UNIT II

CHAPTER 4

Requirement Engineering

Syllabus

Requirements Engineering: User and system requirements, Functional and non-functional requirements, Types & Metrics, A spiral view of the requirements engineering process. Software Requirements Specification (SRS): The software requirements Specification document, The structure of SRS, Ways of writing a SRS, structured & tabular SRS for an insulin pump case study, Requirements elicitation & Analysis: Process, Requirements validation, Requirements management. Case Studies: The information system. Case study - Mental health care patient management system (MHC-PMS).

Syllabus Topic : Requirements Engineering

4.1 Introduction

Requirements engineering helps software engineers and software developers to better understand the problem and the nature of the problem they are going to work on. It includes the set of tasks that lead to understanding of :

- What will be business impact of the software ?
- What the customer wants exactly ?
- How end user will interact with the system ?

Software engineers (sometimes also called as system engineer or analyst) and other project stakeholders (managers, customers and users), all participate in requirements engineering.

Designing and building the elegant computer software will not satisfy customer's requirements. If requirements analysis is wrong, then design will be wrong.

- Ultimately coding will be wrong. Finally, software expectations will not match with outcomes. Hence requirements engineering should be carried out carefully.

4.2 Requirement Engineering

- The requirements engineering is carried out through the execution of following functions. Some of these requirements engineering functions occur in parallel.
- All these seven functions focus on the customer's needs and care must be taken to satisfy it. All these functions collectively form the strong base for software design and construction. These functions are :

1. Inception
2. Elicitation
3. Elaboration
4. Negotiation
5. Specification
6. Validation
7. Requirement management

4.2.1 Inception Q23 SPPU - Feb. 16

University Question

Q. What do you mean by requirement inception ?

(Feb. 2016, 6 Marks)

- Inception is beginning and it is usually said that, 'well beginning is half done'. But it is always problematic for the developer that what should be the starting point i.e. 'from where to start'.
- The customer and developer meet and they decide the overall scope and nature of the problem statement. The aim is :
  - To have the basic understanding of the problem.
  - To know the people who will use the software.
  - To know exact nature of problem that is expected from customer's side.
  - To maintain effectiveness of preliminary communication.
  - To have collaboration between customer and developer.
- The requirements engineering is a 'communication intensive' activity because a requirement gathering is an initial step for design. Hence exact requirements are gathered with customer communication.
- The developer must ask following questions:
  - Who is behind the request for this work?
  - Who will use the solution ?
  - What will be the economical benefits of a successful solution ?
  - Is there another source of solution for the same problem ?

4.2.2 Elicitation Q24 SPPU - Feb. 16

University Question

Q. What do you mean by requirement elicitation ?

(Feb. 2016, 6 Marks)

Elicitation means, 'to draw out the truth or reply from anybody'. In relation with requirements engineering, elicitation is a task that helps the customer to define what is required.

- To know the objectives of the system or the project to be developed is a critical job. Why it is difficult to get a clear understanding of customer wants? To answer this question we have a number of problems like :

Problems of scope

Many times customer states unnecessary technical details. The unnecessary details may confuse developer instead of giving clarity of overall system objectives.

Problems of understanding

Sometimes both customer as well as developer has poor understanding of :

- What is needed ?
- Capabilities and limitations of the computing environment.
- Understanding of problem domain.
- Specify requirements those conflict with other customers and users.

Problems of volatility

Volatility means 'change from one state to another'. The customer's requirements may change time to time. This is also a major problem while deciding fixed set of requirements.

4.2.3 Elaboration

- Elaboration means 'to work out in detail'. The information obtained during inception and elicitation is expanded and modified during elaboration.
- Now requirements engineering activity focuses on developing the technical model of the software that will include :
  - Functions
  - Features
  - Constraints

- Thus, elaboration is an 'analysis modeling' action. This model focuses on 'how the end user will interact with the system'.

4.2.4 Negotiation

SPPU - Feb. 15

University Question

Q. What do you mean by requirement negotiation? **(Feb. 2015, 2 Marks)**

- Here, 'Negotiation' means 'discussion on financial and other commercial issues'.
- In this step customer, user and other stakeholders discuss to decode :
  - To rank the requirements
  - To decide priorities
  - To decide risks
  - To finalize the project cost
  - The impact of above issues on project cost and delivery date.

4.2.5 Specification

- The specification is the final work product produced by requirement engineer. The specification may take different forms :
  - A written document
  - A set of graphical model
  - A formal mathematical model
  - A collection of scenarios
  - A prototype
  - Any combination of above
- The specification is considered as the foundation of all subsequent software engineering activities.
- The specification describes the function and performance of the computer based systems. The specification also describes the constraints are necessary in its development.

4.2.6 Validation

SPPU – Feb.15, Feb. 16

University Question

Q. What do you mean by requirement validation ? **(Feb. 2015, Feb. 2016, 6 Marks)**

- All previous work completed will be just useless and meaningless if it is not validated against the customers expectations.
- The requirement validation checklist includes :
  - All requirements are stated clearly ?
  - Are the requirements misinterpreted ?
  - Does the requirements violate any system domain constrains?
  - Is the system requirement traceable to the system model that is created?
  - Are the requirements associated with performance, behaviour and operational characteristics ?

4.2.7 Requirement Management

- Requirement management is a software engineering task that helps the project team members to identify the requirements, control the requirements, track the requirements and changes to requirements at any time as the project exceeds.
- The requirement management task starts with identification and each of the requirements is assigned a unique identifier.
- After the requirements finalization, the traceability table is developed. Traceability table relates the requirements to one or more aspects of the system or it is environment. The traceability tables are used as requirements database and are useful in understanding how change in one particular requirement will affect other parts or aspects of the system.

4.2.8 Initiating the Requirement Engineering Process

- In requirements engineering process, following are considerable issues :

- Customers may be located at different cities or countries.
- Customers do not have clear idea of product requirements.
- Different customers may have different and conflicting opinions about the system to be built.
- Customers may have limited technical knowledge.
- Customers may have only limited time to interact with requirement engineer.

Hence, considering all these issues, following are the steps required to initiate the requirements engineering process :

Identifying the Stakeholders

- Stakeholder is anyone who has direct interest in or benefits from the system that is to be developed. Hence, we can list following people as stakeholders :
  - Business operations manager
  - Product managers
  - Marketing people
  - Internal and external customers
  - End users
  - Consultants
  - Product engineers
  - Software engineers
  - Support and maintenance engineers
  - Others
- All these people have different view of the system regarding the system that is to be developed.

Recognizing Multiple Viewpoints

As many stakeholders exist, they all have different views regarding the system that is to be developed.

For example

- Marketing people are interested in functions and features having potential market.
- Business people are interested in functions and features those can be set within minimum budget.
- End users are interested in functions and features those will be easy to understand and use.
- Software engineers are interested in functions and features those support their existing infrastructure.
- Support engineer may focus on the maintainability of the software.
- It's a duty of software engineer to consider all these viewpoints of all the stakeholders and find the consistent and balanced set of requirements.

Working towards collaboration

Customers must collaborate with themselves and software engineering practitioners to get successful system. The job of requirement engineer is to find :

1. **Common areas :** The requirements for which all stakeholders agree.
2. **Conflict areas :** The requirements that are purposed by one stakeholder but opposed by another stakeholder.

Now, the higher authorities like business manager or senior technologist can take the decision for the requirements either to forward or to reject these requirements.

Asking the First Questions

The requirements engineering is a 'communication intensive' activity because 'requirements gathering' is an initial step for design. Hence following three sets of questions are used to gain to understand the requirements :

1. **First set : The context free questions**
   - Who is behind the request for this work ?
   - Who will use the solution ?
   - What will be the economical benefits of a successful solution ?
   - Is there another source of solution for the same problem ?

2. **Second set : Questions to gain preliminary understanding of problem**
   - How will you characterize good output that will be generated by successful solution ?
   - What are the probable problems for this solution ?
   - What is the business environment in which this solution will be used ?
   - What are special performance issues and constraints that will affect the solution ?
3. **Third set : Questions those focus on effectiveness of the communication activity**
   - Are the questions official ?
   - Are the questions related to the problem ?
   - Are the questions 'too many' in quantity ?
   - Can anyone else provide additional information ?
   - Are some questions left to ask ?

Syllabus Topic : User and System Requirements

4.3 User and System Requirements

- The fundamental definition of the requirements for a system is :

  "The set of descriptions listed for the system to accomplish the required services and the constraints on its operation."
- These requirements reflect the needs of customers for a system. The process of identifying the customers needs, analyzing, documenting and checking the services and constraints that a system should provide, is called requirements engineering (RE).
- Most of the time, the term 'Requirement' is not used in software organizations. But the requirement is illustrated as 'high-level, abstract statement of services that a system should provide for its users'.
- The problem exists when we talk about distinction between different levels of description. Primarily the requirements specification can divided into two important classes as follows :
  - User requirements
  - System requirements
- The **User requirements** are the statements narrated by the customer in simple natural language and with some useful diagrams.
- The customer describes what services the system should provide and the constraints on the operations and functions of the system.
- **The System requirements** are more detailed descriptions of the software system's functions, services, and operational constraints.
- The system requirements document also called as functional specification, should define exactly what is to be implemented. It is also the contract between the system buyer and the software developers. All these requirements are well documented on paper for future references.
- Following figure exhibits the readers of different types of requirements specification :

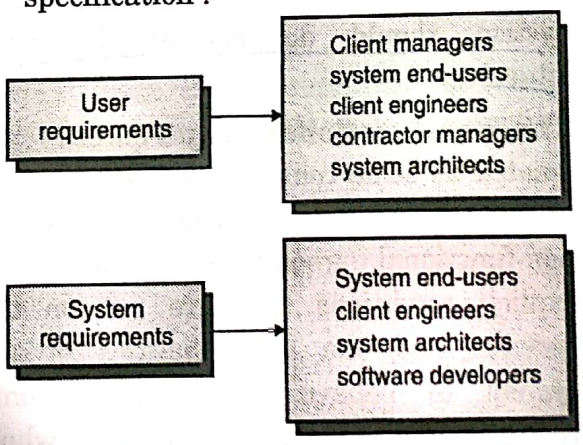


Fig. 4.3.1 Readers of different types of requirements specification

- The readers of the **user requirements** are interested only in identifying detailed requirements from the user that what facilities a system should provide to its end users.
- The readers of the **System requirements** need to know more precisely what the system will do because they are concerned with how it will support the business processes or because they are involved in the system implementation.

Syllabus Topic : Functional and Non - Functional Requirements

4.4 Functional and Non- Functional Requirements

SPPU - Feb. 15, May 16

University Question

Q. What are functional and non functional requirements of software?

(Feb. 2015, May 2016, 7 Marks)

- The system requirements are classified into following two types :
  - Functional requirements
  - Non-functional requirements
- **The functional requirements** are the statements of the services that a system should provide and how the system must react to a given input and the system reacts in some situations. The functional requirement should also mention that a system should do nothing in some situations.
- **The non-functional requirements** are treated as some constraints that a system should behave in some situations. Also the non-functional requirements are the explicit conditions that are put on the services and the functionalities of the system to behave. The non-functional requirements are applied on the whole system and not on the individual system features.

4.4.1 Functional Requirements

- Actually functional requirements describe how a system should behave and it is in a given situation. The functional requirements are often depended on the following parameters :
  - Type of the software use construction,
  - The users of the software,
  - The approach of writing requirements by the organization.
- The functional requirements are always written in such a style that it is easily understood by all type of users.
- Some of the specific functional requirements give the expected input desired output and the system functions.
- The functional requirements also specify the facilities provided by the software system. The functional requirements may have different levels of details. The functional requirements must be complete and consistent in nature.
- The complete functional requirement means it should define all the services that are needed by all categories of the user. And consistent means that the functional requirements should not have any contradictory statement i.e. for one requirement there should not be two definitions.
- But in larger systems and complex systems it is highly impossible to maintain completeness and consistency.

4.4.2 Non-functional Requirements

- The non-functional requirements are not directly related to the functions, services of the system and the desired outputs.

- The non-functional requirements are related to the system properties that are listed below :
  - Reliability
  - Speed of responses i.e. response time,
  - System performance,
  - System security,
  - System availability,
  - Portability,
  - Robustness,
  - Ease of use etc.
- In contrast to functional requirements, the non-functional requirements are more critical. Therefore, if functional requirements are failed then the whole system may be failed and become unusable.
- For example if a air system does not provide reliability, then it can not be used in actual practice. It will not be called as a safe system.
- The failure of an individual non-functional requirement may lead to failure of the whole system.

Syllabus Topic : Types and Metrics

4.5 Types and Metrics

- In actual practice, the classification of different **Types of requirement** is not very much clear according to the definitions of requirements described in earlier sections.
- A user requirement which is directly connected to the parameters like security, authorizations is supposed to be the nonfunctional requirement.
- But when such systems are developed in more detail, this requirement may generate other requirements that are clearly functional, such as the need to include user authentication facilities in the system etc.
- This exhibits that requirements are not independent and that one requirement often generates or constrains other requirements.
- Therefore the system requirements not only specify the services or the features of the system that are desired, but they also specify the necessary functionality to ensure that these services or the features are delivered properly.

4.5.1 Metrics for Specifying Non-functional Requirements

- When we write non-functional requirements, we can test it. Following are some of the metrics that can be used to specify non-functional properties of the system. We can easily measure these properties or characteristics when the software is under testing.
  - **Speed** : While measuring the speed property, we observe the speed of the processed transactions. We also see event response time and event refresh time.
  - **Robustness** : Time to recover after failure occurred, Probability of data corruption on failure and Percentage of events causing failure.
  - **Portability** : Number of target systems and Percentage of target dependent statements.
  - **Size** : Measurement of read only memory (ROM) chips available in the system.
  - **Ease of use** : Number of help frames available and training time required.
  - **Reliability** : It is the mean time to failure (MTTF), or Probability of unavailability, Rate of failure occurrence etc.

- If the non-functional requirements are given separately from the functional requirements, then the relationships between them may be difficult to understand. Non-functional requirements such as reliability, safety, and confidentiality requirements are particularly important for critical systems.

Syllabus Topic : A Spiral View of the Requirements Engineering Process

4.6 A Spiral View of the Requirements Engineering Process

- Basically requirements engineering processes includes following four high-level activities :
  - **Feasibility study** : To check whether the system is useful for the business.
  - **Requirements elicitation and analysis** : Finding the requirements from the user and conduct analysis on them.
  - **Requirements specification** : Requirements received in some natural language and in the form of some diagrams and convert them in some standard form.
  - **Requirements validation** : It is actually the checking of systems functioning. Whether it is in function according to the requirement elicited by the user.

All these activities constitute requirements engineering process and can be organized as an iterative process to form a spiral view. In actual practice, all these activities are interleaved in one another. The spiral view of the activities involved in the requirements engineering process as exhibited as follows :

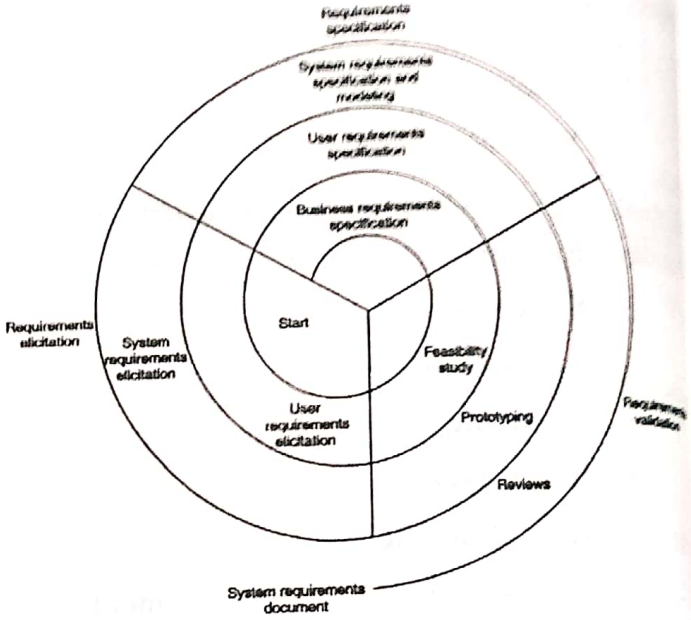


Fig. 4.6.1: Spiral view of the activities involved in the requirements engineering process

Syllabus Topic : Software Requirements Specification (SRS), The Structure of SRS

4.7 Software Requirements Specification (SRS)

Review Question

Q. What is SRS ? Why SRS is known as back-bo specification of the system ? What are maj issues addressed by SRS ?

- Basically SRS or software requiremer specification is an official document or th statement of what the system develope should implement.
- It includes both :
  - System requirement.
  - User requirement.
- The SRS has a set of users like sen management of the organization, engine responsible for developing the software.

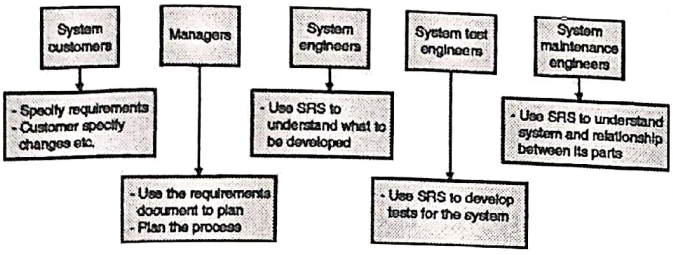


Fig. 4.7.1 : Users of SRS

- The details which are included in SRS depend on the type of system that is being developed and the type of the development process.
- A number of large organisations, such as IEEE have defined standards for software requirements document.
- The most widely used standard is IEEE/ANSI 830-1998. This standard suggests the following structure for requirements document :
  1. Introduction
     - Purpose of SRS
     - Scope of product.
     - Definitions, acronyms, abbreviations.
     - References
     - Overview
  2. General description
     - Product perspective
     - Product Functions
     - User characteristics
     - General constraints
     - Assumptions and dependencies
  3. Specific requirements
  4. Appendices
  5. Index.
- SRS is very useful when an outside contractor is developing the software system.
- For business system, where requirements are unstable, SRS play an important role.

Syllabus Topic : Ways of Writing a SRS

4.7.1 Writing Software Requirements Specifications

- Proper software requirement gathering is very important at the start of the software development. Generally it is done by professional software developers.
- Document is created in the requirement analysis which is generally referred as SRS or software requirement specification document.
- It is first project deliverable.
- SRS records the end user needs in certain format. This is SRS is needed when actual software development process starts.

4.7.2 What is a Software Requirements Specification?

- Before starting software development requirement is captured from the end users or clients or customer. This requirement is written in certain format which is called as SRS. Generally this document is created before starting the development work.
- It signifies both developers and client understood what should be implemented in the software.
- All capabilities and functionalities are stored in SRS.
- Requirement document needs in every steps of software development.
- Analyst uses this document for designing the software. Developer uses this document during the coding whereas software tester uses this document to check the functionalities of the software.
- All remaining project documents are depends upon requirement document because of which it is referred as parent of all document.

- It contains functional and non functional requirements.

Good SRS have accomplishes following goals

- It gives feedback to customer.
- The large problem can be divided into different small components.
- It acts as input to design which is part of design specification.
- It acts as product validation check.

4.7.3 What Kind of Information Should an SRS Include?

Following things are part of SRS :

- Interfaces
- Functional capabilities
- Performance levels
- Data structures/Elements
- Safety
- Reliability
- Security/Privacy
- Quality
- Constraints and limitations

4.7.4 SRS Template

- Different existing SRS templates can be used in writing SRS documents.
- We can select the template which matches our requirement.
- Template will just acts as guiding principles for writing template. Proper changes need to be done whenever necessary in template.
- Table 4.7.1 shows SRS outline :

Table 4.7.1 : A sample of a basic SRS outline

1. **Introduction** : It contains different details like purpose of software, conventions used, target audiences, extra information, SRS team members, references.
2. **Overall Description** : Overall information of the project is given in this sections. It includes perspective, functions, different user classes, working environment, design constraints, assumptions about the software.
3. **External Interface Requirements** : It contains external interface information which includes user interface, hardware interface, software interfaces, communication protocols, etc.
4. **System Features** : Which system features needs to add is given in system features. It includes different features, features description, priority, action result, functionalities, etc.
5. **Other Nonfunctional Requirements** : Non functional requirement of the project is given in this section. It includes performance requirement, safety concerns, security constraints, quality factors, documentation, user documentation.
6. **Other Requirements** It includes Appendices, glossary of the software, etc.

4.7.5 Characteristics of an SRS

Q27

SPPU - Feb. 15, Feb. 16

University Question

Q. Explain four desirable characteristics of a good Software Requirements Specification(SRS) document. **(Feb. 2015, Feb. 2016, 6 Marks)**

Review Question

Q. Explain different characteristics of Software Requirement Specification ?

- **Complete :** All the project functionalities should be recorded by SRS.
- **Consistent :** SRS should be consistent
- **Accurate** : SRS defines systems functionality in real world. We need to record requirement very carefully.

- **Modifiable** : Logical and hierarchical modification should be allowed in SRS.
- **Ranked** : Requirement should be ranked using different factors like stability, security, ease or difficulty.
- **Testable** : The requirement should be realistic and able to implement.
- **Traceable** : Every requirement we must be able to uniquely identify.
- **Unambiguous** : Statement in the SRS document should have only one meaning.
- **Valid** : All the requirements should be valid.

Syllabus Topic : Structured SRS for an Insulin Pump Case Study

4.7.6 Structured Specifications for an Insulin Pump Case Study

- Structured natural language is a way of writing system requirements where the freedom of the requirements writer is limited and all requirements are written in a standard way. This approach maintains most of the expressiveness and understandability of natural language but ensures that some uniformity is imposed on the specification. Structured language notations use templates to specify system requirements. The specification may use programming language constructs to show alternatives and iteration, and may highlight key elements using shading or different fonts.
- The Robertsons, a well known author, recommend that user requirements be initially written on cards, one requirement per card. They suggest a number of fields on each card, such as the requirements rationale, the dependencies on other requirements, the source of the requirements, supporting materials, and so on. This is similar to the approach used in the example of a structured specification shown in Table 4.7.2.

Table 4.7.2 : A structured specification of a requirement for an insulin pump

Function	Compute insulin dose: Safe sugar level.
Description	Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units.
Inputs	Current sugar reading (r2), the previous two readings (r0 and r1).
Source	Current sugar reading from sensor. Other readings from memory.
Outputs	CompDose—the dose in insulin to be delivered.
Destination	Main control loop.
Action	CompDose is zero if the sugar level is stable or falling or if the level is increasing but the rate of increase is decreasing. If the level is increasing and the rate of increase is increasing, then CompDose is computed by dividing the difference between the current sugar level and the previous level by 4 and rounding the result. If the result, is rounded to zero then CompDose is set to the minimum dose that can be delivered.
Requirements	Two previous readings so that the rate of change of sugar level can be computed.
Pre-condition	The insulin reservoir contains at least the maximum allowed single dose of insulin.
Post-condition	r0 is replaced by r1 then r1 is replaced by r2.
Side effects	None.

- To use a structured approach to specifying system requirements, you define one or more standard templates for requirements and represent these templates as structured forms. The specification may be structured around the objects manipulated by the system, the functions performed by the system, or the events processed by the system. An example of a form-based specification, in this case, one that defines how to calculate the dose of insulin to be delivered when the blood sugar is within a safe band, is shown in Table 4.7.2.

Syllabus Topic : Tabular SRS for an Insulin Pump Case Study

4.7.7 Tabular Specifications for an Insulin Pump Case Study

- Using structured specifications removes some of the problems of natural language specification. Variability in the specification is reduced and requirements are organized more effectively. However, it is still sometimes difficult to write requirements in a clear and unambiguous way, particularly when complex computations (e.g., how to calculate the insulin dose) are to be specified.
- To address this problem, you can add extra information to natural language requirements, for example, by using tables or graphical models of the system. These can show how computations proceed, how the system state changes, how users interact with the system, and how sequences of actions are performed.
- Tables are particularly useful when there are a number of possible alternative situations and you need to describe the actions to be taken for each of these. The insulin pump bases its computations of the insulin requirement on the rate of change of blood sugar levels. The rates of change are computed using the current and previous readings. Table 4.7.3 is a tabular description of how the rate of change of blood sugar is used to calculate the amount of insulin to be delivered.

Table 4.7.3 : Tabular specification of computation for an insulin pump

Condition	Action
Sugar level falling (r2 < r1)	CompDose = 0
Sugar level stable (r2 = r1)	CompDose = 0
Sugar level increasing and rate of increase Decreasing ((r2 - r1) < (r1 - r0))	CompDose = 0
Sugar level increasing and rate of increase stable or increasing ((r2 – r1) ≥ (r1 – r0))	CompDose = round ((r2 – r1)/4) If rounded result = 0 then CompDose = MinimumDose

Syllabus Topic : Requirements Elicitation and Analysis : Process

4.8 Requirements Elicitation : Process

Q28 SPPU - Dec. 12

University Question

Q. State and explain the methods for eliciting requirements. **(Dec. 2012, 8 Marks)**

- Elicitation is a task that helps the customer to define what is required. 'Eliciting requirements' step is carried out by series of following steps :

1. Collaborative requirements gathering
2. Quality function deployment
3. User scenarios
4. Elicitation work product

4.8.1 Collaborative Requirements Gathering

- Gathering software requirements is team oriented activity. All the software team members, software engineering manager members of marketing, and product engineering representatives all work together. The aim of this activity is :
  - To identify the problem.
  - To suggest the solution.
  - To negotiate different approaches.
  - To specify the preliminary set of solution requirements.
- The meeting for 'collaborative requirements gathering' is conducted to discuss all above issues.
- The basic guidelines for conducting a collaborative requirements gathering meeting are :

- Meeting is conducted and attended by both software engineers as well as customers.
- An agenda is suggested that is formal enough to cover all points those are to be discussed in the meeting.
- A 'facilitator' may be customer, developer or outsider controls the meeting.
- A definition mechanism including work sheets, charts, wall stickers, chat rooms, projectors is used.

4.8.2 Quality Function Deployment

- Quality function deployment is a technique that translates the customer's needs into technical requirements for software.
- In other words **'Quality Function Deployment'** defines the requirements in a way that maximizes customer satisfaction. Quality function deployment includes three types of requirements :

1. Normal requirements
2. Expected requirements
3. Exciting requirements

1. Normal requirements

These are the requirements clearly stated by the customer. Hence these requirements must be present for customer's satisfaction.

For example

- Graphic displays.
- Specific system functions.
- Specified output formats.

2. Expected requirements

These requirements are implicit type of requirements. These requirements are not clearly stated by the customer but even then the customer expects them.

For example

- The developed system must provide easy human machine interaction.
- The system should be menu driven,
- All hot key buttons help should be provided.
- The system should be 'user friendly'.
- The system should be easy to install.

3. Exciting requirements

These requirements are neither stated by the customer nor expected. But to make the customer more satisfied, the developer may include some unexpected requirements. For example, in word processing software development, only standard capabilities are expected. But, it will be a surprise for the customer if 'page layout capabilities' and 'advanced graphical features' are added.

4.8.3 Usage Scenarios

- It is just impossible to move into more technical software engineering activities until the software team understands how these functions and features will be used by different classes and end users.
- To understand this, the developer and users create a set of scenarios those will identify all these issues. The scenario is called as **'Use Cases'**. Details of Use Case diagrams are included in next chapter.

Try & use error

4.8.4 Elicitation Workproduct SPPU - May 13

University Question

Q. What are the workproducts of elicitation ? **(May 2013, 6 Marks)**

- The work products produced by requirement elicitation depend upon the size of the system or the system to be built.
- The information produced as a consequence of requirements gathering includes :
  - A statement of need and feasibility.
  - A statement of scope for the system or product.

- A list of customers, users and other stakeholders who participated in requirement elicitation.
- Description of system's technical environment.
- The list of requirements and domain constraints.
- The set of scenarios.
- Any prototype developed to define requirements clearly.

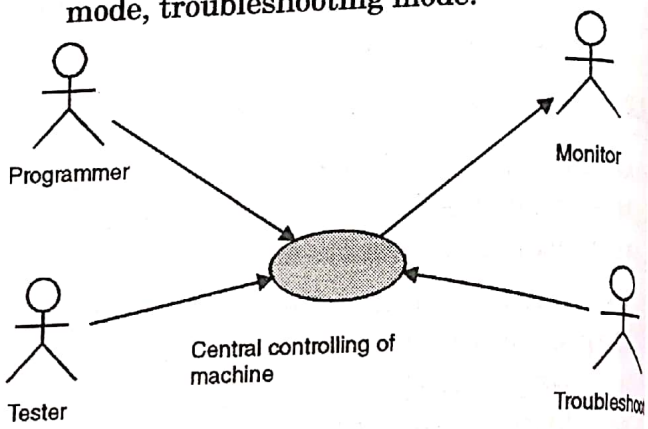
4.8.5 Elicitation Techniques

- These are the data collection techniques.
- They are used in cognitive science, knowledge engineering, linguistic management, psychology, etc.
- Knowledge is directly acquired through human being.
- It includes various techniques including interview, observations, participatory design, focus groups, etc.

4.8.6 Developing Use Cases

- A Use case exhibits the behaviour of the system according to the response received from any of the stakeholders. Alternatively a use case explains how the users will operate the system under any specific circumstances.
- Use cases are defined from actor's point of view. An actor is a role that refers the user or device that interacts with the software.
- The Use case diagram shows the relationship between actors (e.g. person, machine, another system etc) and use cases (sequence of actions i.e. procedures /functions).
- Note that the actor and end user are not necessarily the same thing. A typical user may play number of different roles when using the system, whereas an actor represents a class of external entities that plays just one role in the context of u case.
- **For example**

  Consider the application where th machine operator handles control compute for manufacturing cell. Here, machin operator is a user.
- The software for control machine require four different modes for interaction Programming mode, test mode, monitorin mode, troubleshooting mode.



Actors and use case for controlling the machine in manufacturing cell of factory

Fig. 4.8.1

- Hence four actors can be defined programmer, tester, monitor and troubl shooter.
- **There are two types of actors :**

1. Primary actors
2. Secondary actors

1. Primary actors

These actors interact with the system achieve required system function an derive intended benefits from the syste They work directly with the software.

2. Secondary actors

- These actors support the system so th primary actors can do their work. Afte identifying the actors the use cases ca

be developed. Jacobson suggests the number of questions those are answered by the use cases.

- What are primary and secondary actors?
- What are the goals of actors (i.e. primary and secondary actors) ?
- What are the preconditions that exist before the development begins ?
- What are tasks and functions that are performed by actors ?
- What are considerable exceptions ?
- What are the possible variations in primary and secondary actor's interaction ?
- What are the system information that a actor can acquire, produce ?
- What are the system information that a actor can modify ?
- Is it necessary for a actor to inform the system, the possible changes in the external environment ?
- What is the desired information of a system that an actor want to know ?
- Is it necessary for a system to inform the actor about unexpected changes ?

4.9 Requirements Analysis

Review Question

Q. Explain requirement analysis with example.

- Analysis model uses a combination of text and diagrammatic forms to depict requirements of data, functions and behaviour. So that it will be easy to understand overall requirements of the software to be built.
- The **'Software Engineer'** also called as **'Analyst'** builds the model using requirements stated by the customer.
- Analysis model validates the software requirements and represent the requirements in multiple dimensions.

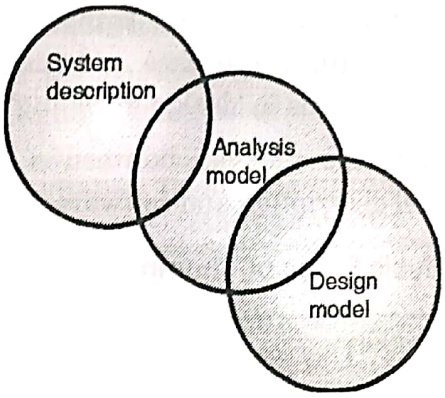


Fig. 4.9.1 : Analysis model, bridge in system description and design model

- Basic aim of analysis modelling is to create the model that represents the information, functions and behaviour of the system to be built.
- These information, functions and behaviour of the system are translated into architectural, interface and component level designs in design modeling.
- Information, functions and behaviour of the system are represented using number of different diagrammatic formats.
- As stated previously, the 'analysis model' acts as a bride between 'system description' and the 'design model'.
- The system description describes overall system functionality of the system including software, hardware, databases, human interfaces and other system elements. And the software design mainly focuses on application architectural, user interface and component level designs.
- Thus, all elements of analysis model are directly traceable to the parts of design model. Hence, the analysis model must have following three objectives :
  - To state clearly what customer wants exactly.

- To establish the basis of the 'design model'. The information, functions and behaviour of the system defined by 'analysis model' are translated into architectural, interface and component level designs in 'design modeling'.
- To bridge the gap between a system level description and software design.

4.9.1 Analysis Rules of Thumb

Review Question

Q. Explain Thumb rule for requirement analysis.

These rules are

- The requirement analysis must state the requirements at business domain and at high level of abstraction. No need to state the details.
- Each element of analysis model must help for clear understanding of software requirements and must focus on information, functions and behaviour of the system.
- Consideration of infrastructure and non-functional model should be delayed to design. For example : Define the databases if required for software under consideration. But no need to consider the details likes classes, functions and behaviour of the databases.
- Try to minimize the coupling throughout the system. The interconnections among different modules is called 'coupling'. That is the coupling measures the 'functional dependency' of different modules. Hence for good design, the interconnection among different components should be minimum.
- The analysis model provides value to all people concern to it.
- **For example :** Customer will use analysis model to validate his requirements. The Designer will use the analysis model for designing the 'design model'. End user can use same model for testing.
- The analysis model must be as simple as possible.
- Only simple model will help the easy understanding of software requirements. Hence analysis model should be simple enough.

4.9.2 Domain Analysis

Review Question

Q. Explain Domain analysis. Also discuss its advantages.

- By definition, software domain analysis is "the identification, analysis and specification of common requirements from a specific application domain." These common software domains can be reused for multiple projects within that application domain.
- In short, software domain analysis leads to 'reusable software components' in software development. Here, in specific application domain common objects, common classes, common frameworks can be identified and can be reused.
- Role of 'domain analyst' is same as job of toolsmith. The job of toolsmith is to design the tools those are used by many people for similar works. The role of domain analyst is to define reusable objects, classes, frameworks those can be used by many people in similar applications.
- **For example :** 'The specific application domain' may be 'bus reservation system'. The software domains of 'Us reservation system' can be used for 'railway reservation system'.

Advantages

- Use of such software domain analysis saves the time.
- It also reduces the development cost.

4.9.3 Requirements Modeling Approaches

Review Question

Q. What are different requirement approaches ?

Following are different requirement modeling approaches :

- **Structured analysis :** It consists of **data** and the **process** that transforms data.
- **Object-oriented analysis :** It emphasizes on the classes that collaborates with one another to focus on customer's requirements.

Syllabus Topic : Requirements Validation

4.10 Requirements Validation

SPPU - Dec. 15, Dec. 16

University Questions

Q. Explain how do we negotiate and validate requirement analysis process. **(Dec. 2015, 7 Marks)**

Q. What do you understand with the Validating requirements ? **(Dec. 2016, 5 Marks)**

- Once requirement model is built, it is checked for inconsistencies and ambiguities. Then the requirements are optimized.
- During the review of the requirements model, following questions arise :
  - Are all the requirements consistent ?
  - Whether requirements followed proper level of abstraction ?
  - Is the requirement really necessary ?
  - Are there any requirements which conflicts with one another ?
  - Is each requirement testable, once implemented ?
  - Does the requirement model properly reflect the information, function and behaviour of the system to be built ?
  - Does the requirements model use the requirements pattern to simplify the model ?
- These questions are answered to make sure that the requirements model is as per the customer's exact needs.

Syllabus Topic : Requirements Management

4.11 Requirements Management

SPPU - May 14

University Question

Q. Draw and explain the traceability table for requirement management. **(May 2014, 6 Marks)**

Review Question

Q. Explain traceability table. How it is requirement management ?

- Requirement management is a software engineering task that helps the project team members to identify the requirements, control the requirements, track the requirements and changes to requirements at any time as the project exceeds.
- The requirement management task starts with identification and each of the requirements is assigned a unique identifier.
- After the requirements finalization, the traceability table is developed. Traceability table relates the requirements to one or more aspects of the system or it is environment. The traceability tables are used as requirements database and are useful in understanding how change in one particular requirement will affect other parts or aspects of the system.
- Following are some examples of traceability table :
  - **Features traceability table** observes product features and make sure that

how it is closely related to customer requirement.

- **Source traceability table** is used to identify the source of each of the requirement.
- **Dependency traceability table** observes the relationships among requirements.
- **Subsystem traceability table** classifies the requirements as per the subsystem they govern.
- **Interface traceability table** indicates the internal and external system interface relate as per the given requirement.

Requirement \ Specific aspect of the system or its environment	A01	A02	A03	A04	A05				All
R01			✓		✓				
R02	✓		✓						
R03	✓			✓					✓
R04		✓			✓				
R05	✓	✓		✓					✓
Rnn	✓		✓						

Table 4.11.1 : Generic traceability table

Syllabus Topic : Case Studies : MHC-PMS

4.12 Case Studies : Mental Health Care Patient Management System (MHC-PMS)

- In case of Mental health care patient management system (MHC-PMS), we illustrate following requirements :
  - Product requirements
  - Organizational requirements, and
  - External requirements
- In **product requirements**, we can say that MHC-PMS will be available only on the normal working hours from Monday to Friday. Downtime for the system should not exceed more than one minute or 30 seconds.
- In **Organizational requirements**, the users of the MHC-PMS should pass through some authentication process in order to limit the access to the system. The authorization may be in the form of some passwords or some clinic chip-enabled identity cards.
- In **External requirements**, the system must provide some privacy to the patients.
- For the case of MHC-PMS, functional requirements can be listed as follows :
  1. A user (in this case, it is patient), shall be able to search the appointments lists for all hospitals.
  2. The MHC-PMS shall generate each day, for each hospital, a list of patients who are expected to attend appointments for that day.
  3. Each staff member using the system shall be identified by his or her employee number. This employee code must be unique so that system can recognize him or her.
- All these functional requirements the facilities provided by the system. All these requirements are taken from requirements document.
- In actual practice, it is highly impossible to achieve consistency and completeness in requirements.

□□□