Software Engineering Class

Dr. Engineer\ Ibrahim Eskandar

Level 3-IT+CS

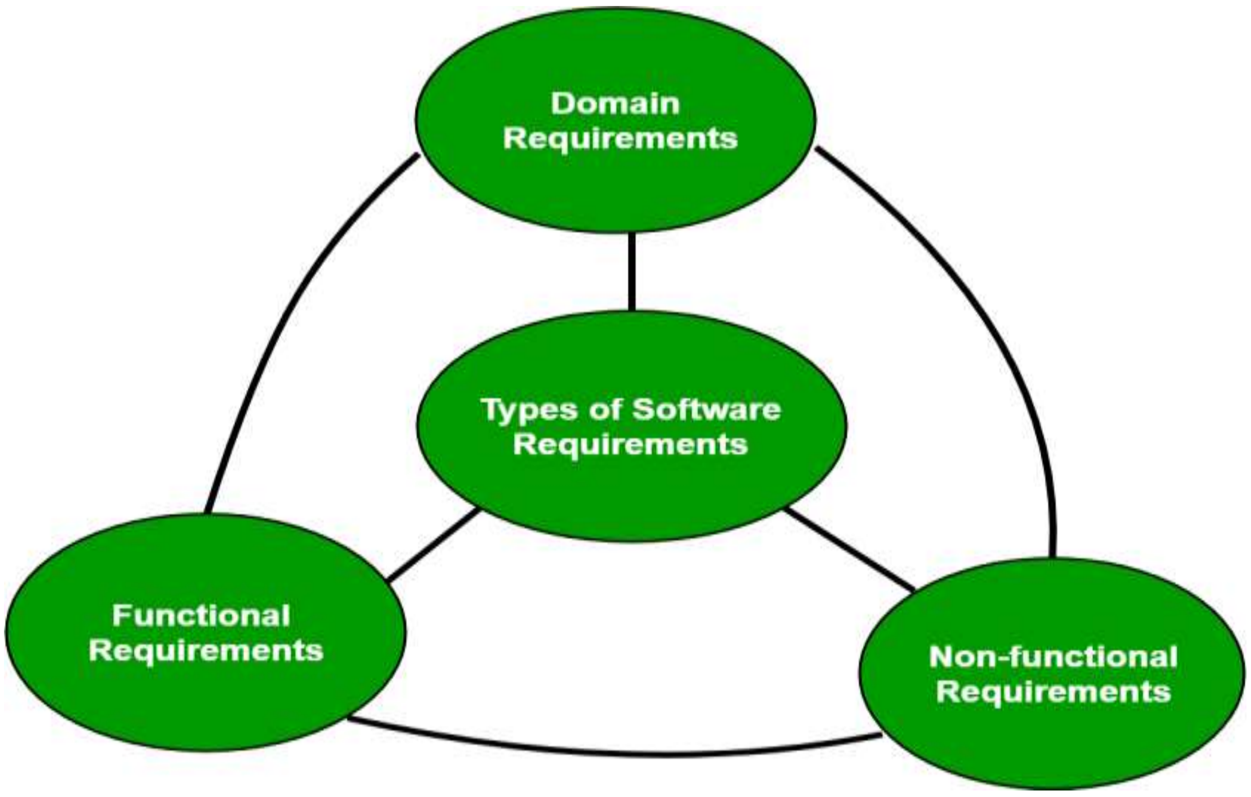
1 Requirements Engineering |

Requirement can be defined as follow

1- A condition or capability needed by a user to solve a problem or achieve an objective
2- A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification or other formally imposed documents
3- A documented representation of a condition or capability as in 1 and 2.

A software requirement can be of 3 types:

- Functional requirements
- Non-functional requirements
- Domain 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. For example, in a hospital management system, a doctor should be able to retrieve the information of his patients. Each high-level functional requirement may involve several interactions or dialogues between the system and the outside world. In order to accurately describe the functional requirements, all scenarios must be enumerated. There are many ways of expressing functional requirements e.g., natural language, a structured or formatted language with no rigorous syntax and formal specification language with proper syntax.

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.

They basically deal with issues like:

- Portability
- Security
- Maintainability
- Reliability
- Scalability
- Performance
- Reusability
- Flexibility

NFR's are classified into following types:

- Interface constraints
- Performance constraints: response time, security, storage space, etc.
- Operating constraints
- Life cycle constraints: maintainability, portability, etc.
- Economic constraints

The process of specifying non-functional requirements requires the knowledge of the functionality of the system, as well as the knowledge of the context within which the system will operate.

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. For instance, in an academic software that maintains records of a school or college, the functionality of being able to access the list of faculty and list of students of each grade is a domain requirement. These requirements are therefore identified from that domain model and are not user specific.

1.1 Requirements Engineering Process

Requirement Engineering is the process of defining, documenting and maintaining the requirements. It is a process of gathering and defining service provided by the system. Requirements Engineering Process consists of the following main activities:

- Requirements elicitation
- Requirements specification
- Requirements verification and validation
- Requirements management

Requirements Elicitation:

It is related to the various ways used to gain knowledge about the project domain and requirements. The various sources of domain knowledge include customers, business manuals, the existing software of same type, standards and other stakeholders of the project.
The techniques used for requirements elicitation include interviews, brainstorming, task analysis, Delphi technique, prototyping, etc. Elicitation does not produce formal models of the requirements understood. Instead, it widens the knowledge domain of the analyst and thus helps in providing input to the next stage.

Requirements specification:

This activity is used to produce formal software requirement models. All the requirements including the functional as well as the non-functional requirements and the constraints are specified by these models in totality. During specification, more knowledge about the problem may be required which can again trigger the elicitation process.
The models used at this stage include ER diagrams, data flow diagrams (DFDs), function decomposition diagrams (FDDs), data dictionaries, etc.

Requirements verification and validation:

Verification:
It refers to the set of tasks that ensure that software correctly implements a specific function.

Validation: It refers to a different set of tasks that ensure that the software that has been built is traceable to customer requirements. If requirements are not validated, errors in the requirements definitions would propagate to the successive stages resulting in a lot of modification and rework.

The main steps for this process include:

- The requirements should be consistent with all the other requirements i.e no two requirements should conflict with each other.
- The requirements should be complete in every sense.
- The requirements should be practically achievable.

Reviews, buddy checks, making test cases, etc. are some of the methods used for this.

Requirements management:

Requirement management is the process of analyzing, documenting, tracking, prioritizing and agreeing on the requirement and controlling the communication to relevant stakeholders. This stage takes care of the changing nature of requirements. It should be ensured that the SRS is as modifiable as possible so as to incorporate changes in requirements specified by the end users at later stages too. Being able to modify the software as per requirements in a systematic and controlled manner in an extremely important part of the requirements engineering process.

1.2 Requirements Validation Techniques

Requirements validation is the process of checking that requirements defined for development, define the system that the customer really wants. To check issues related to requirements, we perform requirements validation. We usually use requirements validation to check error at the initial phase of development as the error may increase excessive rework when detected later in the development process.
In the requirements validation process, we perform a different type of test to check the requirements mentioned in the Software Requirements Specification (SRS), these checks include:

- Completeness checks
- Consistency checks
- Validity checks
- Realism checks
- Ambiguity checks
- Verifiability

The output of requirements validation is the list of problems and agreed on actions of detected problems. The lists of problems indicate the problem detected during the process of requirement validation. The list of agreed action states the corrective action that should be taken to fix the detected problem.

There are several techniques which are used either individually or in conjunction with other techniques to check to check entire or part of the system:

1. **Test case generation:**
   Requirement mentioned in SRS document should be testable, the conducted tests reveal the error present in the requirement. It is generally believed that if the test is

difficult or impossible to design than, this usually means that requirement will be difficult to implement and it should be reconsidered.

2. **Prototyping:**
   In this validation techniques the prototype of the system is presented before the end-user or customer, they experiment with the presented model and check if it meets their need. This type of model is generally used to collect feedback about the requirement of the user.
3. **Requirements Reviews:**
   In this approach, the SRS is carefully reviewed by a group of people including people from both the contractor organisations and the client side, the reviewer systematically analyses the document to check error and ambiguity.
4. **Automated Consistency Analysis:**
   This approach is used for automatic detection of an error, such as nondeterminism, missing cases, a type error, and circular definitions, in requirements specifications. First, the requirement is structured in formal notation then CASE tool is used to check in-consistency of the system, the report of all inconsistencies is identified and corrective actions are taken.
5. **Walk-through:**
   A walkthrough does not have a formally defined procedure and does not require a differentiated role assignment.
   - Checking early whether the idea is feasible or not.
   - Obtaining the opinions and suggestion of other people.
   - Checking the approval of others and reaching an agreement.

1.3 Requirements Elicitation

Requirements elicitation is perhaps the most difficult, most error-prone and most communication intensive software development. It can be successful only through an effective customer-developer partnership. It is needed to know what the users really need.
There are a number of requirements elicitation methods. Few of them are listed below –

1. Interviews
2. Brainstorming Sessions
3. Facilitated Application Specification Technique (FAST)
4. Quality Function Deployment (QFD)
5. Use Case Approach

The success of an elicitation technique used depends on the maturity of the analyst, developers, users and the customer involved.

1. 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.

Interviews maybe be open ended or structured.

1. In open ended interviews there is no pre-set agenda. Context free questions may be asked to understand the problem.
2. In structured interview, agenda of fairly open questions is prepared. Sometimes a proper questionnaire is designed for the interview.

2. 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.

3. Facilitated Application Specification Technique:
It's objective is to bridge the expectation gap – difference between what the developers think they are supposed to build and what customers think they are going to get.
A team oriented approach is developed for requirements gathering.
Each attendee is asked to make a list of objects that are-

1. Part of the environment that surrounds the system
2. Produced by the system
3. 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.

4. Quality Function Deployment:
In this technique customer satisfaction is of prime concern, hence it emphasizes on the requirements which are valuable to the customer.
3 types of requirements are identified –

- **Normal requirements –** In this the objective and goals of the proposed software are discussed with the customer. Example – normal requirements for a result management system may be entry of marks, calculation of results etc
- **Expected requirements –** These requirements are so obvious that the customer need not explicitly state them. Example – protection from unauthorised access.

- **Exciting requirements –** It includes features that are beyond customer's expectations and prove to be very satisfying when present. Example – when an unauthorised access is detected, it should backup and shutdown all processes.

The major steps involved in this procedure are –

1. Identify all the stakeholders, eg. Users, developers, customers etc
2. List out all requirements from customer.
3. A value indicating degree of importance is assigned to each requirement.
4. 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

5. Use Case Approach:
This technique combines text and pictures to provide a better understanding of the requirements.
The use cases describe the 'what', of a system and not 'how'. Hence they only give a functional view of the system.
The components of the use case deign includes three major things – Actor, Use cases, use case diagram.

1. **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.
2. **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.
3. **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.

1.4 Challenges in eliciting requirements

Prerequisite – Requirements Elicitation
Eliciting requirements is the first step of Requirement Engineering process. It helps the analyst to gain knowledge about the problem domain which in turn is used to produce a formal specification of the software. There are a number of issues and challenges encountered during this process. Some of them are as follows:

1. **Understanding large and complex system requirements is difficult –**
   The word 'large' represents 2 aspects:

   - (i) Large constraints in terms of security, etc. due to a large number of users.

- (ii) Large number of functions to be implemented.

The complex system requirements include those requirements which are unclear and difficult to implement.

2. **Undefined system boundaries –**
   There might be no defined set of implementation requirements. The customer may go on to include several unrelated and unnecessary functions besides the important ones, resulting in an extremely large implementation cost which may exceed the decided budget.
3. **Customers/Stakeholders are not clear about their needs. –**
   Sometimes, the customers themselves maybe unsure about the exhaustive list of functionalities they wish to see in the software. This might happen when they have a very basic idea about their needs but haven't planned much about the implementation part.
4. **Conflicting requirements are there –**
   There is a possibility that two different stakeholders of the project express demands which contradict each other's implementation. Also, a single stakeholder might also sometimes express two incompatible requirements.
5. **Changing requirements is another issue –**
   In case of successive interviews or reviews from the customer, there is a possibility that the customer expresses a change in the initial set of specified requirements. While it is easy to accommodate some of the requirements, it is often difficult to deal with such changing requirements.
6. **Partitioning the system suitably to reduce complexity –**
   The projects can sometimes be broken down into small modules or functionalities which are then handled by separate teams. Often, more complex and large projects require more partitioning. It needs to be ensured that the partitions are non-overlapping and independent of each other.
7. **Validating and Tracing requirements –**
   Cross-checking the listed requirements before starting the implementation part is very important. Also, there should be forward as well as backward traceability. For eg, all the entity names should be the same everywhere, i.e., there shouldn't be a case where 'STUDENT' and 'STUDENTS' are used at separate places to refer to the same entity.
8. **Identifying critical requirements –**
   Identifying the set of requirements which have to be implemented at any cost is very important. The requirements should be prioritized so that crucial ones can be implemented first with the highest priority.
9. **Resolving the "to be determined" part of the requirements –**
   The TBD set of requirements include those requirements which are yet to be resolved in the future. The number of such requirements should be kept as low as possible.
10. **Proper documentation, proper meeting time and budget constraints –**
    Ensuring a proper documentation is an inherent challenge, especially in case of changing requirements. The time and budget constraints too need to be handled carefully and systematically.