# Software Engineering

## Unit2

## Requirements Analysis and Specification

Software requirements: functional and non-functional - security requirements - user requirements - system requirements - software requirements document; Requirements engineering process: Feasibility studies - requirements elicitation and analysis - requirements validation - requirements management; classical analysis: structured system analysis; requirement modelling tools.

### * Requirements Engineering

→ establishing the services that the customer requires from a system and the constraints under which it operates and is developed.

### * Requirements

→ Requirements are descriptions of the system services and constraints that are generated during the requirements engineering process.

→ They can be a high-level abstract statement of a service or can be a specific system constraint, or a detailed mathematical functional specification.

→ Requirements may serve a dual function:

(i) may be the basis for a bid for a contract - can be open to interpretation.

(ii) may be the basis of the contract itself - must be defined in detail.

* Types of Requirements

User vs. System requirements

Functional vs. non-functional requirements

A. User vs System Requirements

User Requirements - includes statements in natural language + diagrams of the services the system provides w/ its constraints
- written for customers

System Requirements - A structured document giving detailed descriptions of the system's functions, services or operational constraint
- defines what should be implemented so it may be part of a contract between a client & contractor.

## 6. Functional vs. Non-Functional Requirements

(1) Functional Requirements - statements of services the system should provide

- how the system should react to particular inputs, and how the system should behave in particular situations

→ It describes the functionality of the service itself.

→ This would depend upon the type of software, expected users and the type of system where the software is used.

→ should describe system services in detail

→ Functional requirements also have the following facets:

(i) Requirements Imprecision - arises when the requirements are not properly stated

(ii) - Ambiguous requirements may be interpreted in different ways by developers and users

- It depends upon 'appropriate viewers'. To understand

  - user intention - make a special purpose viewer for each document type
  - developer interaction - provide a text viewer that shows the content of the document

(ii) Requirements completeness and consistency - In principle, requirements should be complete & consistent

- Complete - should include descriptions of all the facilities required

- Consistent - should be no conflicts or contradictions in the descriptions of the system facilities

- In practice, it is impossible to produce a complete and consistent requirements document

Examples of Functional Requirements - Consider a LIBSYS system, which is a library system that provides a single interface to a large database of books/ articles in different libraries, Some functional requirements may be :

(i) search for books

(ii) appropriate viewers can read documents

(iii) each order is associated w/ a unique identifier, which can be copied into the account's permanent storage area (database)

② Non-functional requirements

→ These define system properities and constraints like reliability, respone time and storage requirements.

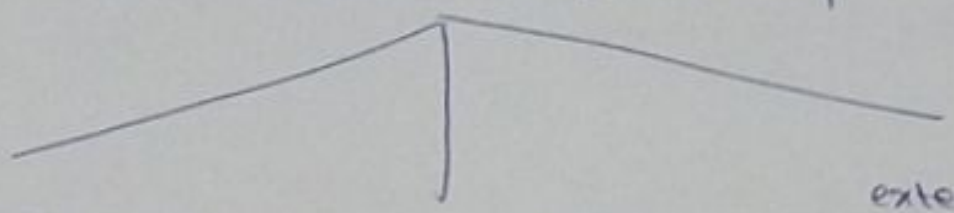
→ Some constraints may be I/o device capability, system representation etc.

→ There also may be mandates for a specific CASE system, programming language.

→ non-functional requirements may be more critical than functional

requirements. Otherwise, the system may be useless

→ Non-functional requirements can be classified as follows:




1. Product Requirements - requirements which specify that the delivered product must behave in a particular way in terms of execution speed & reliability

Example - a constraint specified that the application must be implemented as simple HTML without frames / Java applets

2. Organizational Requirements - a consequence of organizational policies & procedures.

- requirements to conform to certain process standards, implementation requirements

Example - system development process and deliverables must conform to that defined in 'XYZCo-SP-STAN-95'
↙ some standard

3. External Requirements - arises from factors which are external to the system and its development process

- includes interoperability requirements, legislative requirement

Example - the system should not disclose any personal information about customers apart from name & ref. no to system operators.

→ Other facets in non-functional requirement analysis include

(i) System goals and verifiable non-functional requirement

~~(ii) a verifiable non-functional requirement~~

(ii) requirements interaction

(i) System goal, - refers to a very general intention of the user and verifiable non-functional req - verifiable non-functional requirements, are statements using some measure that can be objectively tested

- for eg. a system goal may be increase the ease of use of a system, such that user errors are minimized, and a verifiable non-functional requirement may be that controller is able to use all the system functions after 2 hrs. of training.

(ii) Requirements interaction - there may be conflicts between non-functional requirements in complex systems

Example - in spacecraft systems, one must aim to minimize the no. of separate chips to reduce weight, but to minimize power consumption lower power chips must be use. Using lower power chips may mean that more chips have to be used. The more critical requirement must be studied

## (3) Domain Requirements

→ derived from the application domain & describe system characteristics and features that reflect the domain

→ domain requirements may be new functional requirements, constraints on existing requirements or define specific computations

→ System may be unworkable if domain requirements are not satisfied.

Example - domain requirements in a library system may be that there must be a standard UI

→ copyright restrictions must be handled appropriately.

→ Domain requirements have the following problems:

(i) understandability - requirements are written in the language of the application domain, not understood by software engineers

(ii) implicitness - domain specialists understand the domain so well that do not think of making the domain requirements explicit.

## * Requirements Document

→ the official statement of what is required of the system developers

→ should include user requirements & a specification of the system requirements

→ It is NOT a design document - should specify WHAT the system should do rather than HOW to do it.

→ The IEEE requirements standards defines a generic structure for a requirements document. It must include:

(i) Introduction

(ii) General descriptor

(iii) Specific req.

(iv) Appendices

(v) Index

* Requirements Engineering

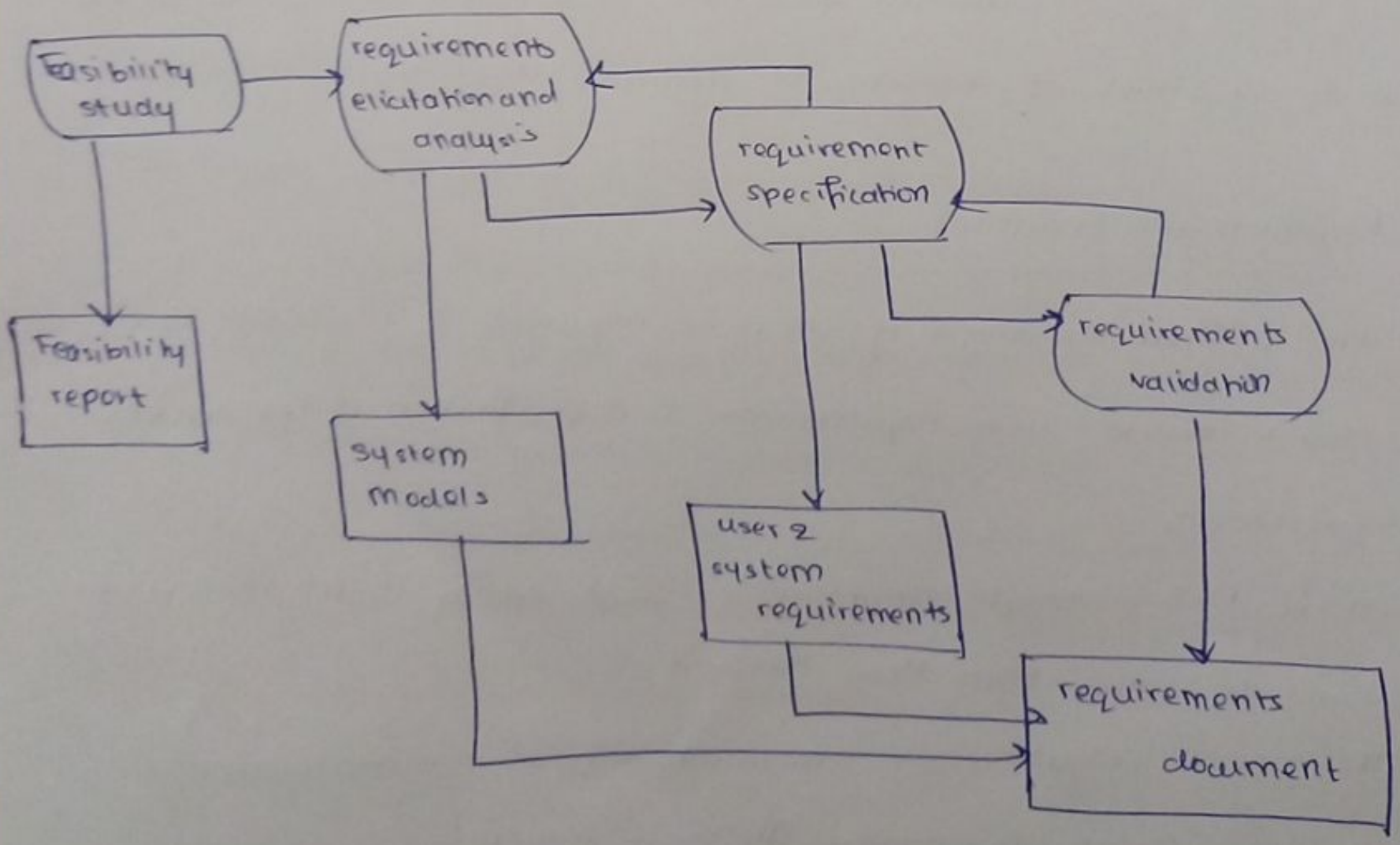
→ The general process of requirements engineering is as follows:-

(i) requirements elicitation - useful to business

(ii) requirements analysis - discovering requirements

(iii) requirement validation - converting requirement into some standard form

(iv) requirements management - checking



## (1) Feasibility Studies

→ decides whether the proposed system is worthwhile

→ It checks:

(i) if the system contributes to organizational objectives

(ii) if the system can be engineered using current technology & within budget

(iii) if the system can be integrated w/ other systems that are used.

→ Feasibility studies are implemented as follows:

- information assessment
- information collection
- report writing

→ The people who are in the org. can be asked

(i) what their current problems

(ii) if the system wasn't implemented

(iii) how the proposed system would help

(iv) what the problems w/ integration would be

## (2) Requirements Elicitation & Analysis

→ Involves technical staff working with customers to find out about the application domain, the services the system should

Provide, and the system's operational constraints

→ may include meetings w/ stakeholders - end-users, managers, engineers, domain experts.

Issues with requirement analysis

→ stakeholders don't know what they need

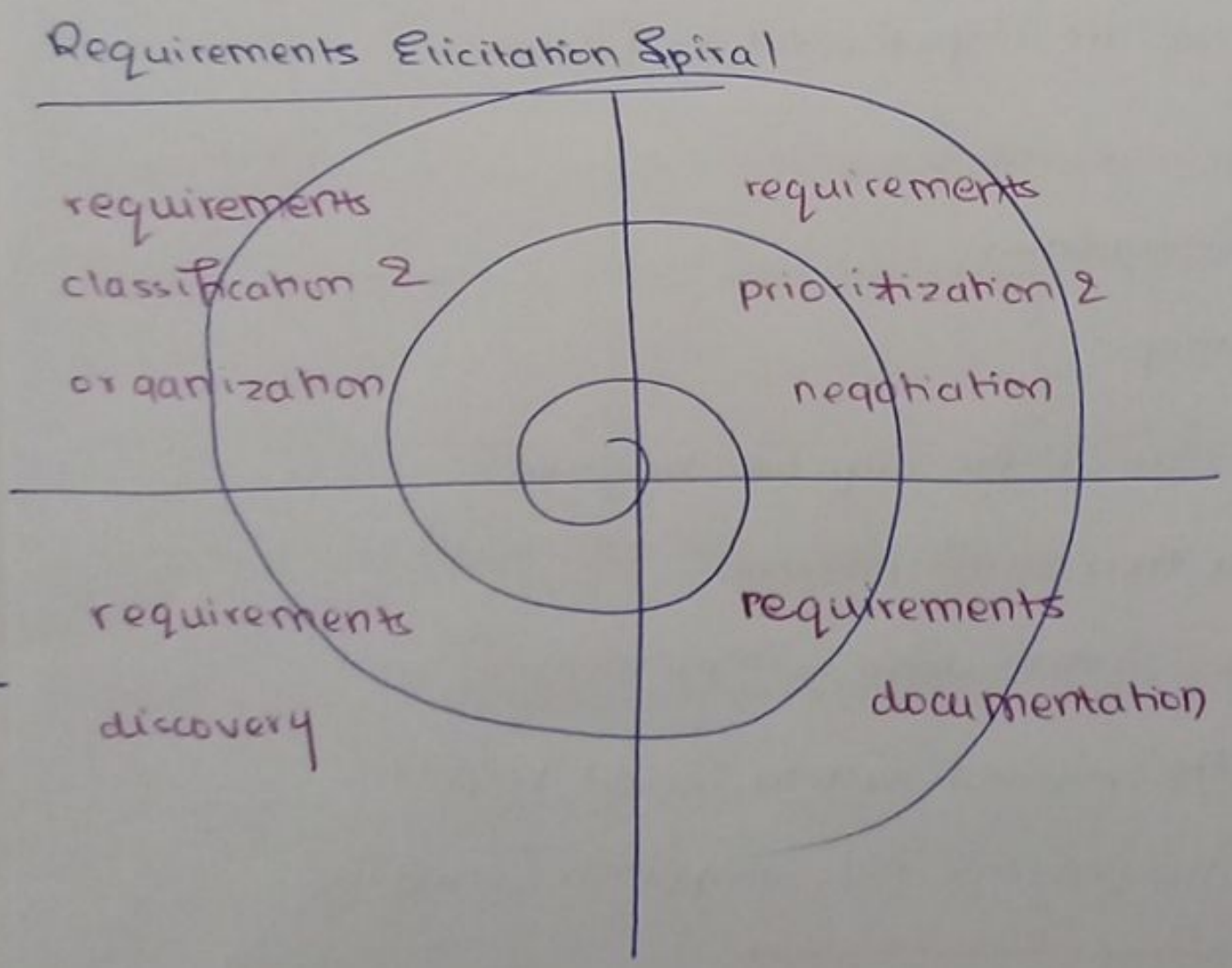
→ they express requirements in their own terms

→ may have conflicting requirements

→ may be influenced by organizational & political factors

→ requirements keep changing

Requirements Elicitation Spiral




(i) Requirements Discovery

→ gather info. about the proposed & existing systems

→ sources of info are documentation, stakeholders & specifications of

similar systems.

## (ii) Requirements Classification & Organization

→ requirements can be classified by analyzing different viewpoints

→ Viewpoints represent the perspectives of different stakeholders.

→ This multi-perspective analysis is important as there is no single way to analyze system requirements

→ Some types of viewpoints include:

(i) Inspector viewpoints - people or other systems that interact directly with the system

(ii) Indirect viewpoints - stakeholders who do not use the system themselves, but who influence the requirements

(iii) Domain viewpoints - domain characteristics & constraints that influence the requirements

→ Viewpoints are identified as follows:

- providers & receivers of system services
- regulations & standards
- engineers who have to develop & maintain the system

## (3) Requirements Validation

### (iii) Requirements validation, prioritizing & negotiating

→ concerned with demonstrating that the requirements define the system that the customer really wants

→ requirements error costs are high so validation is very imp.

→ The following aspects are considered during requirements

requirements checking

(i) validity
(ii) consistency
(iii) completeness
(iv) reality
(v) verifiability

→ The priority of requirements from diff. view points may also change during the development process

→ Requirements can be validated by

(i) requirements reviews – good communication can revolve problems early on
– conduct systematic manual analysis
– both client & staff should be present

(ii) prototyping – using an executable model to check requirements

(iii) test case generation – developing tests for requirements to check testability

→ also checks for

- verifiability
- adaptability
- traceability
- comprehensibility

## 4. Requirements Management

### (iv) Requirements management and documentation

→ requirements management involves managing changing requirements during the engineering & development process.

→ Requirements are inevitably incomplete and inconsistent (13) as:

(i) new requirements emerge during the process

(ii) different viewpoints have different requirements and are contradictory in nature.

## Enduring vs. Volatile Requirements

Enduring – stable requirements derived from the core activity of the customer organization (doctors at hospitals)

Volatile – requirements which change during development or when the system is in use. (eg. health-care policy requirements)

## Requirements Management Planning

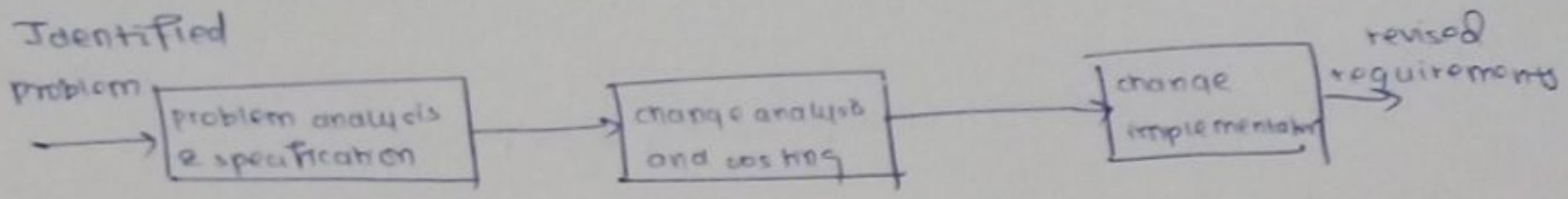
To handle changing requirements, the following steps have to be followed:

① Requirements identification

② A change management process – refers to the process followed when analyzing a requirements change".

– This must be applied to all proposed changes to the requirements

– It involves: (i) problem analysis
(ii) change analysis & costing
(iii) change implementation




(3) Traceability policies

- The amount of information about requirements relationships that is maintained. It can be:

(i) source traceability - links from requirements to stakeholders who proposed the requirements

(ii) requirements traceability - links between dependent requirements

(iii) design traceability - links from the requirements to the design

→ can be represented via a traceability matrix

(4) CASE Tool Support - The tool support required to help manage requirements changes

→ The CASE tool support encompasses the following:

(i) requirements storage - requirements should be managed in a secure managed data store

(ii) change management

(iii) traceability management

## *Ethnography

→ A social scientist, also called a cultural anthropologist who spends a considerable time observing, analyzing & recording how people actually work

→ They may observe social & organizational factors

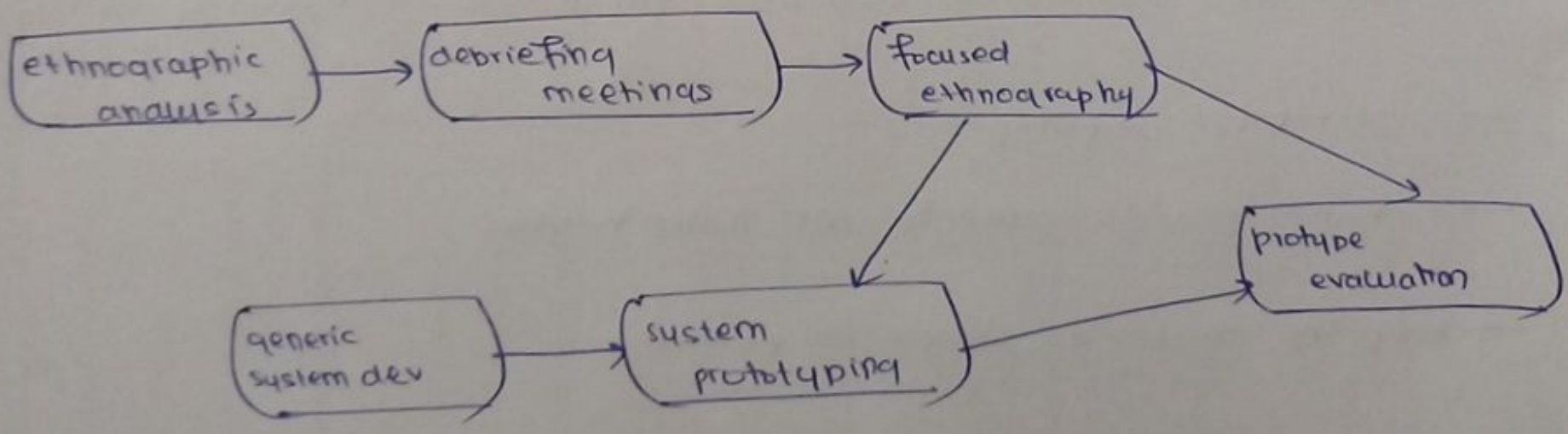
→ Ethnographic studies have shown that work is usually richer and more complex than suggested by simple system models

## Focused Ethnography

→ combines ethnography with prototyping

→ prototype development results in unanswered questions - the next phase of ethnographic analysis focuses on these problem areas

→ The problem with ethnography is that it studies existing practices ~~would~~ which may have some historical basis which is no longer relevant.




## Scope of ethnography

leads to -

(i) requirements derived from the way people actually work rather than the way in which process definitions suggest that they ought to.

(ii) requirements that are derived from cooperation & awareness of other people's activities

## * Structured System Analysis

### Requirement Modeling

Requirement modeling helps in the following:

(i) describes what the customer required

(ii) establishes a basis for the creation of a software design

(iii) define a set of requirements that can be validated once the software is built

### Rules of Thumb for Requirement Modeling?

→ should have a high level of abstraction

→ each element of the model should add to an overall understanding of software requirements & provide insight into the information domain, function & behavior of system

→ minimize coupling

→ should provide value to all stake holders

→ keep the model as simple as possible

## Scenario-Based Models

→ show requirements from the POV of various actors

→ It considers use cases & user stored

→ can be the following ~~three~~ types

(i) use cases

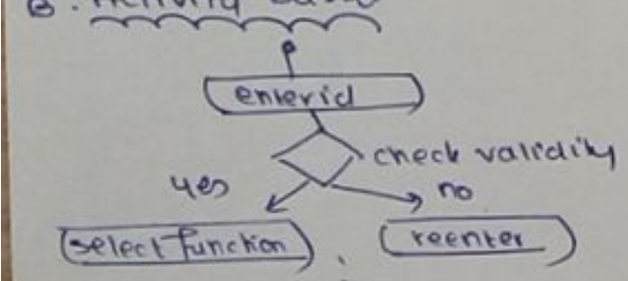
(ii) activity diagram

(iii) swimline diagrams

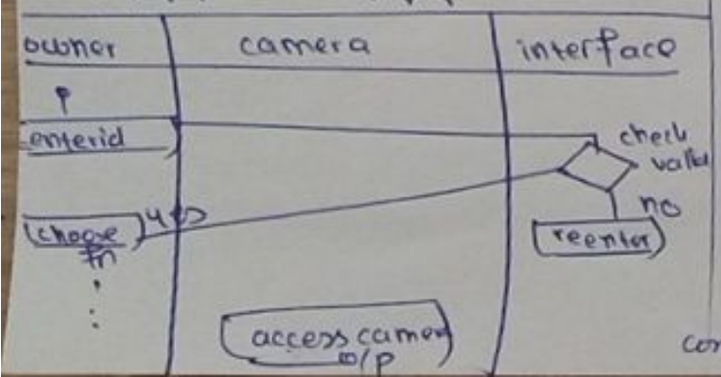
Example of a Safe Home System

A. Use case


B. Activity Based




C. Swimlane Diagram




## Data Models 2

### Class Oriented Models

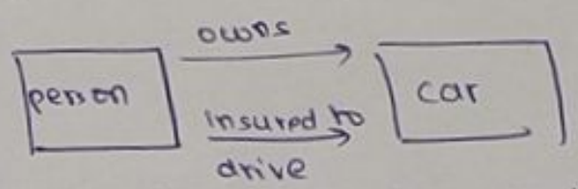
→ data models depict the information domain for the problem

→ class oriented models represent OOPs classes - attributes & operations

Datamodel - car ownership

data objects

| make | model | color | owner |
|---|---|---|---|
| | | | |



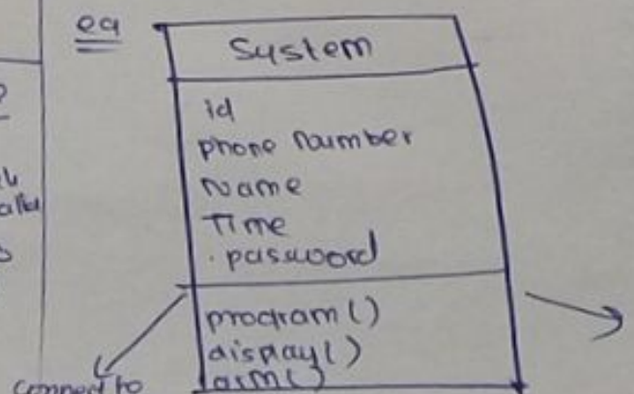

depict relationship between objs

Class Model

attributes - data objects that fully define the class

operations - define behavior of obj.

eg




## Flow Model

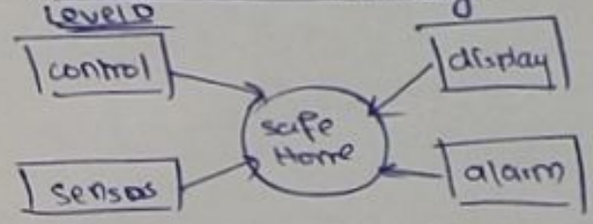
→ represents the functional elements of the system and how they transform data move through the system

→ has 3 levels

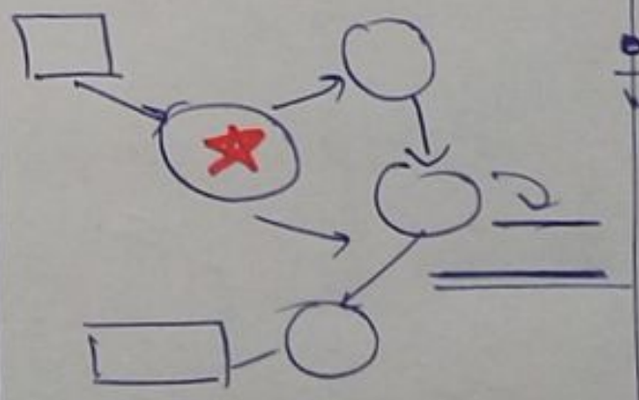
Level 0 - outline
Level 1 - shows relationships
Level 2 - expand on each functionality more

eg. Safe Home Security

Level 0




Level 1



Level 2 Expand upon each ★ showing I/O

## Behavioral Model (17)

→ indicates how the software responds to external events / stimuli

→ Follow the given steps:

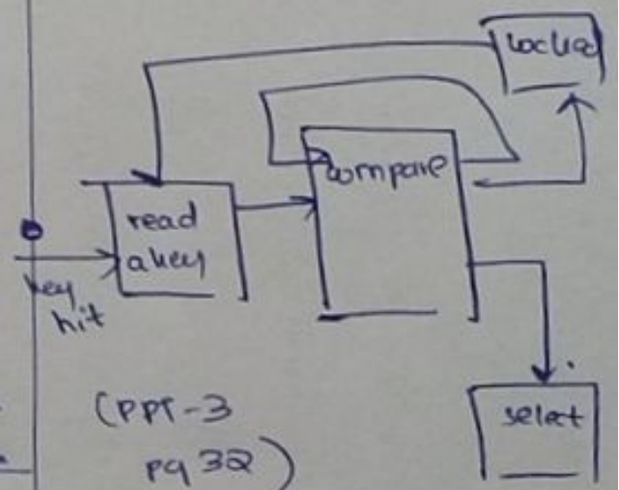
(i) Evaluate all use cases

(ii) Identify events

(iii) Create a sequence

(iv) Build a state diagram

eg. for the control panel of the Safe Home System




(PPT-3 pg 32)

## * Requirement Modelling Tools

→ Requirements can be modelled with:

(i) context diagram

(ii) Functional decomposition

(iii) use-case diagram

(iv) sequence diagram

(v) user stories

→ Some commercial tools are:

(i) StarUML

(ii) Microsoft Visio & Mind Genius

(iii) Visual Paradiam

(iv) OpenText Provision