmers and outside contractors will be more efficient.
- The past faced issues by the developers no longer exists.

Disadvantages of UML

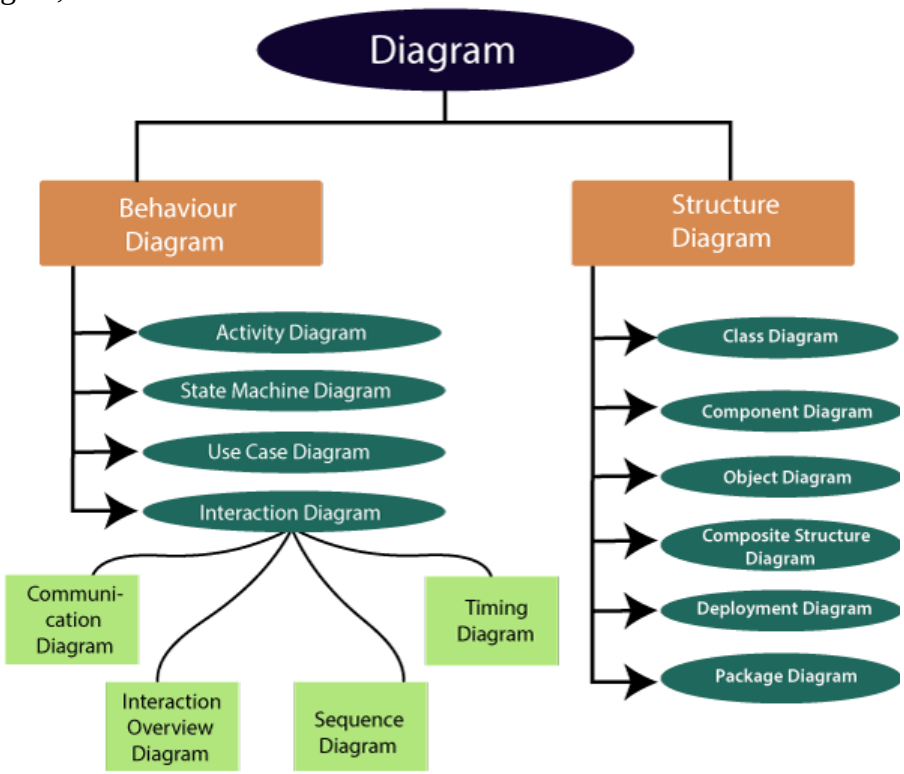
- UML is large and complex.
- UML is difficult to synchronize the code with models.
- It is difficult to keep multiple models or diagrams consistent with each other.
- UML is limited to what the vendor provides out of the box, usually some form of a code generator.
- It does not define a standard file format, which means each UML tool vendor stores the representation of its UML model in proprietary format.

Why is UML used?

- UML is an international standard.
- UML becomes the most successful modeling language in the history of computer technology.
- UML is widely known, used and supported in tools.

Types of diagrams

The UML diagrams are categorized into structural diagrams, behavioural diagrams, and also interaction overview diagrams. The diagrams are hierarchically classified in the following figure,



1. **Structural Diagrams**
   - Structural diagrams depict a static view or structure of a system. It is widely used in the documentation of software architecture.
   - It embraces class diagrams, composite structure diagrams, component diagrams, deployment diagrams, object diagrams, and package diagrams.
   - It presents an outline for the system. It stresses the elements to be present that are to be modelled.

2. **Behavioural Diagrams**
   - Behavioural diagrams portray a dynamic view of a system or the behaviour of a system, which describes the functioning of the system.
   - It includes use case diagrams, state diagrams, and activity diagrams.
   - It defines the interaction within the system.

3. **Interaction Diagrams**
   - Interaction diagrams are a subclass of behavioural diagrams that give emphasis to object interactions and also depicts the flow between various use case elements of a system.
   - In simple words, it shows how objects interact with each other and how the data flows within them.
   - It consists of communication, interaction overview, sequence, and timing diagrams.