

UNIT – II

Requirement Engineering:

**Requirement Engineering tasks,
Initiating the Requirements Engineering Process,
Eliciting Requirements**

Requirements Engg.

- Helps s/w engg's to better understand the pblm they will work to solve.
- Encompasses the set of tasks that lead to an understanding of
 - what the buss. impact of the s/w will be,
 - what the customer wants, and
 - how end-users will interact with the s/w.
- S/w engg's* (sometimes referred to as sys. engg's or analysts in the IT world) and *other prjt stakeholders* (managers, customers, end-users) all participate in requirements engg.

- It is imp. for designing and building an elegant comp. pgm that solves the wrong pblm which serves no one's needs.

A Bridge to Design and Construction

- Designing and building comp. s/w is challenging, creative, and just plain fun.

- RE, like all other s/w engg. activities, must be adapted to the needs of

- i)the process,

- ii)the prjt,

- iii)the product, and

- iv)the people doing the work.

- In perspective process models, RE is a s/w engg. action that begins during the communication activity and continues into the modeling activity.
- It is essential that the s/w team make a real effort to understand the requirements of a pblm before the team attempts to solve the pblm.
- RE builds a bridge to **design** and **construction**.

•The journey across the bridge takes us high above the prjt, allowing

1)s/w team to examine the context of the s/w work to be performed;

2)the specific needs that design and construction must address;

3)the priorities that guide the order in which work is to be completed; and

4)the info., functions, and behaviors that will have a profound **impact** on the **resultant design**.

Requirements Engineering Tasks

The RE process is accomplished through the execution of **7 distinct functions**:

- 1)inception,
- 2)elicitation,
- 3)elaboration,
- 4)negotiation,
- 5)specification,
- 6)validation, and
- 7)management.

1.Inception

•s/w engg's ask a set of context-free questions.

1.First set(focuses on cust. & other stakeholders) of context-free questions:

1)Who is behind the request for this work?

2)Who will use the solution?

3)What will be the economic benefit of a successful solution?

4)Is there another source for the solution that you need?

II. Next set of questions (enables the s/w team to gain a better understanding of the pblm):

- 1) How would you characterize "good" output that would be generated by a successful solution?**
- 2) What pblm (s) will this solution address?**
- 3) Can you show me (or describe) the buss environment in which the solution will be used?**
- 4) Will special performance issues or constraints affect the way the solution is approached?**

III.Final set of questions(focuses on the effectiveness of the communication activity itself):

**1)Are you the right person to answer these questions?
Are your answers "official"?**

2)Are my questions relevant to the problem that you have?

3)Am I asking too many questions?

4)Can anyone else provide additional information?

5)Should I be asking you anything else?

Intent:

- To establish a **basic understanding** of the **pblm**,
- The **people** who want a **solution**,
- The **nature of the solution** that is desired,
and
- The effectiveness of **preliminary communication and collaboration** b/n the customer and the developer.

2.Elicitation

It certainly seems simple enough

—ask the **customer**, the **users**, and others what the **objectives for the system or product** are,

-what is to be **accomplished**,

-how the **system or product is to be used** on a day-to-day basis.

- But it isn't simple—it's very hard.

-**Christel and Kang** identified a no. of pblms that help us understand why requirements elicitation is difficult:

i)Problems of scope:

- The **boundary** of the system is **ill-defined** or the **customers/users** specify **unnecessary technical detail** that may confuse.

ii)Problems of understanding:

- The **customers/users** are not completely **sure of what is needed**, have a poor understanding of the capabilities.

- don't have a **full understanding of the pblm** domain,

iii) Problems of volatility:

- The **requirements change** over time.
- To overcome these pblms, **REs must approach the requirements gathering activity** in an organized manner.

3.Elaboration

- The **info. obtained** from the **customer** during **inception and elicitation** is **expanded** and **refined** during “elaboration”.
- This requirements engg. activity **focuses on** developing a **refined technical model** of **s/w** **functions**, **features** and **constraints**.

4.Negotiation

- It is unusual for **customers and users** to ask for **more than can be achieved**, given **limited buss resources**.
- It is also relatively common for **diff. customers or users** to **propose conflicting requirements**, **arguing** that their version is "essential for our special needs."
- The requirements engg. must reconcile these conflicts through a process of negotiation.

- Rough "guesstimates" of development effort are made and used to assess the impact of each requirement on project cost and delivery time.

- Using an iterative approach, requirements are eliminated, combined, and/or modified so that each party achieves some measure of satisfaction.

5.Specification: In the context of s/w, the term specification means **diff. things to diff. people.**

- It can be

- 1)a written doc.,

- 2)a set of graphical models,

- 3)a formal mathematical model,

- 4)a collection of usage scenarios,

- 5)a prototype, or any combination of these.

- "standard template" should be developed and used for a specification.

- It describes the **function and performance** of a comp.-based sys. and the constraints that will govern its development.

6.Validation

- The work products produced as a consequence of RE are assessed for quality during a validation step.
- It examines the specification to ensure that all s/w requirements have been stated **unambiguously**, that **inconsistencies, omissions, and errors** have been detected and corrected

7.Requirements management

- It is a **set of activities** that help the **prjt team** **identify, control, and track requirements & changes to requirements** at any time as the project proceeds.
- It is similar to SCM.
- RM begins with identification.
- Each requirement is assigned a **unique identifier**.
- Once requirements are identified, traceability table(TT) are developed, each **TT relates requirements to one or more aspects of the sys. or its environment**.

The following are the diff. possible TTs:

- **Fig 7.1 generic traceability table(from TB)**

Software Requirements Specification Document (SRS)

Systems Requirements Specification

- SRS is the official statement of what the system developers should implement.
- SRS is a complete description of the behavior of the system to be developed.
- SRS should include both a definition of user requirements and a specification of the system requirements.
- The SRS fully describes what the software will do and how it will be expected to perform.

Systems Requirements Specification

Table of Contents:

- I. Introduction
- II. General Description
- III. Functional Requirements
- IV. Non Functional Requirements

1. Introduction
 - 1.1 Purpose
 - 1.2 Scope
 - 1.3 Definitions, acronyms, and abbreviations
 - 1.4 References
 - 1.5 Document overview
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
 - 3.1 Functional requirements
 - 3.1.1 Functional requirement 1
 - 3.1.1.1 Introduction
 - 3.1.1.2 Input
 - 3.1.1.3 Processing
 - 3.1.1.4 Output
 - 3.1.N Functional requirement M
 - 3.2 External interface requirements
 - 3.3 Performance requirements
 - 3.4 Design constraints
 - 3.5 Security requirements
 - 3.6 Maintainability requirements
 - 3.7 Reliability requirements
 - 3.8 Availability requirements
 - 3.9 Database requirements
 - 3.10 Documentation requirements
 - 3.11 Safety requirements
 - 3.12 Operational requirements
 - 3.13 Site adaptation

Systems Requirements Specification

1.INTRODUCTION

- **Purpose**
 - ✓ Describe the purpose of the SRS, not the purpose of the software being developed.
 - ✓ Intended audience for SRS.
- **Scope**
 - ✓ Describe application of software (benefits, objectives).
 - ✓ Explain what software will (not) do.

Example (video Rental System)

Purpose

The purpose of the Software Requirements Specification document is to clearly define the system under development, namely the Video Rental System (VRS). The intended audience of this document includes the owner of the video store, the clerks of the video store, and the end users of the VRS.

*Other intended audience includes the development team such as the requirements team, requirements analyst, design team, and other members of the developing organization.

Scope

The owner of a local video store wanted to create a new business plan where everything about renting a video (except the picking up and returning of videos) was done online.

Therefore, the new VRS will allow the following functionality online:

- to search for videos,
 - to become members,
 - to rent videos,
 - to modify membership information,
 - and to pay overdue fees.
- The store personnel may use the VRS to process the rented or returned videos, to add or remove videos to/from his store's video inventory and to update video information.
- The VRS is intended to increase the owner's profit margin by increasing video sales with this unique business approach and by allowing him to reduce the staffing needed in his stores.

- **Definitions/acronyms/abbreviations**
 - ✓ Definitions of terms and abbreviations that are used in the SRS.
E.g.
 - ✓ User: The person operating and/or using the software system.
- **References**
 - ✓ A complete list of all documents referenced elsewhere in the SRS.
 - ✓ Specify the sources from which the references can be obtained.
- **Overview**
 - ✓ Brief description of rest of SRS.
 - ✓ How the SRS is organized

2.OVERALL DESCRIPTION

- **Product Perspective**
 - ✓ If the product is independent and totally self-contained, it should be stated here.
 - ✓ Describe the functions of each component of the larger system or project, and identify interfaces.
- **Product Functions**
 - ✓ Provide a summary of the functions that the software will perform.
- **User Characteristics**
 - ✓ Describe those general characteristics of the eventual users of the product that will affect the specific requirements.

- **General Constraints**
 - ✓ Provide a general description of any other items that will limit the **developer's options for designing the system.**
 - ✓ In this section, the constraints of the system are listed. They include **hardware**, **network**, **system software**, and **software constraints**. It also includes **user constraints**, **processing constraints**, **timing constraints**, and **control limits**.

E.g.

1. The software system will run under Windows.
2. All code shall be written in Java.

- **Assumptions and Dependencies**

- ✓ **List and description of each of the factors that affect the requirements stated in the SRS.**

Example

A Product Perspective

- *The VRS is a web-based system.*
- *The system interfaces with two other systems, the owner's email system, the video distributor's video system, and the browsers used by VRS customers.*
- *The system provides a secure environment for all financial transactions and for the storing and retrieving of confidential member information.*

Example

Product Functions

The VRS allows customers to search the video inventory provided by this video store. To rent videos through the VRS, one must register as a member using the VRS. Upon becoming a member and logging into the VRS, the VRS provides the functionality for renting videos, modifying membership information, and paying overdue fines.

The clerks of the video store use VRS to process the return of rented videos. The owner of the video store uses VRS to add new videos into the system, remove videos from the system, and modify video information.

The VRS sends emails to members concerning video rentals. One day before a rented video is due to be returned, VRS emails the member a reminder of the due date for the video(s). For any overdue videos, VRS emails the member every 3rd day with overdue notices. At the 60-day limit for outstanding videos, VRS debits the member's credit card with the appropriate charge and notifies the member of this charge.

Systems Requirements Specification

User Characteristics

The three main groups of VRS users are customers, members, and store personnel. A customer is anyone who is not a member. The customer can only search through the video inventory. The amount of product training needed for a customer is none since the level of technical expertise and educational background is unknown. The only skill needed by a customer is the ability to browse a website.

Member is someone who has registered with VRS. A member can rent videos and pay fees online. As with a customer, these activities require no product training since the level of technical expertise and educational background of a member is unknown. The only skill needed by a member is the ability to browse a website.

The store personnel are divided into two groups: the clerk-level personnel and owner-level personnel. Their educational level is unknown and both group needs little to no training.

Systems Requirements Specification

General Constraints

This system provides web access for all customer and member functions. The user interface will be intuitive enough so that no training is required by customers, members, or store personnel. All online financial transactions and the storage of confidential member information will be done in a secure environment. Persistent storage for membership, rental, and video inventory information will be maintained.

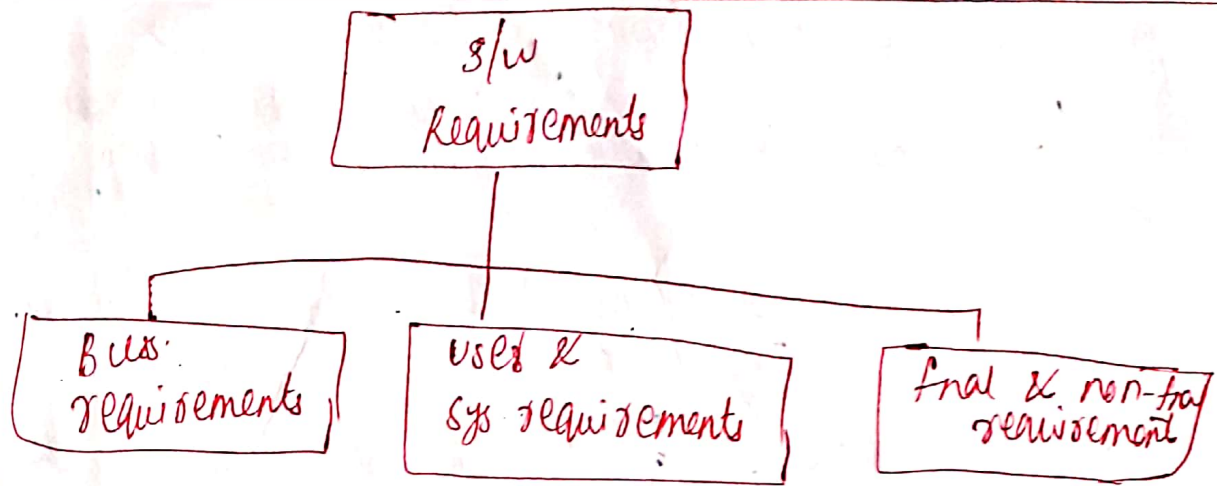


fig: Types of requirements

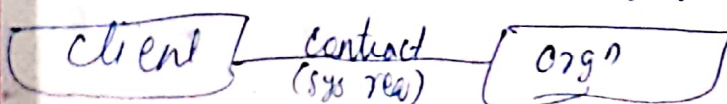
Bus. requirements:-

- profit goal & expected bus. benefits
- bus. needs
- prod. domain & prod. demand.

— Bus. anal.

User Requirements & sys. Requirements:-

- supplied by the customer, end-users, or other stakeholders
- Sys Req — functionalities that the sys. is expected to provide within certain constraints / environment.
- user req $\begin{cases} \text{final} \\ \text{non-final} \end{cases}$
 - h/w & s/w
- sys. Req — technical functionalities written in a systematic manner to achieve user req.



(P.T.O.)

③ Functional & Non-functional Req.:-

↳ to complete the product domain
↳ features which differentiate the sys. from other's products

functional — behavior / frs that the sys. must support.

Eg: services
tasks
behaviors.

non-functional — specify how a sys. must behave

Eg: Qualities
std's
constraints

Requirements Anal.:-

- * Analyzing stakeholder's needs,
 - constraints
 - assumptions
 - qualifications.

Adv: Refines & defines req. in a clear and unambiguous manner.

Types:

1. Structured analysis
2. Data-oriented
3. obj-oriented
4. prototyping

[Contd...]

Structured Analysis:- (process modeling / data flow modeling)

1. DFD - sys. behavior
2. Data Dictionary - Composite data structures defined in DFD.

→ symbols :

- + → composition
- * → repetition
- | → selection
- [] → combination of repeated data

Eg: restaurant bill

- restaurant_bill = dish_price + sales_tax + service_charge + seat_charge + grand_tot.
- dish_price = [dish_name + quantity + price] *
- seat_charge = [reserved_table | common_table]

- DFD + DD's
3. structured Anal. method - techniques and graphical tools to represent the scenario of the working sys. in an orgn

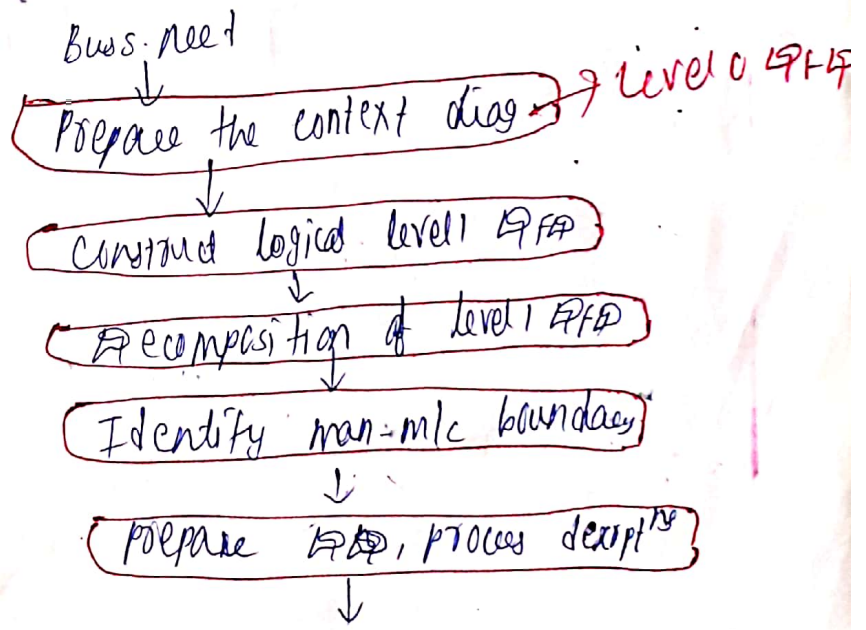


Fig: structured anal. approach

Requirements specification

(P-T-C)

② Data-oriented Anal.:-

1. E-R modeling (ERM)

Entities attributes R-s's

Aggregatⁿ → part-of / part-whole R-s's b/w entities

Specialization → is-a R-skip.

Generalization →

2. Data-oriented Anal. method.

- Identification of entity & R-s's
- Construct the basic E-R diag
- Add key attributes
- Add non-key "
- Apply hierarchical relation
- Perform normalization
- Add integrity rules.

③ Obj-oriented Anal.:-

OOA - Engg. Req, modeling the app^{re}

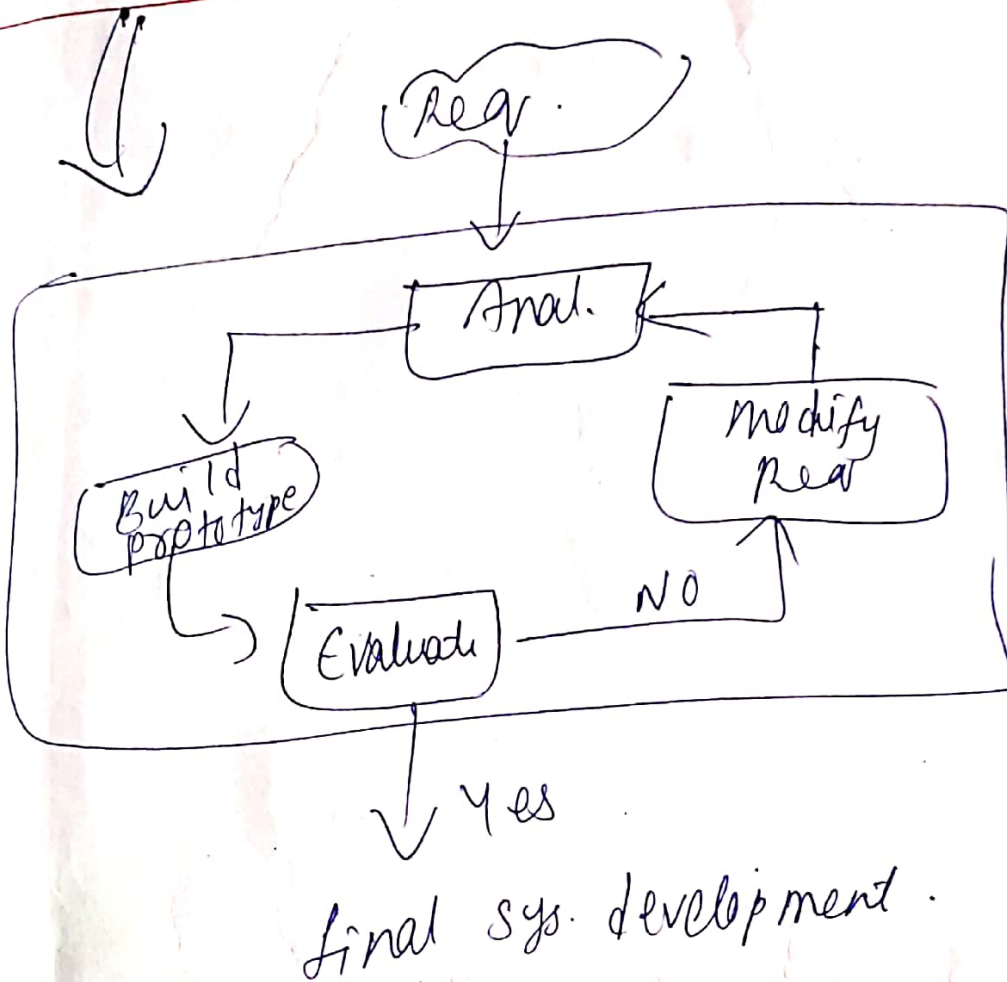
OOD - translates OOA into pgm^{ing}. Constructs to produce practical

- class diag
- obj modeling :
- dynamic modeling : seq. diag
- functional modeling : st-chart diag

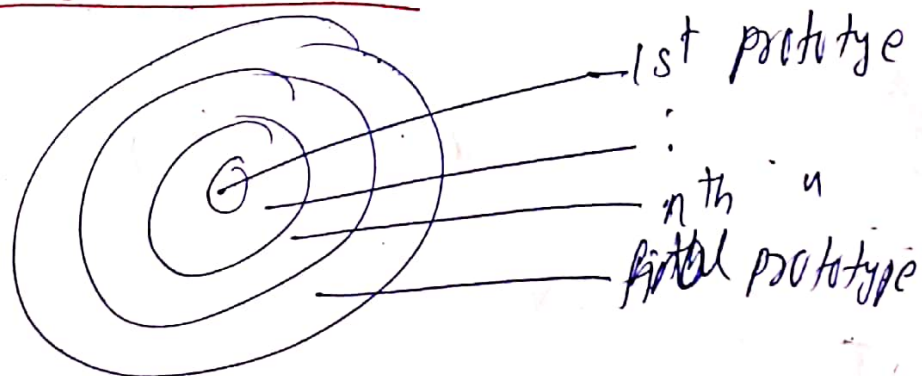
Contd--

④ Prototyping: Throwaway prototyping
Evolutionary

① Throwaway Prototyping:



② Evolutionary prototyping: Iterative



→ use RAA

SRS specific methods.

- ① A. Tree
- ② A. Table

①

③ PAL - specifies procedures of functional requirements in slw.

- Derived from Ada
- Pseudo code.

Eg: SPS

Procedure SPS is

BEGIN

LOOP

read student_result(credits, sgpa)

IF credits is greater than 52

THEN calculate sgpa

ELSE

msg incomplete credit

ENDIF

IF sgpa is greater than 4.00

THEN promote()

ELSE readmit()

ENDIF

END LOOP

④ Regular Expr - describes syntactical structure of a set of strings.

P.T.O

⑤ Graphical methods — PFD, ER, FDM (represents behavior of a sys.)

⑥ Mathematical methods — used to model complex sys's in mathematical technol^y.

- algebraic specificaⁿ
- axiomatic^y
- theorems.

Specificaⁿ = (Sort, Operⁿ, Axioms);

set of values
that describes
data type or class

name of
the operⁿ,
its parameters
and return type

rules that must
hold true in any
legal implementaⁿ
of sort.

Requirements validaⁿ:- To avoid ~~an~~ inconsistency,
incorrect fact, ambiguity, omission of unnecessary
requirements

→ ① Requirements Review:-

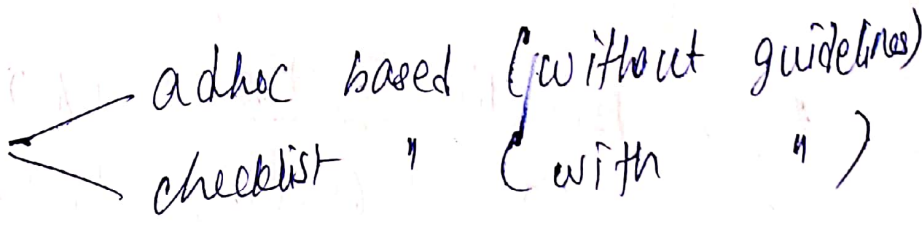
- ↳ done by group of people
- ↳ involves readers from both sides
(clients & developer)

- Review plan
- " team
- Time & place

② Requirements Inspection:- detects defects at early stages. (up to 90%)

- moderator keeps doc's ready
- inspectors inspect
- organiser holds meeting

③ T-c's Generation:- Black box test cases are created.

④ Reading: 

- Ambiguity
- understandability
- completeness
- Redundancy
- Testability
- modifiability → Traceability

⑧ Prototyping:- understand, analyse, and validate req's.

Prototyping validation — identifies errors like missing, incomplete and incorrect req's from req. doc.

Req's mgmt:- Process of systematically collecting, organising, documenting, prioritizing, and negotiating on the req's for a projt.