<u>UNIT II</u>

SOFTWARE REQUIREMENTS

IEEE defines Requirement as :

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 a system component to satisfy constract, standard, specification or formally imposed document
3. A documented representation of a condition nor capability as in 1 or 2

SOFTWARE REQUIREMENTS

• Encompasses both the User's view of the requirements(the external view) and the Developer's view(inside characteristics)

User's Requirements

--Statements in a natural language plus diagram, describing the services the system is expected to provide and the constraints

• System Requirements --Describe the system's function, services and operational condition

SOFTWARE REQUIREMENTS

• System Functional Requirements

--Statement of services the system should provide

--Describe the behavior in particular situations

--Defines the system reaction to particular inputs

• Nonfunctional Requirements

- Constraints on the services or functions offered by the system

--Include timing constraints, constraints on the development process and standards

--Apply to system as a whole

• Domain Requirements

--Requirements relate to specific application of the system

--Reflect characteristics and constraints of that system

FUNCTIONAL REQUIREMENTS

• Should be both complete and consistent

• Completeness

-- All services required by the user should be defined

• Consistent

-- Requirements should not have contradictory definition

• Difficult to achieve completeness and consistency for large system

NON-FUNCTIONALREQUIREMENTS

Types of Non-functional Requirements 1.Product Requirements

-Specify product behavior

-Include the following

- Usability
- Efficiency
- Reliability
- Portability

2. Organizational Requirements

--Derived from policies and procedures
--Include the following:

- Delivery
- Implementation
- Standard

3. External Requirements

-- Derived from factors external to the system and its development process
--Includes the following

- Interoperability
- Ethical
- Legislative

PROBLEMS FACED USING THE NATURAL LANGUAGE

1. Lack of clarity-- Leads to misunderstanding because of ambiguity of natural language
2. Confusion-- Due to over flexibility, sometime difficult to find whether requirements are same or distinct.
3. Amalgamation problem-- Difficult to modularize natural language requirements

STRUCTURED LANGUAGESPECIFICATION

- Requirements are written in a standard way
- Ensures degree of uniformity
- Provide templates to specify system requirements
- Include control constructs and graphical highlighting to partition the specification

SYSTEM REQUIREMENTS STANDARD FORM

- Function
- Description
- Inputs
- Source
- Outputs
- Destination
- Action
- Precondition
- Post condition
- Side effects

Interface Specification

- Working of new system must match with the existing system
- Interface provides this capability and precisely specified

Three types of interfaces

1. Procedural interface-- Used for calling the existing programs by the new programs 2.Data structures--Provide data passing from one sub-system to another 3.Representations of Data

-- Ordering of bits to match with the existing system
--Most common in real-time and embedded system

The Software Requirements document

The requirements document is the official statement of what is required of the system developers. Should include both a definition of user requirements and a specification of the system requirements. It is NOT a design document. As far as possible, it should set of WHAT the system should do rather than HOW it should do it

The Software Requirements document

Suggests that there are 6 requirements that requirement document should satisfy. It should

- specify only external system behavior
- Specify constraints on the implementation.
- Be easy to change
- Serve as reference tool for system maintainers
- Record forethought about the life cycle of the system.
- Characterize acceptable responses to undesired events

Purpose of SRS

- Communication between the Customer, Analyst, system developers, maintainers,
- firm foundation for the design phase
- support system testing activities
- Support project management and control
- controlling the evolution of the system

IEEE requirements standard

Defines a generic structure for a requirements document that must be instantiated for each specific system.

– Introduction.
– General description.
– Specific requirements.
– Appendices.
– Index.

IEEE requirements standard 1.Introduction

Purpose
Scope
Definitions, Acronyms and Abbreviations
References
Overview

2. General description

Product perspective
Product function summary
User characteristics
General constraints
Assumptions and dependencies

3. Specific Requirements

- Functional requirements

-External interface requirements

- Performance requirements
- Design constraints
- Attributes eg. security, availability, maintainability, transferability/conversion
- Other requirements
- Appendices
- Index

REQUIREMENTS ENGINEERING PROCESS

To create and maintain a system requirement document. The overall process includes four high level requirements engineering sub-processes:

1. Feasibility study

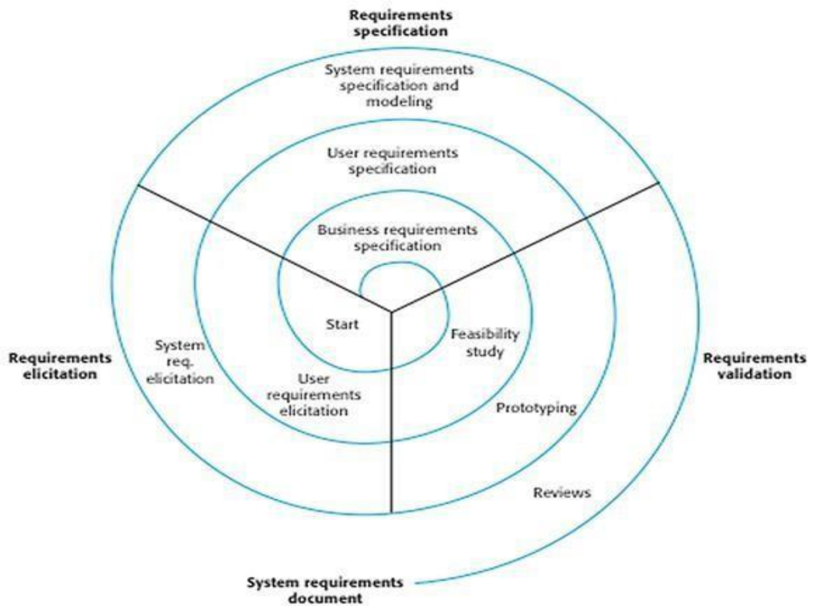
--Concerned with assessing whether the system is useful to the business 2.Elicitation and analysis
--Discovering requirements 3.Specifications
--Converting the requirements into a standard form 4.Validation
-- Checking that the requirements actually define the system that the customer wants

SPIRAL REPRESENTATION OF REQUIREMENTSENGINEERING PROCESS

Process represented as three stage activity. Activities are organized as an iterative process around a spiral. Early in the process, most effort will be spent on understanding high-level business and the use requirement. Later in the outer rings, more effort will be devoted to system requirements engineering and system modeling

Three level process consists of: 1.Requirements elicitation

2. Requirements specification
3. Requirements validation


Requirements specification
System requirements specification and modeling
User requirements specification
Business requirements specification
Start
Feasibility study
Requirements elicitation
System req. elicitation
User requirements elicitation
Prototyping
Requirements validation
Reviews
System requirements document

FEASIBILITY STUDIES

Starting point of the requirements engineering process

- Input: Set of preliminary business requirements, an outline description of the system and how the system is intended to support business processes
- Output: Feasibility report that recommends whether or not it is worth carrying out further Feasibility report answers a number of questions:

1. Does the system contribute to the overall objective
2. Can the system be implemented using the current technology and within given cost and schedule
3. Can the system be integrated with other system which are already in place.

REQUIREMENTS ELICITATION ANALYSIS

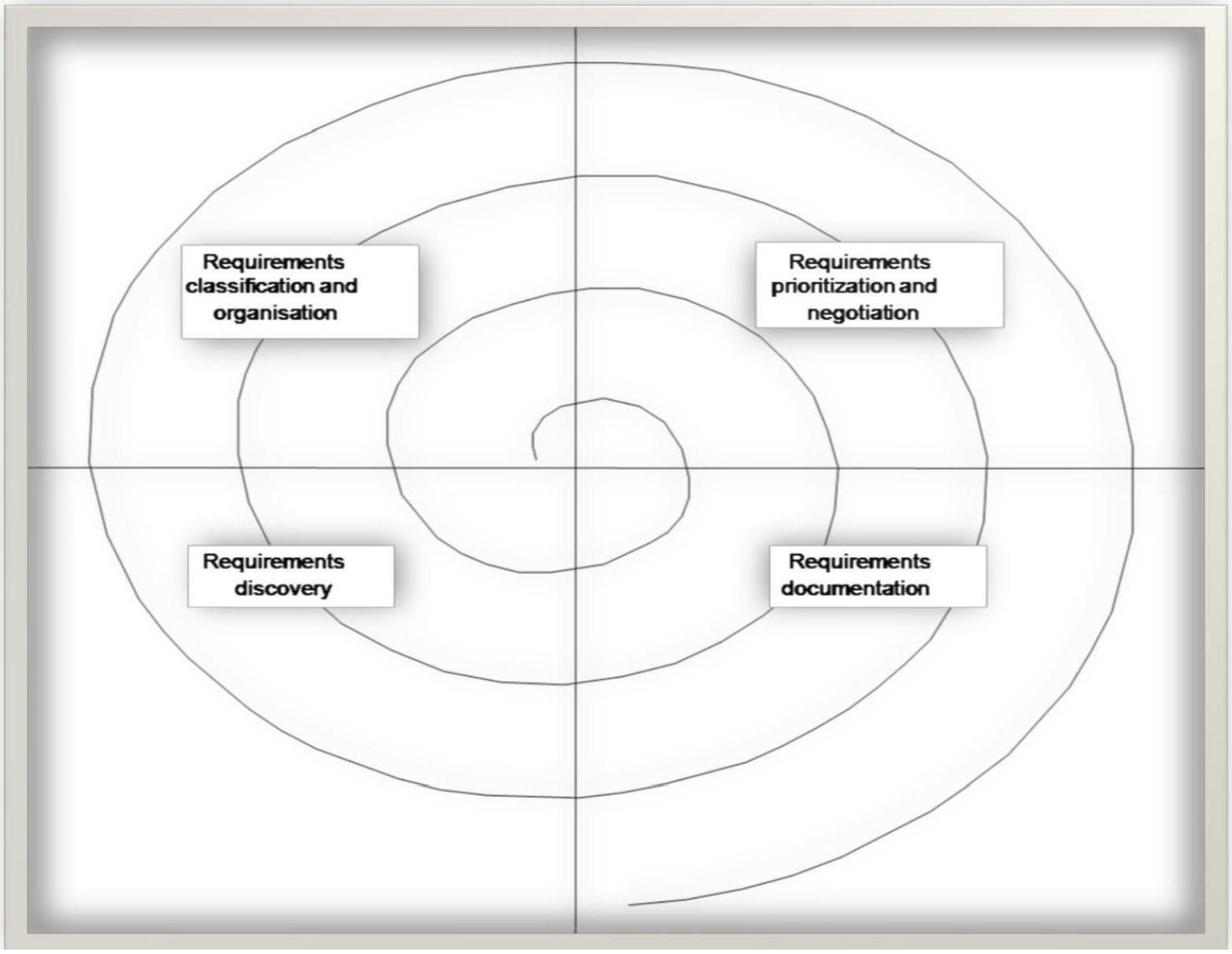
Involves a number of people in an organization.
Stakeholder definition-- Refers to any person or group who will be affected by the system directly or indirectly i.e. End-users, Engineers, business managers, domain experts.

Reasons why eliciting is difficult
1. Stakeholder often don't know what they want from the computer system. 2. Stakeholder expression of requirements in natural language is sometimes difficult toUnderstand.
3. Different stakeholders express requirements differently
4. Influences of political factors Change in requirements due to dynamic environments.

REQUIREMENTS ELICITATION PROCESS

Process activities
1. Requirement Discovery -- Interaction with stakeholder to collect their requirements including domain and documentation
2. Requirements classification and organization -- Coherent clustering of requirements from unstructured collection of requirements
3. Requirements prioritization and negotiation -- Assigning priority to requirements
--Resolves conflicting requirements through negotiation
4. Requirements documentation -- Requirements be documented and placed in the next round of spiral

The spiral representation of Requirements Engineering


Requirements classification and organisation
Requirements prioritization and negotiation
Requirements discovery
Requirements documentation

REQUIEMENTS DICOVERY TECHNIQUES

1. <u>View points</u> --Based on the viewpoints expressed by the stake holder
--Recognizes multiple perspectives and provides a framework for discovering conflicts in the requirements proposed by different stakeholders
Three Generic types of viewpoints
1. Interactor viewpoint--Represents people or other system that interact directly with the system
2. Indirect viewpoint--Stakeholders who influence the requirements, but don't use the system
3.Domain viewpoint--Requirements domain characteristics and constraints that influence the requirements.

2. <u>Interviewing</u>--Puts questions to stakeholders about the system that they use and the system to be developed. Requirements are derived from the answers.
Two types of interview
– Closed interviews where the stakeholders answer a pre-defined set of questions.
– Open interviews discuss a range of issues with the stakeholders for better understanding their needs.

Effective interviewers

a) Open-minded: no pre-conceived ideas

b) Prompter: prompt the interviewee to start discussion with a question or a proposal

3. Scenarios --Easier to relate to real life examples than to abstract description. Starts with an outline of the interaction and during elicitation, details are added to create a complete description of that interaction

Scenario includes:

- 1. Description at the start of the scenario
- 2. Description of normal flow of the event
- 3. Description of what can go wrong and how this is handled
- 4.Information about other activities parallel to the scenario
- 5.Description of the system state when the scenario finishes

LIBSYS scenario

- **Initial assumption**: The user has logged on to the LIBSYS system and has located the journal containing the copy of the article.
- **Normal**: The user selects the article to be copied. He or she is then prompted by the system to either provide subscriber information for the journal or to indicate how they will pay for the article. Alternative payment methods are by credit card or by quoting an organizational account number.
- The user is then asked to fill in a copyright form that maintains details of the transaction and they then submit this to the LIBSYS system.
- The copyright form is checked and, if OK, the PDF version of the article is downloaded to the LIBSYS working area on the user's computer and the user is informed that it is available. The user is asked to select a printer and a copy of the article is printed

LIBSYS scenario

- **What can go wrong**: The user may fail to fill in the copyright form correctly. In this case, the form should be re-presented to the user for correction. If the resubmitted form is still incorrect then the user's request for the article is rejected.
- The payment may be rejected by the system. The user's request for the article is rejected.
- The article download may fail. Retry until successful or the user terminates the session..
- **Other activities**: Simultaneous downloads of other articles.
- **System state on completion**: User is logged on. The downloaded article has been deleted from LIBSYS workspace if it has been flagged as print-only.

4. Use cases -- scenario based technique for requirement elicitation. A fundamental feature of UML, notation for describing object-oriented system models. Identifies a type of interaction and the actors involved. Sequence diagrams are used to add information to a Use case

Article printing use-case Article printing

LIBSYS use cases Article printing Article search
User administration Supplier Catalogue services LibraryUser Library Staff

REQUIREMENTS VALIDATION

Concerned with showing that the requirements define the system that the customer wants. Important because errors in requirements can lead to extensive rework cost

Validation checks

1. Validity checks --Verification that the system performs the intended function by the user 2.Consistency check --Requirements should not conflict
3. Completeness checks --Includes requirements which define all functions and constraints intended bythe system user
4. Realism checks --Ensures that the requirements can be actually implemented
5. Verifiability -- Testable to avoid disputes between customer and developer.

VALIDATION TECHNIQUES 1.REQUIREMENTS REVIEWS

Reviewers check the following:

(a) Verifiability: Testable
(b) Comprehensibility
(c) Traceability
(d) Adaptability 2.PROTOTYPING

3. TEST-CASE GENERATION

Requirements management

Requirements are likely to change for large software systems and as such requirements management process is required to handle changes.

Reasons for requirements changes

(a) Diverse Users community where users have different requirements and priorities
(b) System customers and end users are different
(c) Change in the business and technical environment after installation Two classes of requirements
(a) Enduring requirements: Relatively stable requirements
(b) Volatile requirements: Likely to change during system development process or during operation

Requirements management planning

An essential first stage in requirement management process. Planning process consists of the following

1. Requirements identification -- Each requirement must have unique tag for cross reference and traceability
2. Change management process -- Set of activities that assess the impact and cost of changes 3.Traceability policy -- A matrix showing links between requirements and other elements of software development
4. CASE tool support --Automatic tool to improve efficiency of change management process. Automatedtools are required for requirements storage, change management and traceability management

Traceability

Maintains three types of traceability information.

1. Source traceability--Links the requirements to the stakeholders
2. Requirements traceability--Links dependent requirements within the requirements document
3. Design traceability-- Links from the requirements to the design module

Req. id	**1.1**	**1.2**	**1.3**	**2.1**	**2.2**	**2.3**	**3.1**	**3.2**
1.1		D	R					
1.2			D			D		D
1.3	R			R				
2.1			R		D			D
2.2								D
2.3		R		D				
3.1								R
3.2							R	

A traceability matrix Requirements change management
consists of three principal stages:
1. Problem analysis and change specification-- Process starts with a specific change proposal and analysed to verify that it is valid
2. Change analysis and costing--Impact analysis in terms of cost, time and risks
3. Change implementation--Carrying out the changes in requirements document, system design and its implementation

<u>SYSTEM MODELS</u>
Used in analysis process to develop understanding of the existing system or new system. Excludes details. An abstraction of the system
Types of system models 1.Context models
2. Behavioural models 3.Data models 4.Object models 5.Structured models

<u>CONTEXT MODELS</u>
A type of architectural model. Consists of sub-systems that make up an entire system First step: To identify the subsystem.
Represent the high level architectural model as simple block diagram
- Depict each sub system a named rectangle
- Lines between rectangles indicate associations between subsystems Disadvantages

--Concerned with system environment only, doesn't take into account other systems, which may take data or give data to the model

The context of an ATM system consists of the following Auto-teller system
Security system Maintenance system Account data base Usage database
Branch accounting system Branch counter system

<u>Behavioral models</u>
Describes the overall behaviour of a system. Two types of behavioural model
1.Data Flow models 2.State machine models

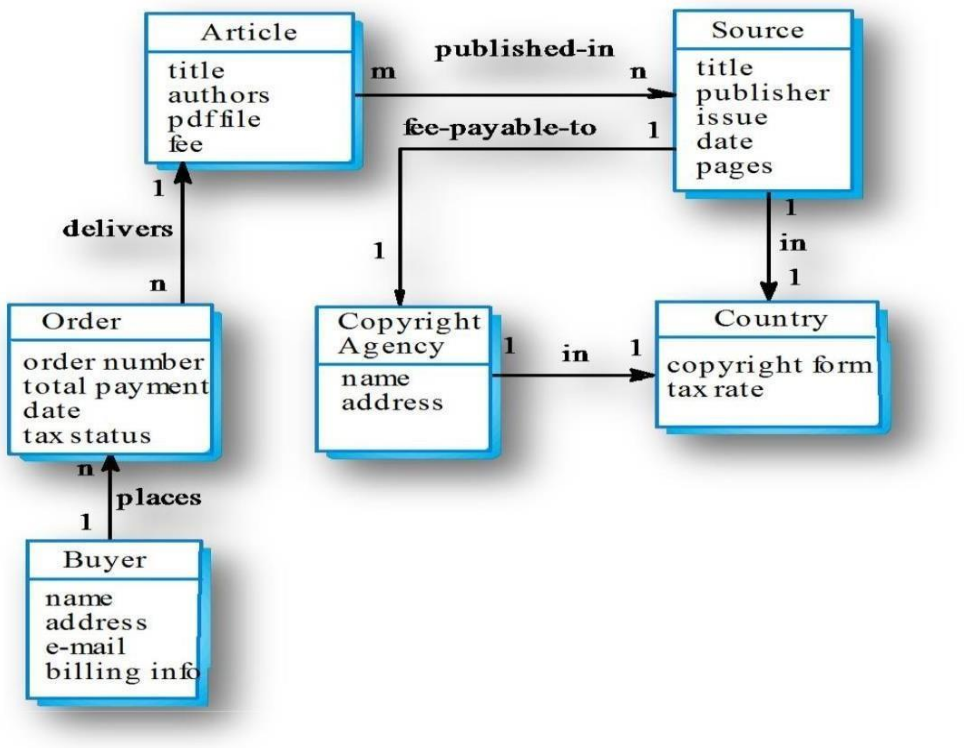
<u>Data flow models</u> --Concentrate on the flow of data and functional transformation on that data. Show the processing of data and its flow through a sequence of processing steps. Help analyst understand what is going on

Advantages
-- Simple and easily understandable
-- Useful during analysis of requirements

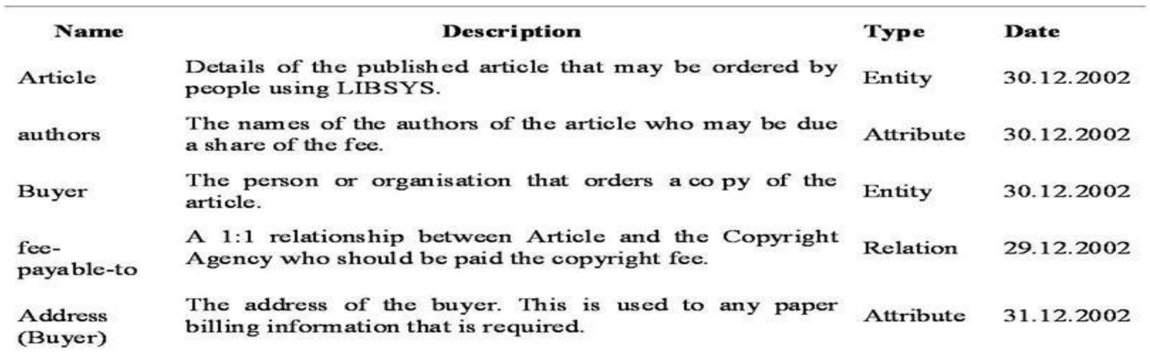
State machine models
Describe how a system responds to internal or external events. Shows system states and events that cause transition from one state to another. Does not show the flow of data within the system. Used formodeling of real time systems
Exp: Microwave oven
Assumes that at any time, the system is in one of a number of possible states. Stimulus triggers atransition from on state to another state
Disadvantage
-- Number of possible states increases rapidly for large system models

<u>DATA MODELS</u>
Used to describe the logical structure of data processed by the system. An entity-relation-attribute model sets out the entities in the system, the relationships between these entities and the entity attributes.Widely used in database design. Can readily be implemented using relational databases. No specific notation provided in the UML but objects and associations can be used.

Library semantic model



Data dictionary entries



Name	Description	Type	Date
Article	Details of the published article that may be ordered by people using LIBSYS.	Entity	30.12.2002
authors	The names of the authors of the article who may be due a share of the fee.	Attribute	30.12.2002
Buyer	The person or organisation that orders a co py of the article.	Entity	30.12.2002
fee-payable-to	A 1:1 relationship between Article and the Copyright Agency who should be paid the copyright fee.	Relation	29.12.2002
Address (Buyer)	The address of the buyer. This is used to any paper billing information that is required.	Attribute	31.12.2002

OBJECT MODELS
An object oriented approach is commonly used for interactive systems development. Expresses thesystems requirements using objects and developing the system in an object oriented PL such as c++A object class: An abstraction over a set of objects that identifies common attributes. Objects are instances of object class. Many objects may be created from a single class.
Analysis process
-- Identifies objects and object classes Object class in UML
--Represented as a vertically oriented rectangle with three sections
(a) The name of the object class in the top section
(b) The class attributes in the middle section
(c) The operations associated with the object class are in lower section.

OBJECT MODELS INHERITANCE MODELS
A type of object oriented model which involves in object classes attributes. Arranges classes into aninheritance hierarchy with the most general object class at the top of hierarchy Specialized objects inherit their attributes and services
UML notation
-- Inheritance is shown upward rather than downward
--Single Inheritance: Every object class inherits its attributes and operations from a single parent class
--Multiple Inheritance: A class of several of several parents.

OBJECT MODELS OBJECT AGGREGATION
Some objects are grouping of other objects. An aggregate of a set of other objects. The classes representing these objects may be modeled using an object aggregation model A diamond shape on thesource of the link represents the composition.

OBJECT-BEHAVIORAL MODEL
-- Shows the operations provided by the objects
-- Sequence diagram of UML can be used for behavioral modeling