Software Engineering

Unit II

System Requirements

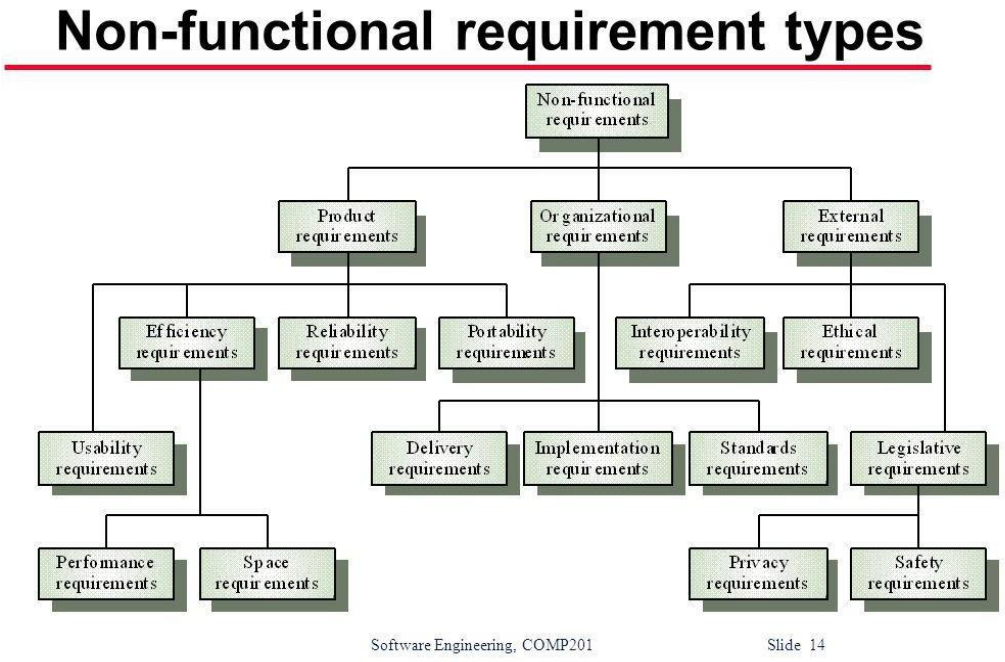
- Requirements analysis is very critical process that enables the success of a system or software project to be assessed.
- Requirements are generally split into two types:
  - *Functional* and
  - *Non-functional requirements*.

Functional Requirements:

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

NON- FUNCTIONAL REQUIREMENTS:

- NON- FUNCTIONAL REQUIREMENTS, As the name suggests, are requirements that are not directly concerned with the specific functions delivered by the system.
- 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.



The type of non-functional requirements are:

- **Product requirements**: These requirements specify product behavior.
- **Organizational requirements**: These requirements are derived from policies and procedures in the customer's and developer's organization.
- **External requirements**: This broad heading covers all requirements that are derived from factors external to the system and its development process.

USER REQUIREMENTS:

- The user requirements for a system should describe the functional and non-functional requirements so that they are understandable by system users without detailed technical knowledge.
- If you are writing user requirements , you should not use software jargons, structured notations or formal notations or describe the requirements by describing the system implementation.

- You should write user requirements in simple language , with simple tables and forms and intuitive diagrams.

Various problems can arise when requirements are written in natural language sentences in a text document:

1. **Lack of clarity**: It is some times difficult to use language in a precise and unambiguous way without making the document wordy and difficult to read.
2. **Requirements confusion**: functional requirements, non- functional requirements, system goals and design information may not be clarity distinguished.
3. **Requirements amalgamation**: several different requirements may be expressed together as a single requirement.

System Requirements:

- System requirements are expanded version of the user requirements that are used by software engineers as the starting point for the system design.
- They add detail and explain how the user requirements should be provided by the system.
- System requirements should simply describe the external behaviour of the system and its operational constraints.

Natural language is often used to write system requirements specifications as well as user requirements. However because system requirements are more detailed than user requirements, natural language specifications can be confusing and hard to understand:

1. Natural language understanding relies on the specification readers and writers using the same words for the same concept. This leads to misunderstanding because of the ambiguity of natural language.
2. A Natural language requirements specification is overflexible. You can say the same thing in completely different ways.
3. There is no easy way to modularise Natural language requirements. It may be difficult to find all related requirements.

Structured language specifications:

Structured natural language is a way of writing system requirements where the freedom of the requirements written is limited and all requirements are written in a standard way.

When a standard form is used for specifying functional requirements, the following information should be included:

- Description of the function or entry being specified.
- Description of its inputs and where these come form
- Description of its outputs and where these go to
- Indication of what other entities are used
- Description of the action to be taken
- If a functional approach is used, a pre-condition setting out what must be true before the function is called and post-condition specifying what is true after the function is called.
- Description of the of the operation.

Interface requirements:

- Almost all software systems must operate with existing systems that have already been implemented and installed in an environment.
- If the new system and the existing system must work together, The interfaces of existing systems have to be precisely specified.
- These specifications should be defined early in the process and included in the requirements document.

There are three types of interface that may have to be:

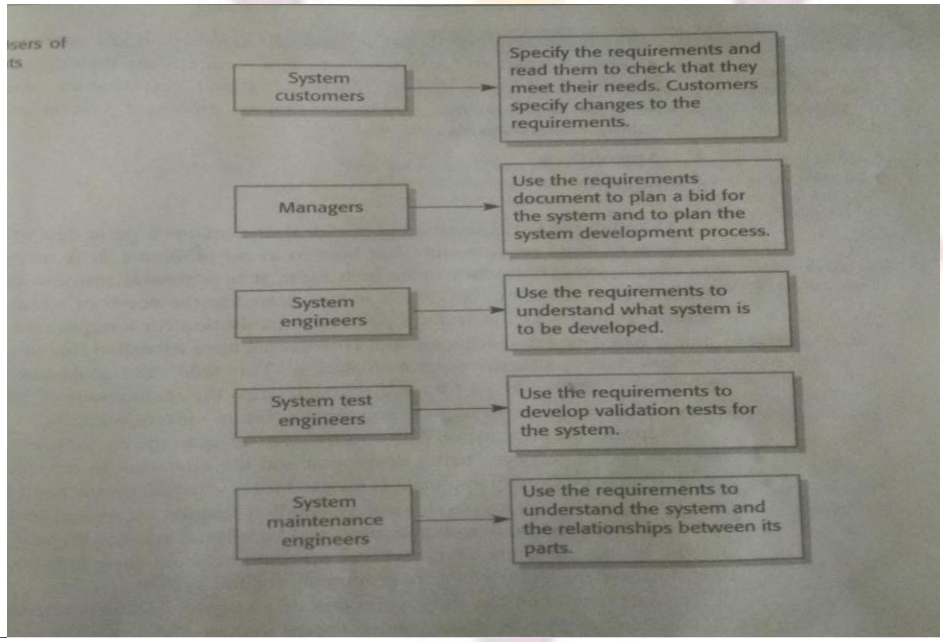
- Procedural interface: Where existing programs or sub-systems offer a range of services that accessed by calling interface procedures. These interfaces are sometimes called Application Programming Interfaces(APIs)
- Data structures: That are passed from one sub-system to another. Graphical data models are the best notations for this type of description.
- Representation of data: That have been established for an existing sub-system. These interfaces are most common in embedded, real-time-system. Some programming languages such as Ada support this level of specification.

SOFTWARE REQUIREMENTS DOCUMENT:

The software requirements document(sometimes called the software requirements specification or SRS) is the official statement of what the system developers should implement.

It should include both the user requirements for a system and a detailed specification of the system requirements.

The users of a requirements document:



Characteristics of a good SRS

An SRS should be:

a) Correct
b) Unambiguous
c) Complete
d) Consistent
e) Ranked for importance and/or stability
f) Verifiable
g) Modifiable
h) Traceable

IEEE Standards suggests the following structure for requirements documents:

1. Introduction
1.1 Purpose of the requirements document
1.2 Scope of the product
1.3 Definitions, acronyms and abbreviations
1.4 References
1.5 Overview of the remainder of the document
2. General description
2.1 Product perspective
2.2 Product functions
2.3 User characteristics
2.4 General constraints
2.5 Assumptions and dependencies
3. **Specific requirements**, covering functional, non-functional and interface requirements. This is obviously the most substantial part of the document but because of the wide variability in organizational practice, it is not appropriate to define a standard structure for this section. The requirements may document external interfaces, describe system functionality and performance, and specify logical database requirements, design constraints, emergent system properties and quality characteristics.

4. **Appendices**

5. **Index:** Although the IEEE standard is not ideal, it contains a great deal of good advice on how to write requirements and how to avoid problems. It is too general to be an organizational standard in its own right. It is a general framework that can be tailored and adapted to define a standard geared to the needs of a particular organization (6).

Figure 1. The SRS structure according the IEEE 830-1998 standard

REQUIREMENTS ENGINEERING PROCESS

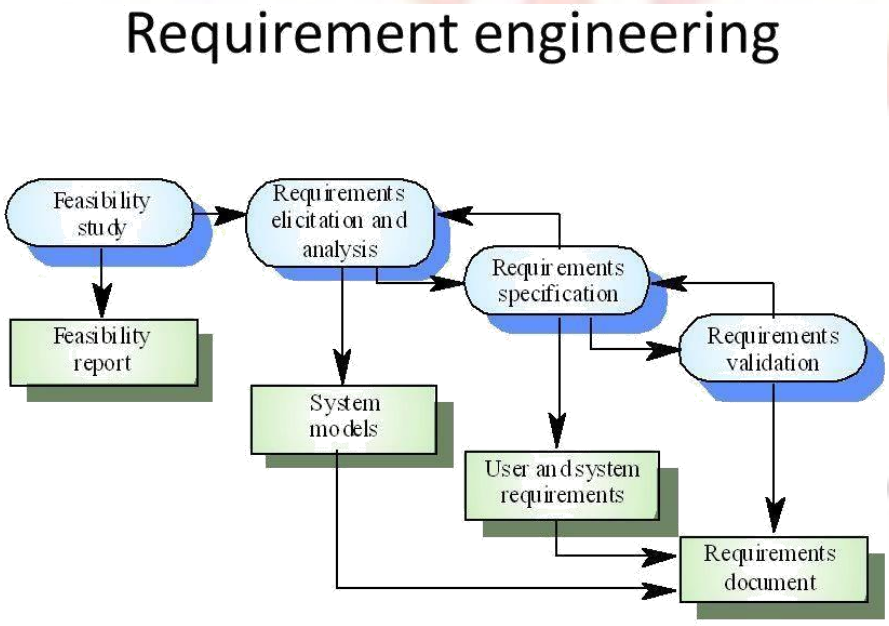
To create and maintain a system requirement document. The overall process includes four high level requirements engineering sub-processes:

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

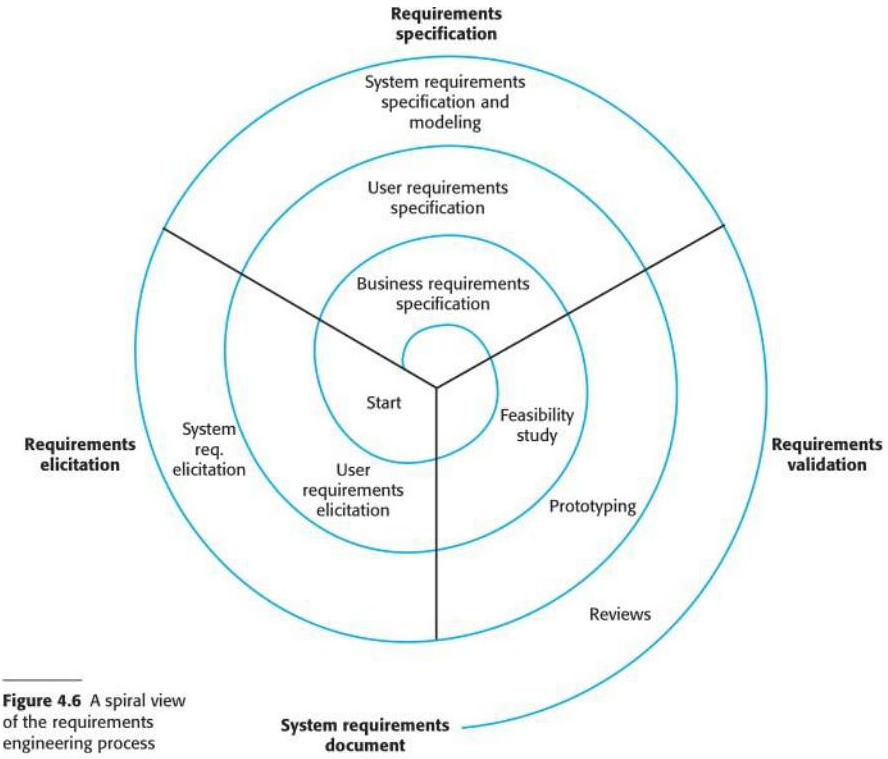


Figure 4.6 A spiral view of the requirements engineering process

FEASIBILITY STUDIES

The requirement engineering process should starts with a feasibility study.
Starting point of the requirements engineering process

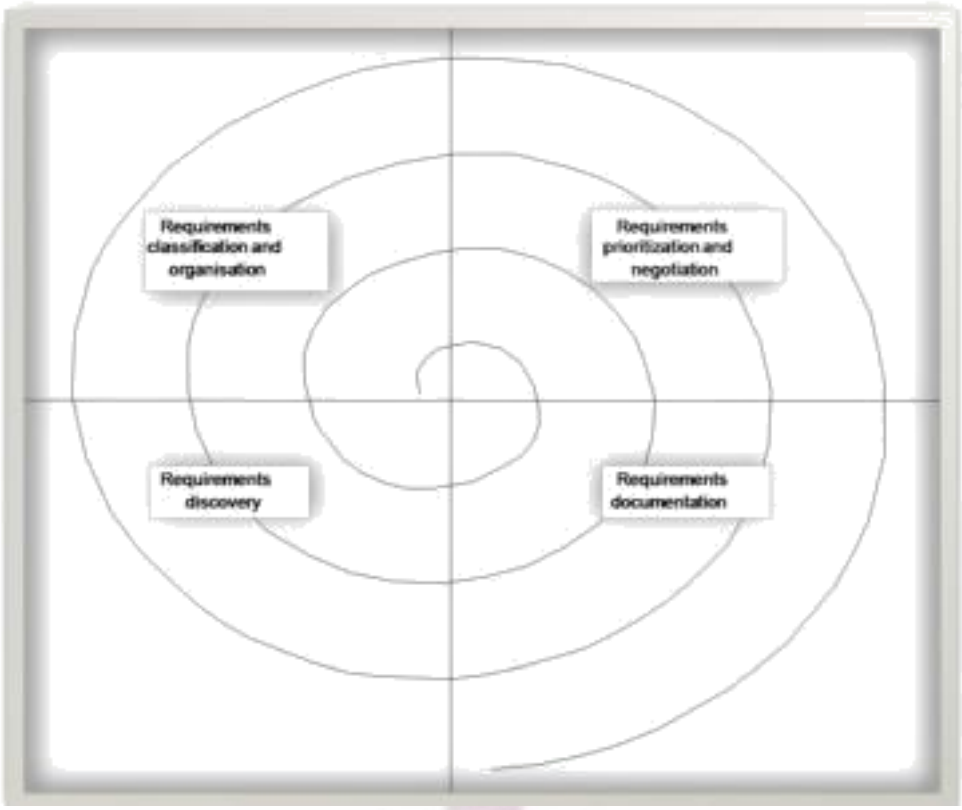
- **Input:** Set of preliminary business requirements, an outline description of the system and howthe system is intended to support business processes
- **Output:** Feasibility report that recommends whether or not it is worth carrying out further Feasibility report answers a number of questions:
  - Does the system contribute to the overall objective:
  - Can the system be implemented using the current technology and within given cost and schedule
  - 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 to Understand.
3. Different stakeholders express requirements differently

4. Influences of political factors Change in requirements due to dynamic environments.
5. the economic and business environment in which the analysis takes place is dynamic



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 ofspiral

REQUIEMENTS DICOVERY TECHNIQUES

1. **View points** --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

- **Interactor viewpoint**--Represents people or other system that interact directly with the system
- **Indirect viewpoint**--Stakeholders who influence the requirements, but don't use the system
- **Domain viewpoint**--Requirements domain characteristics and constraints that influence the requirements.

2. **Interviewing**--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

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.

During the requirement validation process, checks should be carried out on the requirements in the requirements document.

These checks include:

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

VALIDATION TECHNIQUES:

- Requirements reviews – the requirements are analysed systematically by a team of reviewers.

  Reviewers check the following:

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

- Prototyping- an executable model of the system is demonstrated to end- users and customers.
- Test- case generation – Requirements should be testable.

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
d) Enduring requirements: Relatively stable requirements
e) 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. Automated tools are required for requirements storage, change management and traceability management

Traceability

Maintains three types of traceability information.

- **Source traceability**--Links the requirements to the stakeholders
- **Requirements traceability**--Links dependent requirements within the requirements document
- **Design traceability**-- Links from the requirements to the design module

CASE tools:

- **Requirement storage** – should maintained in a secure, managed data store.
- **Change management** – simplified if active tool support is available.
- **Traceability management** – allows related requirements to be discovered.

Requirements change management:

There are Three principle stages to a change management process:

1. **Problem analysis and change specification**-- Process starts with a specific change proposaland 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

SYSTEM MODELS

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. Behavioral models 3.Data models 4.Object models 5.Structured models

CONTEXT MODELS

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

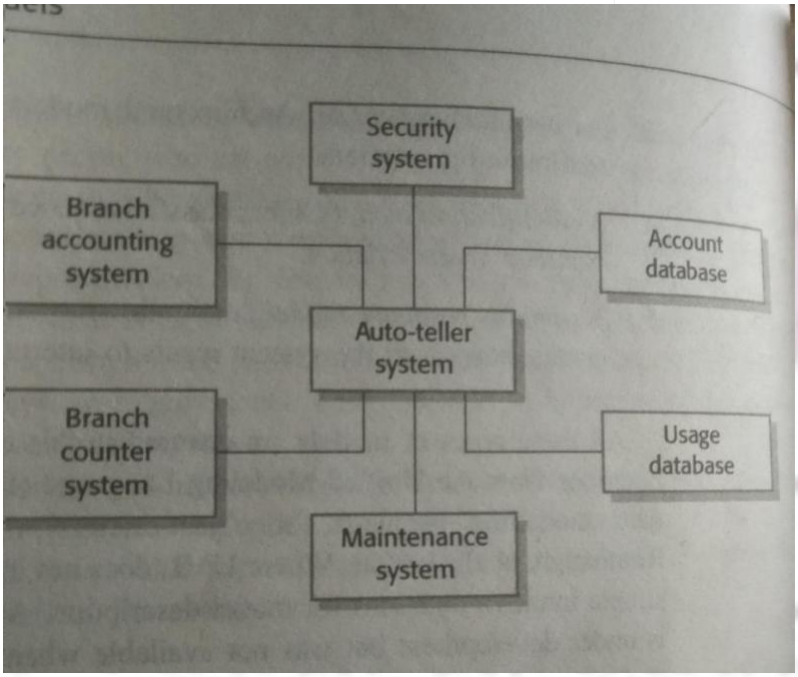
• Lines between rectangles indicate associations between subsystems Disadvantages

--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



Behavioral models

Describes the overall behavior of a system. Two types of behavioral model

1. Data Flow models 2.State machine models

Data flow models --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 for modeling 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 a

transition from on state to another state

Disadvantage

-- Number of possible states increases rapidly for large system models

DATA MODELS

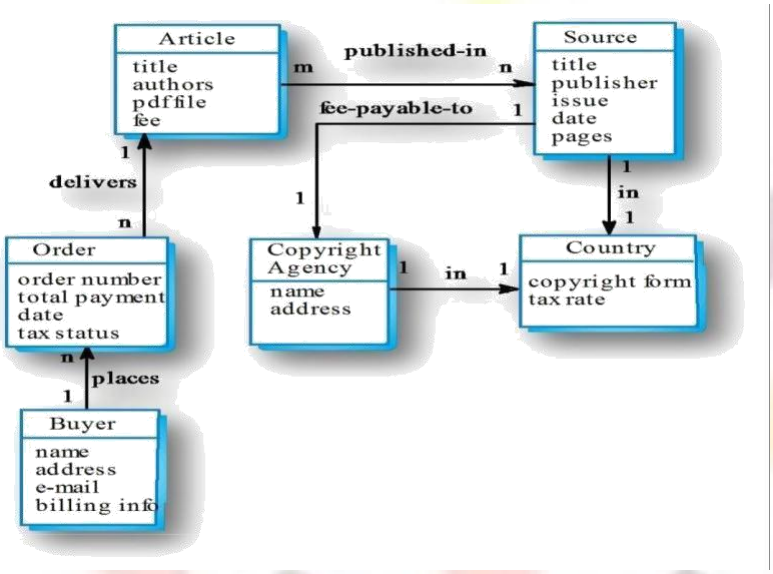
Used to describe the logical structure of data processed by the system. An entity-relation-attribute

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.

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 the systems 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
The name of the object class in the top section

The class attributes in the middle section

The operations associated with the object class are in lower section.

Object name
Class attribute
Operation()

OBJECT MODELS INHERITANCE MODELS

A type of object oriented model which involves in object classes attributes.
Arranges classes into an inheritance 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 the source 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