Topic : Requirement Fund... Unit...

3.1 Requirement Fundamentals :-

Understanding the requirements of a problem is among the most difficult tasks that face a s/w engineer.

Initially req. engg. doesn't seems that hard.

But customer doesn't know what is required?

Even an end-users have a good understanding of the features & functions that will provide benefit.

So, finally req. engg. is hard.

Ref. :- R1, P.No → 174

3.2 What is Software Requirements :-

Requirements engg. helps s/w engineers to better understand the problem they will work to solve. It encompasses the set of tasks that lead to an understanding of what the business impact of the s/w will be, what the customer wants, & how end-users will interact with the s/w.

S/w engineers & other Project

stakeholders (managers, customers, end-users) all participate in requirements engg.

Ref :- R1 : P.No. → 174

3.3 Characteristics of Requirements :-

To ensure that both developers & customers understand the requirements properly, it is important that the requirements be of high quality. To this end, it is to be checked that the requirements should have the following characteristics.

1) **Correct :-** Each requirements must accurately describe the functionality to be delivered. A s/w requirement that conflicts with a corresponding system requirements is not correct. only the inclusion of user can make the requirements correct.

2) **Feasible :-** It must be possible to implement each requirement within the known capabilities &

Name of Lecturer : .....

limitations of the system & its environment. The developer can provide a reality check on what can & cannot be done technically.

3) Necessary :- Each requirement should be documented properly so that the necessary requirements should not be missed or ommitted during implementation

4) Complete :- No requirements or necessary information should be missing. Completeness is also desired characteristic of an individual requirement (self explanatory).

5) Verifiable :- check whether you can use tests & verification approaches such as inspection or demonstration, to determine whether each requirement is properly implemented in the product. If a requirement is not verifiable, determining whether it was correctly implemented is a matter of opinion.

6) Traceable :- you should be able to link each s/w req's. to its source i.e. higher level system req. or with customer statement

Also link each s/w requirements to the design elements, source code, & test cases. All traceable requirements are

Name of Lecturer : Abhishek Jain.

→ req has a unique no. or identify
→ cannot be separated or broken into smaller req. after giving unique no.
→ req. are uniquely identified whether it is in implementation or testing phase

uniquely numbered or labeled & are written in a structured fine-grained way, as opposed to large, narrative paragraphs or bullet lists.

7) Modifiable :- The req's. must be flexible enough to make changes or modifications in further or futer development.

8) Consistent :- Requirements do not conflict with other requirements in the requirement specification. Uses the same terminology throughout the requirement specification. does not duplicate/redundance

Ref :- R4

3.4 Types of Requirements :-

There are various types of req. for developing a Sw. are described as follows:

Name of Lecturer : .....

1) Business Requirement

2) User Interface Requirement

3) Users & Human Factors

4) Functional Requirements (FR)

5) Non-Functional Requirements (NFR)
   i) Product Requirements
   ii) organizational Requirements
   iii) External Requirements.

6) Resources Requirements

7) Documentation Requirement.

1) Business Requirement (BR):-

→ These are high-level business goals of the organization building the Product, or the customer who sponsored the Project.

→ These are usually provided as a single Page of high-level bullets.

2) User Interface Requirements :-

→ Interface refers the medium by which user can interact with the s/w. or system.

It also Point;

→ is the i/p comming from one or more systems

→ is the o/p going to one or more systems.

3) User & Human Factor :-

Who will use the system ?

will there be a special target ordians) or a layman also ?

What is the skill-level of each type of user ?

What kind of training will be required for each type of user.

4) Functional Requirements :-

These are statements of services the system should provide, how the

Name of Lecturer : .....

System should behave in particular situations. In some cases, the functional requirements may also explicitely state what the system should not to do.

5) Non-Functional Requirements :-

These are constraints on the services or functions offered by the system. They include timing constraints, constraints on the development Process, & constraints imposed by standards. Non functional requirements often apply to the system as a whole, rather than individual system features or services.

i) Product Requirements :-

The requirements specify or constraint the behavior of the S/w. Examples include Performance requirement on how fast the system must execute & how much memory it requires, reliability requirements that set out the acceptable failure rate, security requirements, & usability requirements.

ii) Organizational Requirements :-

These requirements are broad system req.'s.

derived from policies & procedures in the customer's & developer's orgzn. Examples include operational process req's. that define how the system will be used, development process req's. that specify the programming language, the development environment or process standards to be used & operational requirements that speify the O.S. environment

iii) External Requirements :-

This broad heading covers all req's. that are derived from factors external to the system and its development process. These may include regulatory req's. that set out what must be done for the system to be approved for use by a regulator, legislative req's. that must be followed to ensure that the system operates with in the law, & ethical requirements that ensures that the system will be acceptable to its users & the general public.

6) Resources Requirements :-

It includes H/w, S/w, skilled developers

Name of Lecturer : .....

proper labs & cost or amount of money required.

7) Documentation Requirement :-

Proper documentation is required to develop a good system. The document may be in a hard or soft format or may be both. The Document must be self explainatory & addressed to all audience (developer, user or customer).

Ref :- R2; P.No. → 84-89, R5

3.5 Differentiating the User Requirements & System Requirements :-

User Requirements	Client Managers System End-users Client Engineers Contractor Managers System Architects
System Requirements	System End-Users Client Engineers System Architects Software Developers

→ User Requirements :-

→ It referred to as user needs, describes what user wants from system, such as what activities that users must be able to perform.

→ User requirements are generally documented in a User Requirement Document (URD) using narrative (explained) text.

→ User requirements are generally signed off by the user & used as the primary input for creating system requirements.

→ Most of the time user are not able to communicate his req. properly (incomplete or conflicting).

→ System Requirements :-

→ It is a building block used by developers to build the system.

→ These are traditional "shall" statements that describes what the system "shall do".

Name of Lecturer : .....

→ System requirements are classified as either functional & non-functional req's.

→ A functional requirements specifies something that a user needs to perform their work. For example; a system may be required to enter & print cost estimates.

→ Non-functional req's. specify all the remaining req's. not covered by the functional req's.

→ Non-functional req's. are sometimes called quality of service req's.

→ System req's. also specify the necessary functionality to ensure that these services/features are delivered properly.

Ref. :- R2; P.No. → 85, R6

3.6 Why are Requirements Important :-

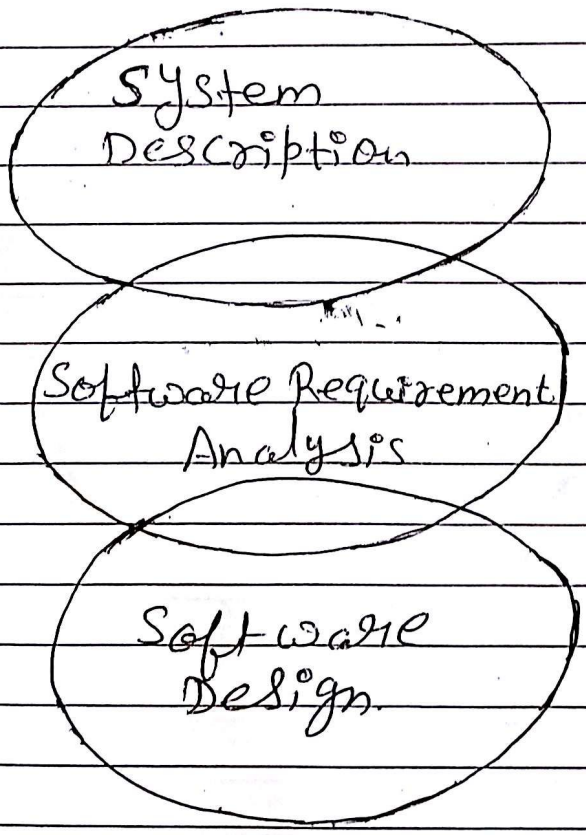
Designing & building an elegant computer program that solves the wrong problem serves no one's needs. That's why it is important to understand what the customer

wants before you ..... begin to design & build a computer based system. That's why the requirements are so important to understand the consumer or customer's need.

Ref :- R1 ; P.No. → 174

3.7 Software Requirement Analysis :-

S/w Requirement Analysis bridges a gap b/w system description & s/w design.



Analysis Model as a bridge between Software Description & software Design.

Name of Lecturer : .....

Requirement analysis results in the specification of s/w's operational characteristics; indicates s/w's interface with other system elements; & establishes constraints that s/w must meet. Req's analysis allows the s/w engineer to elaborate on basic requirements established during earlier req. engg. tasks & build models that depict user scenarios, functional activities, Problem classes & their relationships, system & class behavior & the flow of data & it is transformed.

Throughout analysis modeling, the s/w engineer's primary focus is on what, not how;

→ What objects does the system manipulate?

→ What functions must the system perform?

→ What behaviors does the system exhibit?

→ What interface are defined?

→ What constraints apply?

3.8 Need Of Requirement Analysis :-

The requirement analysis is one of the important phase of the s/w development life cycle. There are some needs of this phase which are described below;

1) Requirement analysis enable the system engineer:

→ to specify s/w function & Performance

→ to indicate s/w's interface with other system elements.

→ to establish design constraints that the s/w must meet.

2) It allows the s/w engineer to refine the s/w allocation & build models of the Process, data & behavioral domains that will be treated by s/w.

3) It describes the s/w's application, architecture, user interface & component level structure.

4) Finally the req's. specification provides the developer & the customer the means

Name of Lecturer : .....

to assess quality, once the S/w is built.

Ref:- R1; P.No.→208-209

3.9 Requirement Analysis Tasks :-

The req's. analysis process is accomplished through the execution of seven distinct tasks.

1) Inception.
2) Elicitation.
3) Elaboration.
4) Negotiation.
5) Specification.
6) Validation.
7) Management.

1) **Inception :-** (begining of any act).

At Project inception, S/w engineers ask a set of content-free questions (Stage wise questions)

The intent is to establish a basic understand -g of the problem, the people who want a Solution & the effectiveness of priliminary communication & collaboration b/w the customer & the developer.

2) Elicitation:-

→ In this exact req's are gathered from users, customers & others. (Stakeholders)

→ The questions ask by engineers are

→ What is to be accomplished?

→ What are objectives for the system or product are?

→ How the system or product fits into the needs of the business.

→ Finally, How the system or product is to be used on a day-to-day basis.

→ It isn't a simple, its very hard.

3) Elaboration:-

→ The info. obtained from the customer during inception & elicitation is expanded & refined during elaboration.

→ Elaboration focuses on developing a refined technical model of S/w functions,

Name of Lecturer: .....

features & constraints.

→ Elaboration describes how the end-user will interact with the system.

4) Negotiation:-

→ It is very common for different customer or users to propose conflicting requirements.

→ The requirements engineer must reconcile these conflicts through a process of negotiation

→ Customers, users, & other stakeholders are asked to ranked their requirements & then discuss conflicts in priority.

→ Risk associated with each req's. are identified & analyzed.

5) Specification:-

→ The specification is the final work product (documented) produced by the requirements engineer.

→ It serves as the foundation for subsequent SW engg. activities

→ It describes the function & performance of a s/w system & the constraints that will govern its development.

6) Validation :-

→ Requirement validation examines the specification to ensure that all s/w req's. have been stated unambiguously; that inconsistencies, omissions & errors have been detected & corrected; and that the work products conform to the standards established for the process, the project & the product.

→ Validation criteria are specified to demonstrate understanding of a successful s/w implementation.

7) Management :-

Req's. mgmt. is a set of activities that help the project team identify, control & track requirements & changes to req's. at any time as the project proceeds.

Ref. :- R.J; P. No. → 176-180

Name of Lecturer : .....

3.10 **Requirement Elicitation :-**

As discussed earlier in 3.9 ; Point 2 ,

Now further explaination as follows.

Requirement elicitation also sometime referred as "requirement gathering".

Commonly used elicitation processes are the stakeholder meetings or reviews. For example, an important first meeting could be b/w s/w engineers & customers where they discuss their perspective of the req's.

Problems :-

1) **Problems of scope :-**

The boundary of the system is ill-defined or the customers/users specify unnecessary technical detail that may confuse, rather than clarify, overall system objectives.

2) **Problems of Understanding :-**

The customers/users are not completely sure what is needed, have a poor understanding of the computing environment, don't have a

full understanding of the problem domain, have trouble communicating needs to the system engineer, omit info. that is believed to be "obvious", specify req's. that conflict with the needs of other customers/users, or specify req's. that are ambiguous or untestable.

3) Problems of Volatility :-

The req's change over time. The rate of change is sometimes referred to as the level of req. volatility.

Requirements quality can be improved through these approaches;

1) Visualization :- Using tool that promote better understanding of the desired end-product such as visualization & simulation.

2) Consistent Language :-

Using simple, consistent definitions for req's. described in natural language & use the business terminology i.e. prevalent in the enterprise.

Name of Lecturer : .....

3) Guidelines :- Following organizational guidelin-s that describe the collection techniques & the types of req's. to be collected. Thes: guidelines are then used consistently across Projects.

4) Consistent Use of Templates :-

Producing a consistent set of models & templates to document the req's.

5) Documenting Dependencies :-

Documenting dependencies & interrelationships among req's.

6) Analysis of Changes :-

Performing root cause analysis of changes to req's. & making corrective actions.

Ref. :- R1; P.No. :- 177, R7

3.11 Requirement Analysis Principles :-

Davis suggests a set of 6 guiding Princip-s for req. analysis;

1) Understand the Problem before you begin to create the analysis model :-

There is a tendency to rush to a solution, even before the problem is understood. This often leads to elegant s/w that solves the wrong Problem.

2) Develop Prototypes that Enable a user to understand how Human/Machine Interaction will occur :-

Since the of the quality of s/w is often based on the Perception of the "friendliness" of the interface, Prototyping (the interaction that results) are highly recommended.

3) Record the origin of and the reason for every requirement :-

This is the first step in establishing traceability back to the customer.

Name of Lecturer : .....

4) Use Multiple views of Requirements :-

Building data, functional & behavioral models provide the s/w engineer with three different views. This reduces the likelihood that something will be missed & increases the likelihood that inconsistency will be recognized.

5) Rank Requirements :-

Tight deadlines make it impossible of every s/w req. If an incremental process model is applied, those req's. to be delivered in the first increment must be identified.

6) Work to Eliminate Ambiguity :-

Because most req's. are described in a natural language, the opportunity for ambiguity abounds. The use of formal technical reviews is one way to uncover & eliminate ambiguity.

A s/w engineer who takes these principles to heart is more likely to develop a s/w specification that will provide an excellent foundation for design.

Ref :- R8

3.12 Software Prototyping:-

Definition:-

"The S/w Prototype refers to building S/w application prototypes which display the functionality of the Product under development but may not actually hold the enact logic of the original S/w".

Ref:- R9

→ Prototyping is composed of collecting req's & using them to build a prototype.

→ It enables users to see a potential solution & get a better feel for what they require.

→ Users then add or amend their req's, with the prototype.

→ The prototype being reworked on an iterative basis, until req's are feered up.

Name of Lecturer : .....

Advantages:-

→ Good for exploring how a particular s/w function req. could work in a s/w development.

→ It can identify Problems with req's. & can improve the quality of req's. and hence the ultimate solution.

Disadvantages:-

→ It is less useful for the initial req's identification.

→ Prototyping is not really suitable for large applications.

→ It can be expensive technique for identifying req's.

Ref. :- R1o

3.13 Software Requirement Specification (SRS):-

Definition:- "SRS is a vital piece of documentation that is crucial to the success of any s/w development Project".

→ An SRS is basically an organization's understanding (in writing) of a customer or clients requirements.

→ It is a two-way insurance policy that assures that both the client & the organization understand each other's req's.

→ The SRS document itself states in precise & explicit language those functions & capabilities of a s/w sys. must provide.

→ The SRS also functions as a blueprint for completing a project with as little cost growth as possible.

→ The SRS is often referred to as the "Parent" document because all subsequent project mgmt. documents, such as design specifications, statements of work, s/w architecture specifications, testing & validation plans, & documentation plans, are related to it.

→ It's important to note that an SRS contains functional & non-functional req's. only.

Ref:- R11.

Name of Lecturer : .....

[3.14] SRS Fundamentals :-

A well-designed, well-written SRS accomplishes four major goals;

1) It Provides Feedback to the Customer :-

→ An SRS is the customer's assurance that the development organization understands the issues or problems to be solved & the s/w behavior necessary to address those Problems.

→ Therefore, the SRS should be written in natural language in an unambiguous manner that may also include charts, tables, data flow diagrams, decision tables, and so on.

2) It Decomposes the Problem into Component Parts :-

The simple act of writing down s/w req's. in a well-designed format organizes formation Places borders around the problem & helps breakdown the Problem into its component Parts in an orderly fashion.

3) It Serves as an Input to the Design specification :-

→ The SRS serves as the Parent document to subsequent documents, such as the S/W design Specification & Statement of work.

→ The SRS must contain sufficient detail in the functional system req's. so that a design solution can be devised.

4) It Serves as the Parent Document:-

→ S/W req. specification are typically developed during the first stage of "Requirements Development", which is the initial product development phase in which info. is gathered about what req's. are needed - and not.

→ The info. gathering stage include, onsite visits, surveys, interviews & needs analysis of the customer or client's current business environment.

→ Parent document serves for testing & validation strategies that will be applied to the req's of verification.

Ref:- R11

3.15 **SRS Principles:-** SRS may be viewed as a representation process. Req's. are represented in a manner that ultimately leads to successful S/w implementation. Balzer & Goldman Propose eight Principles of good Specification:

1) Separate functionality from implementation.

2) Develop a model of the desired behavior of a system that encompasses data & the functional responses of a system to various stimuli from the environment.

3) Establish the content in which S/w operates by specifying the manner in which other system components interact with S/w.

4) Define the environment in which the system operates & indicate how a highly intertwined collection of agents react to stimuli in the environment (changes to objects) Produced by those agents.

5) A specification must be operational.

6) The System specification must be tolerant of incompleteness & augmentable (increasing). A specification is always a model-an

Name of Lecturer : .....

abstraction of some real situation i.e. normally quite complex. Hence, it will be incomplete & will exist at many levels of details.

7) A specification must be localized & loosely coupled so establish the content & structure of a specification in a way that will enable it to be possible to change.

8) A system specification must be req. of the user.

Ref :- R3

3.16 Characteristics of SRS :-

A good SRS document, should have the following characteristics:

1) **Complete :-** SRS precisely defines (requirement) everything that S/w is supposed to do. SRS also defines system's capability of S/w

2) **Consistent :-** Req's. at all levels must be consistent with each other.

Any conflict b/w req's within the SRS must be identified & resolved.

- Format, details of real-world entity interfacing with the system may be conflicting for ex:- one req. states that an individual can work up to 6 hrs, whereas another req. state it as 8 hrs.

- The terminology used for some entities/ events may be different,
for ex:- diff req's may use different terms to refer to the same object.

3) Accurate:- SRS Precisely defines the system's capability in a real-world enviroment, as well as how it interfaces & interacts with it. This aspect of req's. is a significant problem area for many SRS's.

4) Modifiable:- The SRS should be flexible enough to facilitate any necessary modifications. certain good practices that can lead to high modifiability are- minimal redundancy & the numbering (labeling) of the req's.

5) Ranked:- Generally, the req's are stated acc. to their priorities.

→ All req's are important, out of which some are critical & some or not.

→ Some req's are are 'core' req's which are not likely to change as time passes, while others are more dependent on time.

→ Each req. are ranked acc. to their importance & stability.

6) Testable :- An SRS must be stated in such a manner that each & every req. must be testable after implementation.

7) Traceable :- An SRS is traceable if the origin or source of each of its req's is clear & if it facilitates the referencing of each req. in future development.

8) Valid :- A valid SRS is one in which all parties & Project Participants can understand, analyze, accept, or approve it. This is one of the main reasons SRSs are written using natural language.

Name of Lecturer : .....

9) **Unambiguous :-** SRS must contain req's. Statements that can be interpreted in one way only (with no confusion). This is another area that creates significant Problems for SRS development because of the use of natural language.

10) **Verifiable :-** A SRS is verifiable if & only if every stated req. is verifiable (testable). Unambiguity is essential for verifiability. As verification of req's. is often done through reviews, it also implies that SRS is understandable, at least by the developer, the client & the users.

Ref :- R13

3.17 Components of SRS :-

In the previous section, we discussed various characteristics that will help in completely specifying the req's. Here we describe some of System Properties that an SRS should specify. The basic issues, an SRS must address are:

1) Functional Requirements.
2) Performance Requirements.
3) Design Constraints.
4) External Interface Requirements.

1) Functional Requirements :-

Functional req's specify what o/p should be produced from the given i/p's. So they basically describes the connectivity blw the i/p & o/p of the sys. For each functional req :

→ A detailed description of all the data i/p's & their sources & the range of valid i/p's be specified.

→ All the operations to be performed on the i/p data obtain the o/p should be specified.

→ Care must be taken not to specify any algorithms that are not parts of the system but that may be needed to implement the system.

2) Performance Requirements (Speed Req's) :-

→ This part of an SRS specifies the Performance Constraints on the s/w sys.

→ All the req's related to the Performance Characteristics of the system must be clearly specified.

Name of Lecturer : .....

→ Performance req's. are typically expressed as processed transactions Per second or response time from the system for a user event or screen refresh time or a combination of these.

→ It is a good idea to Pin down performance req's. for the most used or critical transactions, user events & screens.

3) Design Constraints :-

The client environment may restrict the designer to include some design constraints that must be followed. The various design constraints are:

→ Standard Compliance :- It specifies the req's. for the standard, the system must follow. The standards may include the report format & according procedures.

→ Hardware Limitations :- The S/w needs some existing or Predetermined H/w to operate, thus imposing restrictions on the design. H/w limitations can includes the types of machines to be used, O.S. availability, memory space etc.

→ Fault Tolerance :- Fault tolerance can place a major constraint on how the system is to be designed. Fault tolerance req's. often make the system more complex & expensive, so they should be minimized.

→ Security :- Security req's have become essential & major for all types of systems. Security req's. place restrictions on the use of certain commands, control access to database, Provide different kinds of access req's. for different people, require the use of passwords & cryptography techniques, & maintain a log of activities in the system.

4) External Interface Requirements :-

For each external interface req's.

→ All the possible interactions of the s/w with People, H/w & other s/w should be clearly specified.

→ The characteristics of each user interface of the s/w product should be specified.

Name of Lecturer : .....

→ The SRS should specify the logical characteristics of each interface b/w the s/w product & the H/w components for H/w interfacing.

Ref:- R12

3.18 Contents of SRS:-

The SRS is one of the Contract deliverable Data Item Descriptions, or have other forms of organizationally-mandated content.
The SRS content are as follows:

1) Introduction:-

1.1 Purpose of the Requirement Document.
1.2 Scope of the Product.
1.3 Definition, Acronyms or Abbreviations.
1.4 Intended Audience
1.5 References
1.6 System overview
1.7 Contact information/SRS Team members.

2) Overall Description:-

2.1 Product Perspective
2.1.1 System Interfaces
2.1.2 User Interfaces
2.1.3 Hardware Interfaces
2.1.4 Software Interfaces

2.1.5 Communication Interfaces
2.1.6 Memory constraints
2.1.7 Operations
2.1.8 Site Adaptation Requirements.

2.2 Product Functions
2.3 User Characteristics
2.4 Constraints, assumptions & dependencies.

3) Specific Requirements :-
3.1 External Interface Requirements
3.2 Functional Requirements
3.3 Performance Requirements
3.4 Design constraints
    3.4.1 Standard Compliance
3.5 Logical Database Requirement
3.6 Software System Attributes
    3.6.1 Reliability
    3.6.2 Availability
    3.6.3 Security
    3.6.4 Maintainability
    3.6.5 Portability.

4) Other Requirements :-

4.1 Appendix A : Terminology / Glossary / Definition List
4.2 Appendix B : To be determided.

Ref :- R11, R13

Name of Lecturer : .....

3.19 Common Problems with SRS :-

When writing a SRS there are some Problems occurs in it. Here some of the common Problems with SRS:

1) **Omission of Requirements :-**

It is a common error in requirements. In this type of error, some user req's. are simply not included in the SRS. The omitted req. may be related to the behavior of the system, its Performance, Constraints or any other factor.

2) **Inconsistency :-** It can be due to contradictions within the req's. themselves or to incompatibility of the stated req's. with the actual req's. of the client or with the environment in which the system will operate.

3) **Incorrect Fact :-** This error occurs when some fact recorded in the SRS is not correct.

4) **Ambiguity :-** This error occurs when there are some req's. that have multiple meanings, i.e., there interpretation is not unique

5) Writing Implementation (HOW) Instead of Requirements (WHAT) :-

Req. Specifications should state WHAT is needed, not HOW to be provided.

Ref. :- R14

3.20 SRS Review :-

Req's. validation determines whether the req's correct enough to design the system. The Problems encountered during req's. validation are listed below;

1) Unclear stated req's.

2) Conflicting req's. are not detected during Req. Analysis

3) Errors in the req's. elicitation & Analysis.

4) Lack of conformance to quality Standards.

→ To avoid the Problem stated above, a req's. review is conducted.

Name of Lecturer : .....

→ For this purpose a review team is created.

→ The review team consist of s/w engineers, users & other stakeholders.

→ Review team checks all the problem encountered & whether the work products produced during the req's. phase are following specified standards or not.

→ A 'good' review checklist consist of the following :

1) Is the initial state of the system defined?

2) Is there a conflicts b/w one requirement & the other?

3) Are all req's specified at the appropriate level of Abstraction.

4) Is the req. necessary or does it represent an add-on feature that may not be essentially implemented?

5) Is the req. bounded & has a clear defined meaning?

6) Is each req is feasible to implement to technical environment.?

7) IS testing possible once the req. is implemented.?

8) Are the req's. consistent with the overall objective specified for the system/Product?

9) Have all H/w resources been defined?

10) Is the provision for possible future modifications specified?

11) Are functions included as desired by the user.

12) Can the req's. be implemented in the available budget & technology?

13) Are the resources of req's. or any system model (created) stated clearly?

Rod:- R15

3.21 Data Dictionary :-

Definition :-

"The data dictionary is an organized listing of all data elements that are used to build a system, with precise rigorous definitions so that both user & system analyst (Programmer) will have a common understanding of i/p's, o/p's, Processing & Storation."

Ref. :- R3

The format of data dictionary varies from tool to tool, the most common attributes are,

1) Name :- The primary name of the data or control item, the data store or an external entity.

2) Alias :- A temporary name use for the first entry (Name).

3) Where-used/how-used :- A listing of the processes that use the data or control item & how it is used, (eg. I/p to the process, o/p from the process, as a store).

4) Content Description :- A standard format for representing a content.

5) Supplymentary Information :-

Other information about data types. Preset values (if known), restrictions or limitations etc. (ex. only CAPS)

Name	Alias	Use	Content Description	Supplement / Additional I/P
Telephone Number	TN	• Phone set up • Dial phone	TN = [local No. / STD No.] local No. = Pbx/int access No. STD No. = Int access No.	use only + sign (phone) & Numbers 0 to 9

Example of Data Dictionary Table

Ref :- R 16

3.22 Advantages & Disadvantages of Data Dictionary :-

→ Advantages :-

1) It is a valuable reference in any organization because it provides documentation.

2) It improves a communication blw system analyst (or programmer) & user by establishing consistent definitions of various items & procedures.

3) It is a good tool manage operators & other members of the development team to understand req's. & design.

4) It helps the analyst to simplify the structure for meeting the data req's. of the system.

5) It is just like a store of all data elements information that can link all Phases of SDLC.

6) It is used to remove redundancy in data definition.

7) It is an important ... step building a database. Most database management system has a data dictionary as a standard feature.

8) During implementation, it serves as a base against which developers compare their data descriptions.

→ Disadvantages :-

1) It does not provide functional details.

2) It is not acceptable to many non-technical users.

Ref :- R17

3.23 Finite-State Machine Models :-

Definition :- "A Finite-State Machine (FSM) or Finite-State Automation or simply a State Machine, is a mathematical model of computation used to design both computer programs digital logic".

→ It also seems as a mathematical abstraction.

→ It is a behaviour model composed of a finite no. of states.

→ The machine is in only one state at a time.

→ The state it is in at any given time is called the current state.

→ It can change from one state to another when initiated by a triggering

Name of Lecturer : .....

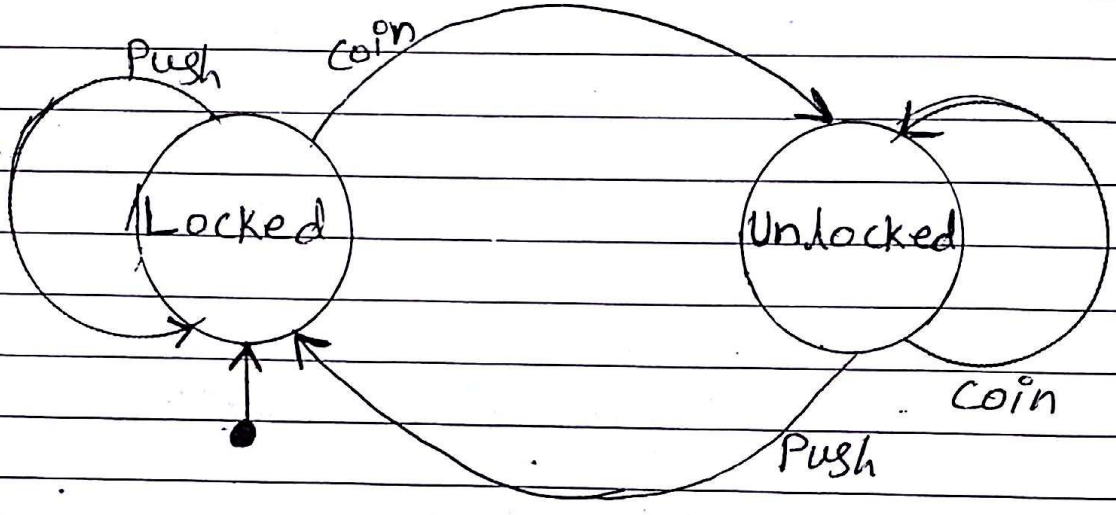
event or condition; this is called a transition.

→ A particular FSM is defined by a list of its states, & the triggering condition for each transition.

Example: a turnstile,

→ A turnstile, used to control access to subways & amusement park rides, is a gate with free arms at waist height, one across the entryway.

→ Initially the arms are locked, barring the entry, preventing customers from passing through.



State Diagram for a Turnstile.

→ Deposit a coin or token in a slot on the turnstile unlocks the arms, allowing a single customer to push through.

→ After the customer passes through, the arms are locked again until another coin is inserted.

→ Consider as a state machine, the turnstile has two states: Locked & Unlocked.

→ There are two i/p's that affect its state: Putting a coin in the slot (coin) & Pushing the arm (Push).

→ In the locked state, Pushing on the arm has no effect; no matter how many times the i/p push is given, it stays in the locked state.

→ Putting a coin in - that is, giving the machine a coin i/p - shifts the state from Locked to Unlocked.

→ In the unlocked state, putting additional coins in has no effect; i.e. giving additional coin i/p's does not change the state.

→ However, a customer pushing through the arms, giving a Push i/p, shifts the state back to locked.

→ The turnstile state machine can be represented by a state transition table, showing for each state the new state & the o/p (action) resulting from each i/p.

Current State	Input	Next State	Output
Locked	Coin	Unlocked	Unlock turnstile so customer can push through.
	Push	Locked	None.
Unlocked	Coin	Unlocked	None
	Push	Locked	When customer can push through lock turnstile.

Ref :- R18

3.24 Various Elements of the Analysis Model :-

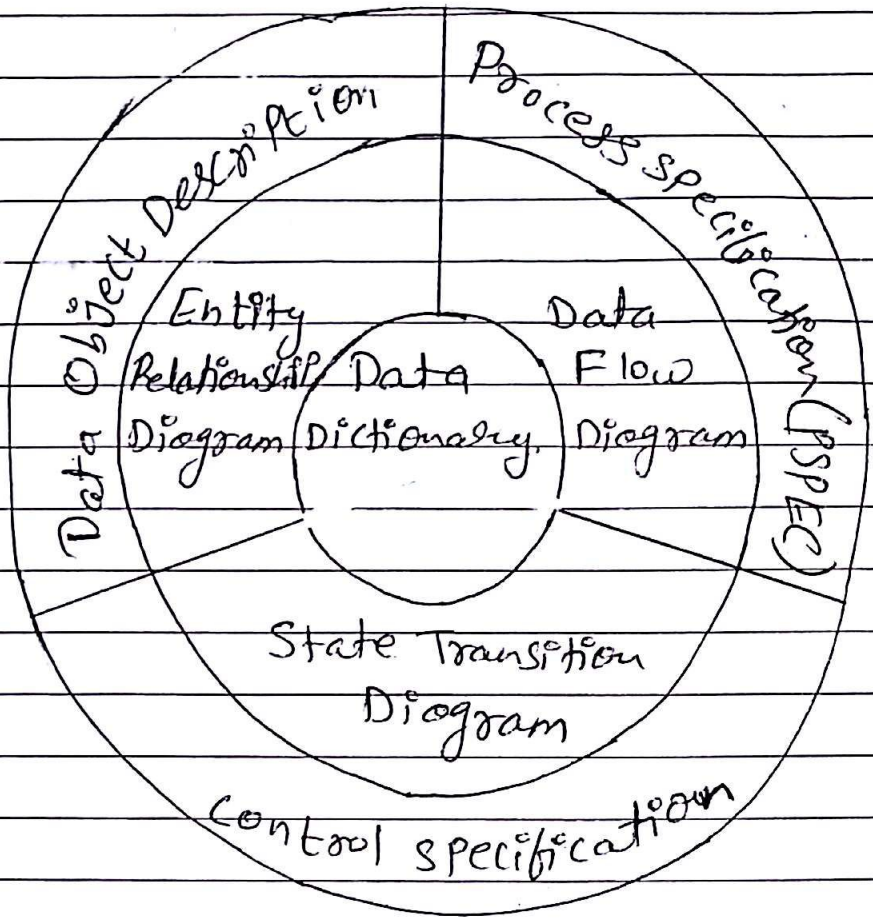
The analysis model must achieve three primary objectives :

1) To describe what the customer requires.

2) To establish a basis for the creation of a s/w design

3) To define a set of req's. that can be validated once the s/w is built.

To achieve these objectives, the analysis model is drawn below.



Structure of Analysis Model

→ **Data Dictionary :-** A repository that contains descriptions of all data objects consumed or produced by the s/w.

→ **Entity Relationship Diagram (ERD) :-**

Depits the relationship b/w data objects. The attributes of each data object noted in the ERD can be described using a data object description.

→ Data Flow Diagram (DFD) :-

- It provide an indication of how data are transformed as they move through th system.
- It depicts the functions that transform the data flow.

→ Process specification (PSPEC) :-

A description of each function presented in the DFD is contained in a Process specification

→ State Transition Diagram (STD) :-

The STD indicates how the system behaves as a consequence of external events. The STD represents the various modes of behavioi (called states) of the syjtem & the manner in which transitions are made from State to State.

→ Control Specification :-

Additional information about the control specifications of the S/w is contained in the Control Specification (CSPEC).

A more detailed discussion of these elements of the analysis model is presented in the next few Points;

Ref :- R19

3.25 Data Modeling :-

Data modeling answers a set of specific questions that are relevant to any data processing application.

→ What are the Primary data objects to be processed by the system?

→ What is the composition of each data object & what attributes describe the object?

→ Where do the object currently resided?

→ What are the relationship b/w diff objects?

→ What are the relationship b/w objects &

Name of Lecturer : .....

the processes that transform them?

To answers these questions, dat modeling methods make use of the entity relationship diagram (ERD), which is described later.

A Data modeling consists of three interrelated pieces of information:

1) Data objects

2) Attributes

3) Relationships

Ref:- R19

Definition:-

3.26.1 Data Objects:- "A data object can be an external entity, (tangible & intangible)"

ex:- Intangible → a report, a display (thing)
→ a telephone call (an occurence)
→ an alarm (an event)

Tangible → Salesperson (a role)
→ warehouse (a place)

A person or a car can be viewed as a data object in the sense that either can be defined in terms of a set of attributes.

Definition

3.26.2 Attributes :- "It defines Properties of a data object."

ex:- Date object — Attribute

car
- → Manufacturere (Ford--).
- → Model No.
- → Bodytype (Sedan, Sports)
- → Colour

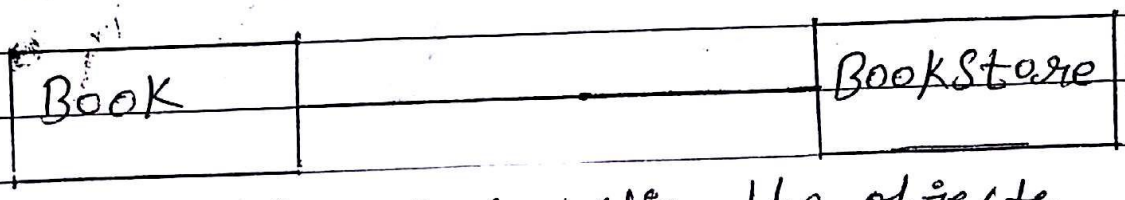
Person
- → Name
- → address
- → Age

3.26.3 Definition of Relationships :-

"Data objects are connected to one another in different ways".

ex:- Consider two data objects; book & bookstore.

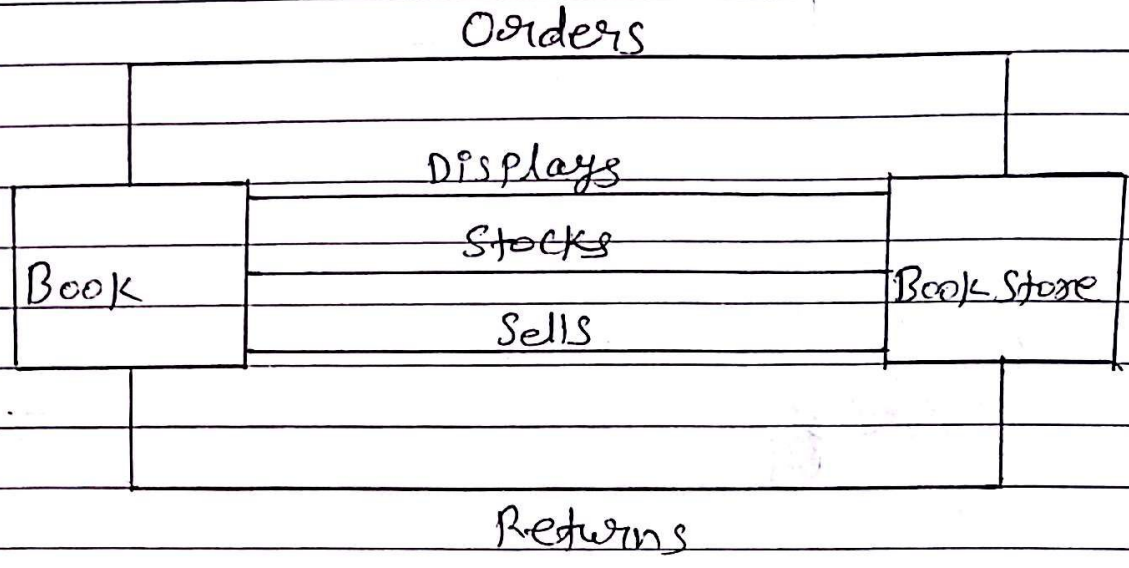
The following relationships are occured b/w these two data objects.



(a) A Basic connection b/w objects.

other relationships b/w these two objects are.

→ A book store orders books.
→ A book store displays books.
→ A book store stocks books.
→ A book store sells books.
→ A book store returns books.



(b) Relationships Between Data Objects

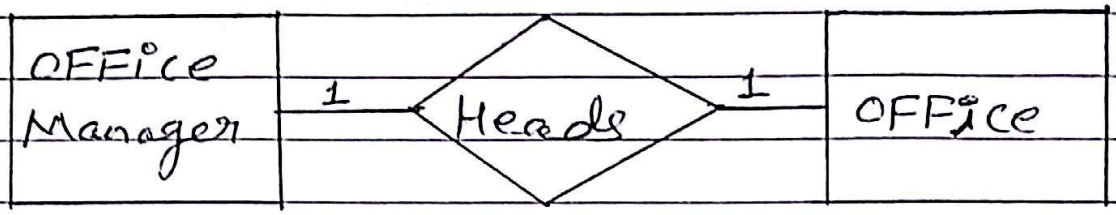
Three main types of relationship occurs.

1) One-to-One Relationship (1:1)
2) One-to-Many Relationship (1:M)
3) Many-to-Many Relationship (M:M)

1) **One-to-one Relationship (1:1) :-**

An occurence of object 'A' relate to one & only one occurence of object 'B' & occurence of 'B' can relate to only one occurence of 'A'.

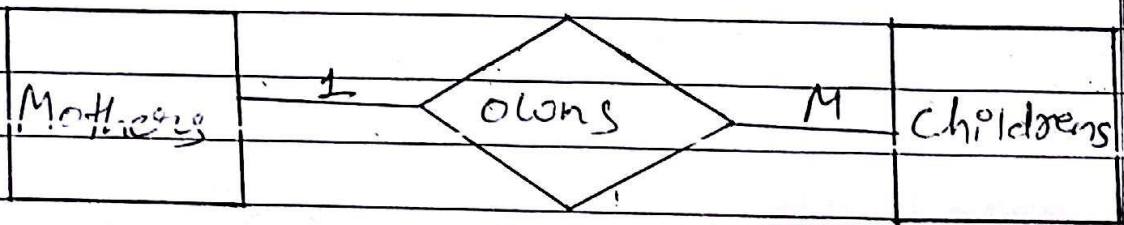
example :- Office manager heads only one office



2) **One-to-Many Relationship (1:M) :-**

One occurent of object 'A' can relate to one or many occurences of object 'B', but an occurence of 'B' can relate to only one occurence of 'A'

Examples- a mother can have many children, but a child can have only one mother.

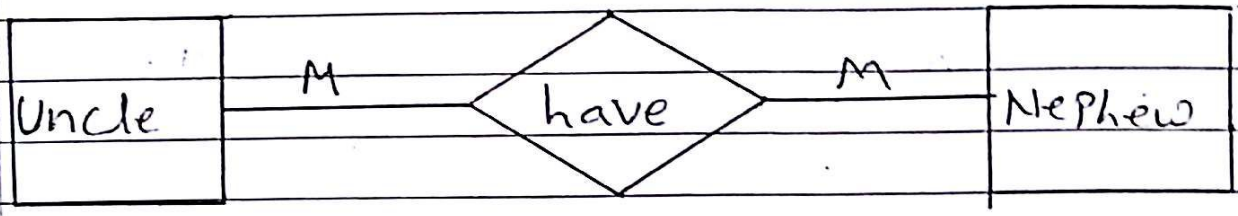


Name of Lecturer : .....

3) Many-to-Many Relationship (M:M) :-

An occurence of object 'A' can relate to one or more occurences of 'B' while an occurence of 'B' can relate to one or more occurences of 'A'

example :- An uncle can have many nephews, while a nephew can have many uncles.



Ref :- R19

3.27 Definition of Cardinality :-

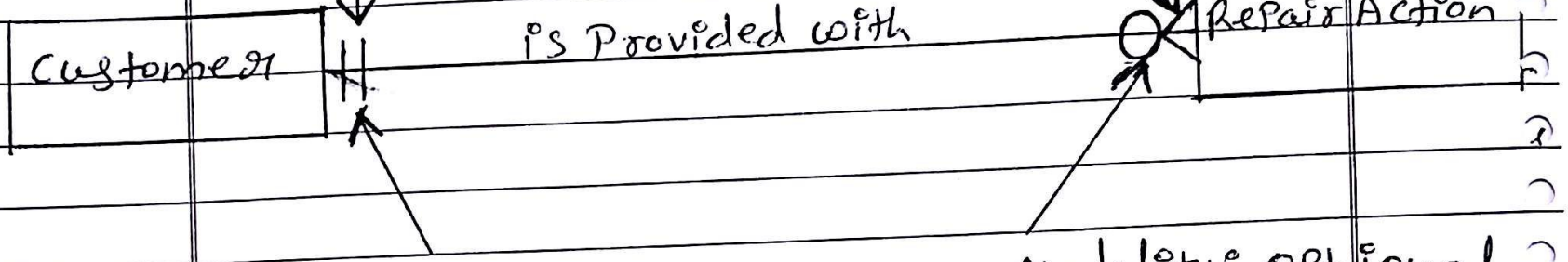
"Cardinality is the specification of the no. of occurences of one object that can be related to no. of occurences of another object".

Definition of Modality :- "The modality of a relationship is 0 if there is no explicit need for the relationship

to occur or the relationship is optional. The modality is 1 if an occurence of the relationship is Mandatory: i.e. not optional."

Cardinality:
implies that a single customer awaits repair action(s)

Cardinality:
implies that there may be many Repair action(s)



Modality: Mandatory
implies that in order to have a repair action(s) we must have a customer

Modality: Optional
implies that there may be a situation in which a repair action is not nece

Cardinality & Modality

3.28 Entity Relationship Diagram (ERD) :-

Definition:- "An Entity-Relationship Diagram (ERD) is a graphical model or representation of the information system that depicts the relationships among system entities".

Ref. :- R20

3.29 Creating an ERD :-

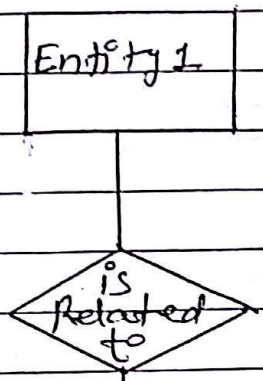
1) Identify the entities.

2) Determine all significant events, transactions, or activities that occur b/w two or more activities.

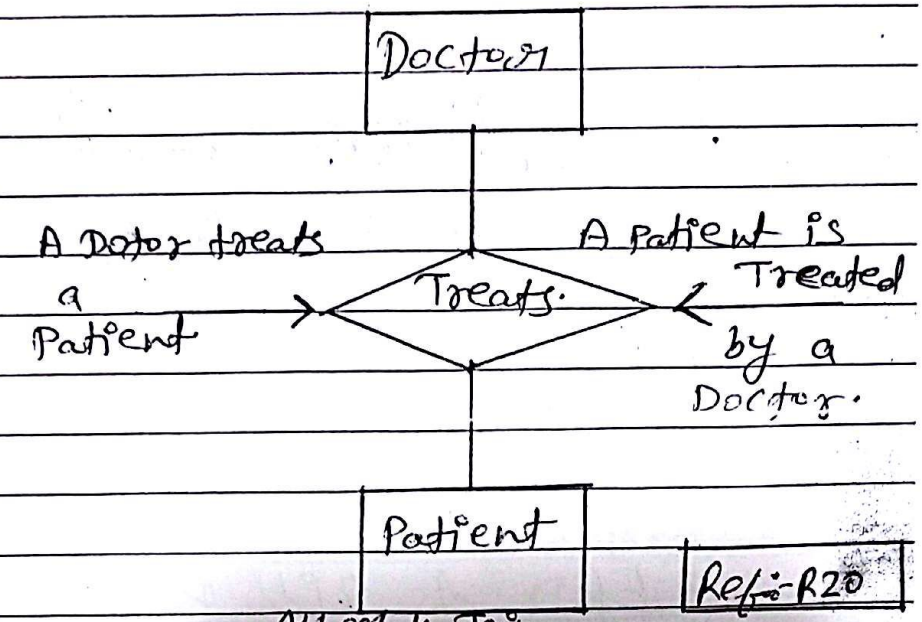
3) Analyze the nature of the interaction.

4) Draw the ERD.

Syntax

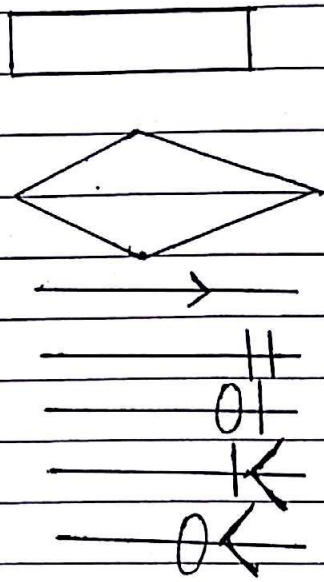


example



Ref:- R20

Symbols of ERD & Cardinality.



Entity

Relationship

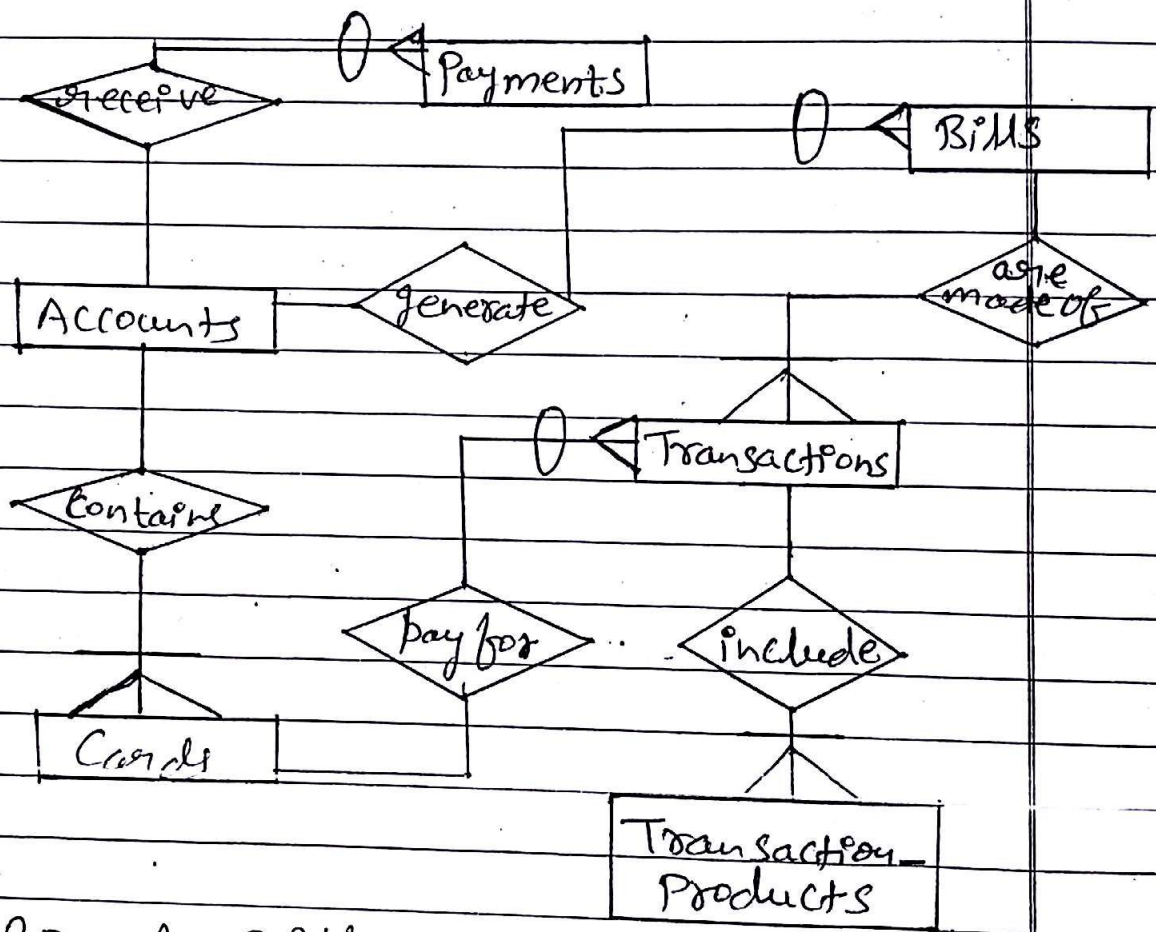
Associated object

Cardinality - Exactly one

Cardinality - Zero or one

Cardinality - Mandatory Many

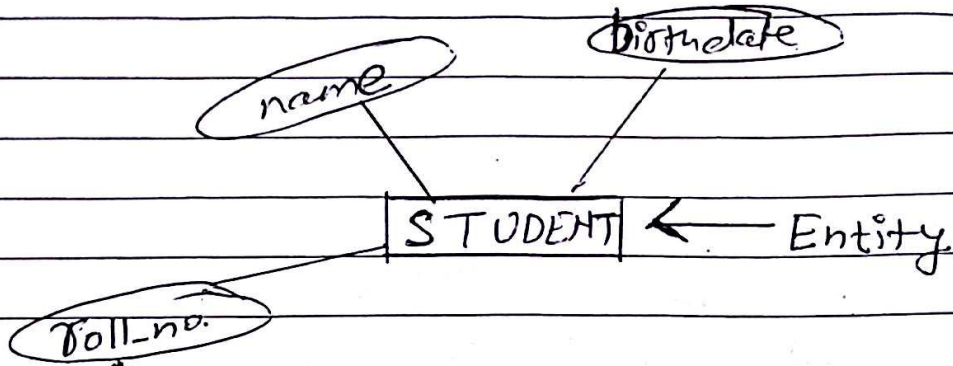
Cardinality - Optional Many



ERD of Bill payment

Ref:- R27

Name of Lecturer : .....



Entity- Attribute diagram

Ref. :- R20

3.30 **Functional Modeling :-**

Definition :- "A functional modeling in s/w Engg. is a structured representation of the functions (activities, actions, processes, operations) with in the modeled system".

A function model is a graphical representation of an enterprise's function within a defined scope. The purpose of the function model are to describe the functions & processes, assist with discovery of information needs, help identify opportunities, & establish a basis for determining product & service tools.

Ref. :- R21

3.31 Data Flow Diagram:- (DFD)

Definition:- "A Data Flow Diagram (DFD) is a graphical representation to depict the flow of data through a system & the work or Processing Performed by that system".

→ It shows how data moves through an information system but does not show program logic or processing steps.

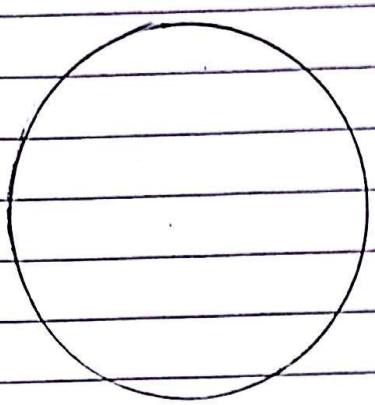
→ It only represent a logical model that shows what the system does, not how it does it.

* Why DFD?

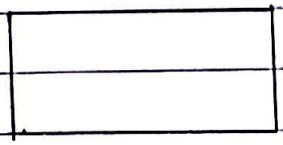
→ Language Description is subject to interpretation, it may omit crucial info.

→ Graphical Description of the flow of data within an organization with DFD covers all crucial info.

3.32 Symbols Used in DFD's :-

1)  → Process / Function Symbol.

2) → Data Store symbol

3)  → External Entity Symbol.

4) → → Data Flow.

De Marco & Yourdon Notations.

Description of DFD Symbol :-

1) Process / Function Symbol :

→ A process is a work or action performed on i/p data flow to produce an o/p data flow.

→ A Process must have at least one i/p data flow & at least one o/p data flow.

→ Ex:- Apply rent payment, Verify order, Pay Bill.

2) Data Store Symbol :-

A data store is a storage of data: It contains information.

Data stores depicted on a DFD store all instances of data entities

Ex. :- Accounts Receivable.

3) External Entity Symbol :-

→ An External Entity is a provider (source) or receiver (sink) of data & info of the system.

→ An External Entity is not Part of the system: the externality depends on how the system is defined.

Name of Lecturer : .....

→ Ex.:- Customer, Student, Supplier.

4) Data Flow :-

A data flow represents a movement of data (information) among processes or data stores.

A data flow does not represent a document or a physical good; it represents the exchange of information in the document or about the good.

A data flow represents an i/p of data to a process, or the o/p of data from a process.

Ex. :- Deposit, invoice payment, delivery slip.

Ref. :- R20

3-33 Creating a Data Flow Diagram :-

Here are some guidelines to create a DFD :

1) Level 0 DFD should depict the entire system as single bubble along with all the related entities processing with the system.

2) Primary input should be carefully noted.

3) All arrows & bubbles should be named with meaningful names.

4) Information flow Continuity must be maintained at the next level.

5) only one bubble should be refined at a time

6) A good DFD should not Possess :

→ loops.

→ a process which is taking a pure decision.

→ Flow lines crossing each other

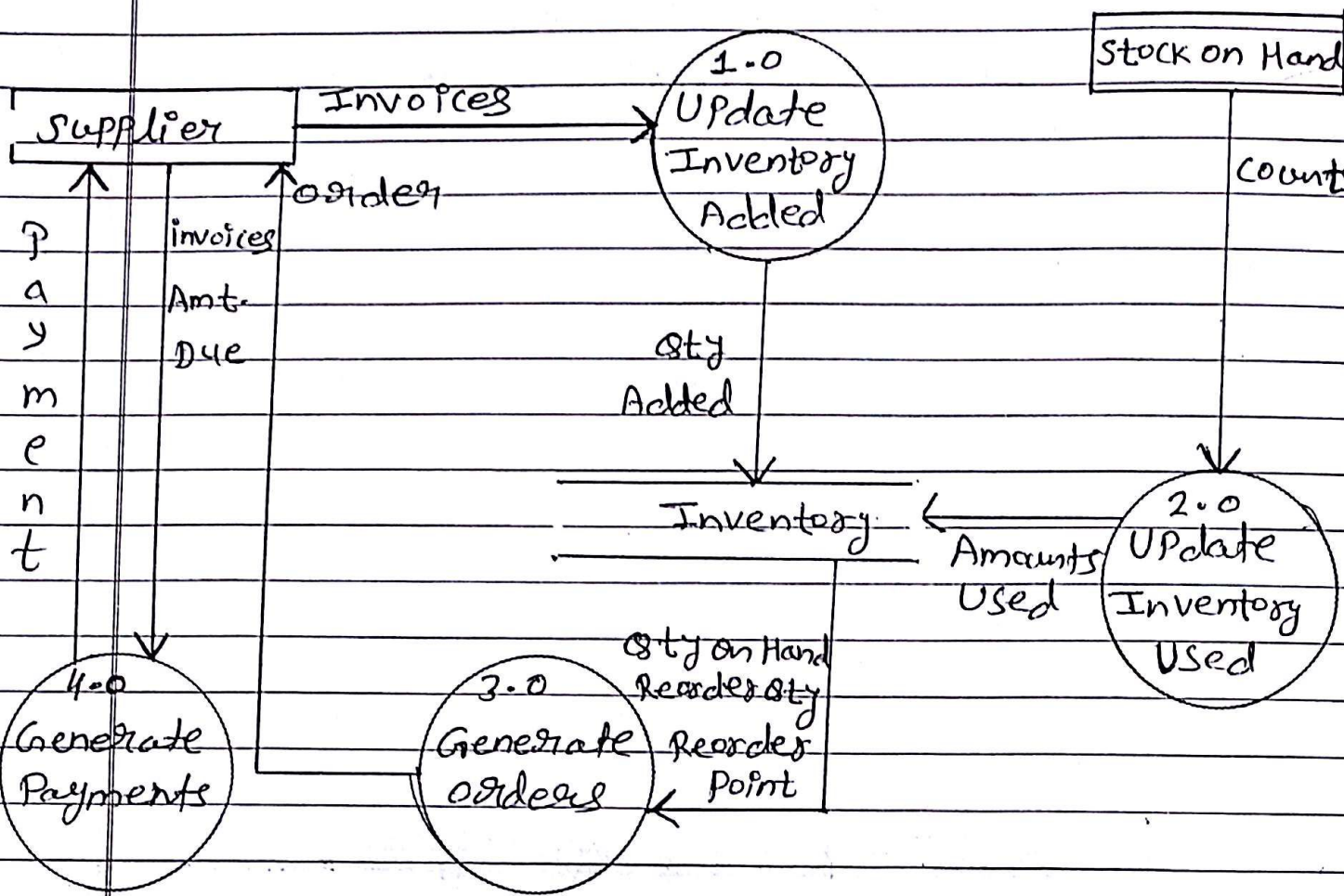
→ Splitting of a data flow into difficult flows with different meaning.

Ref :- RS; P.No:→227

3.34 Types of DFD [Logical & Physical DFD's] :-

Data flow diagrams are either Logical or Physical.

1) **Logical DFD :-** This type of DFD concentrates on logical Process that are to be performed by the System, and flow of data in the System.

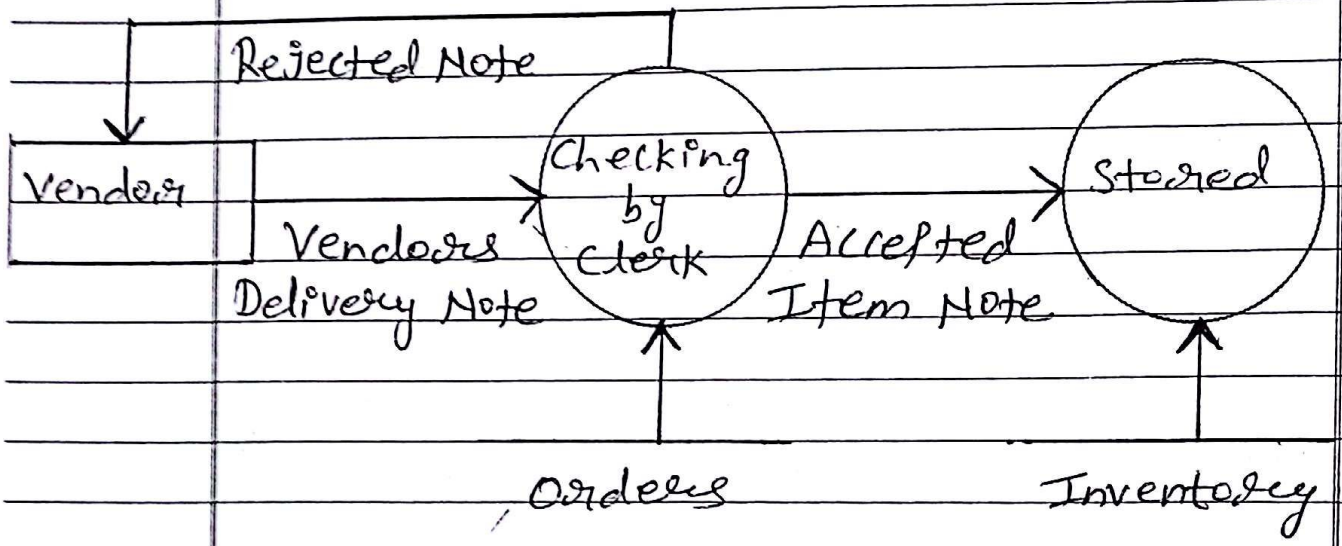


Logical DFD of Inventory Activities

Ref :- R22, R23

2) Physical DFD :-

This type of DFD shows how the data flow is actually implemented in the system. It represents objects which perform the logical operations. It is more specific & close to the implementation.



Physical DFD of Inventory Activities

When vendor requests for his delivery note then it is checked by the clerk; if accepted then stored otherwise if rejected then returned back.

3.35 | Leveling of DFD's:-

Definition:- "Leveling is the process of drawing a series of increasingly detailed diagrams, until the desired degree of detail is reached".

Leveling DFD's are of 3-types.

1) 0-Level DFD

2) 1-Level DFD

3) 2-Level DFD

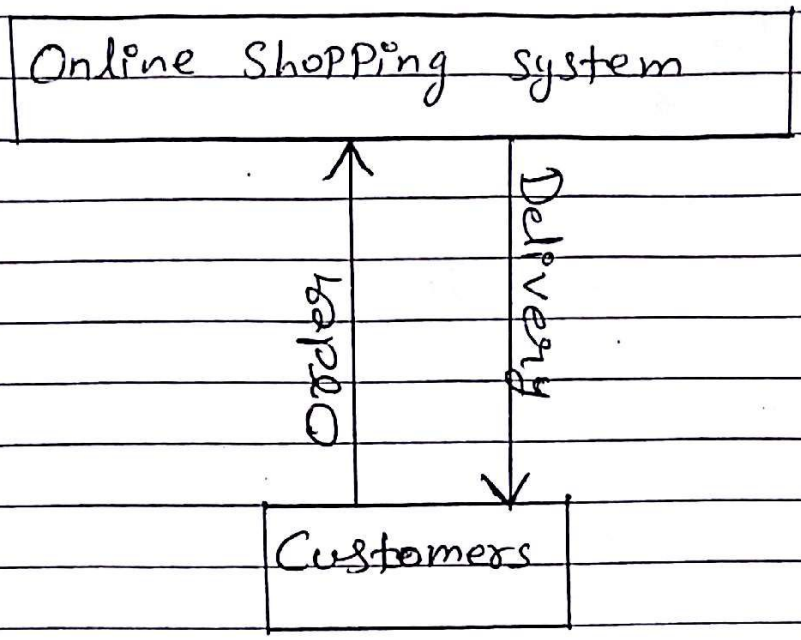
1) 0-Level of DFD:-

→ Highest abstraction level DFD is known as 0-Level DFD.

→ It depicts the entire information system as one diagram concealing all the underlying details.

→ Level 0 DFDs are also known as context level DFD.

→ It identifies external data flows (input, output).



0-Level DFD of Online Shopping System

2) 1-Level DFD :-

→ The level 0 DFD is broken in to more specific level; i.e. Level 1 DFD.

→ Level-1 DFD depicts basic modules in the system & flow of data among various modules.

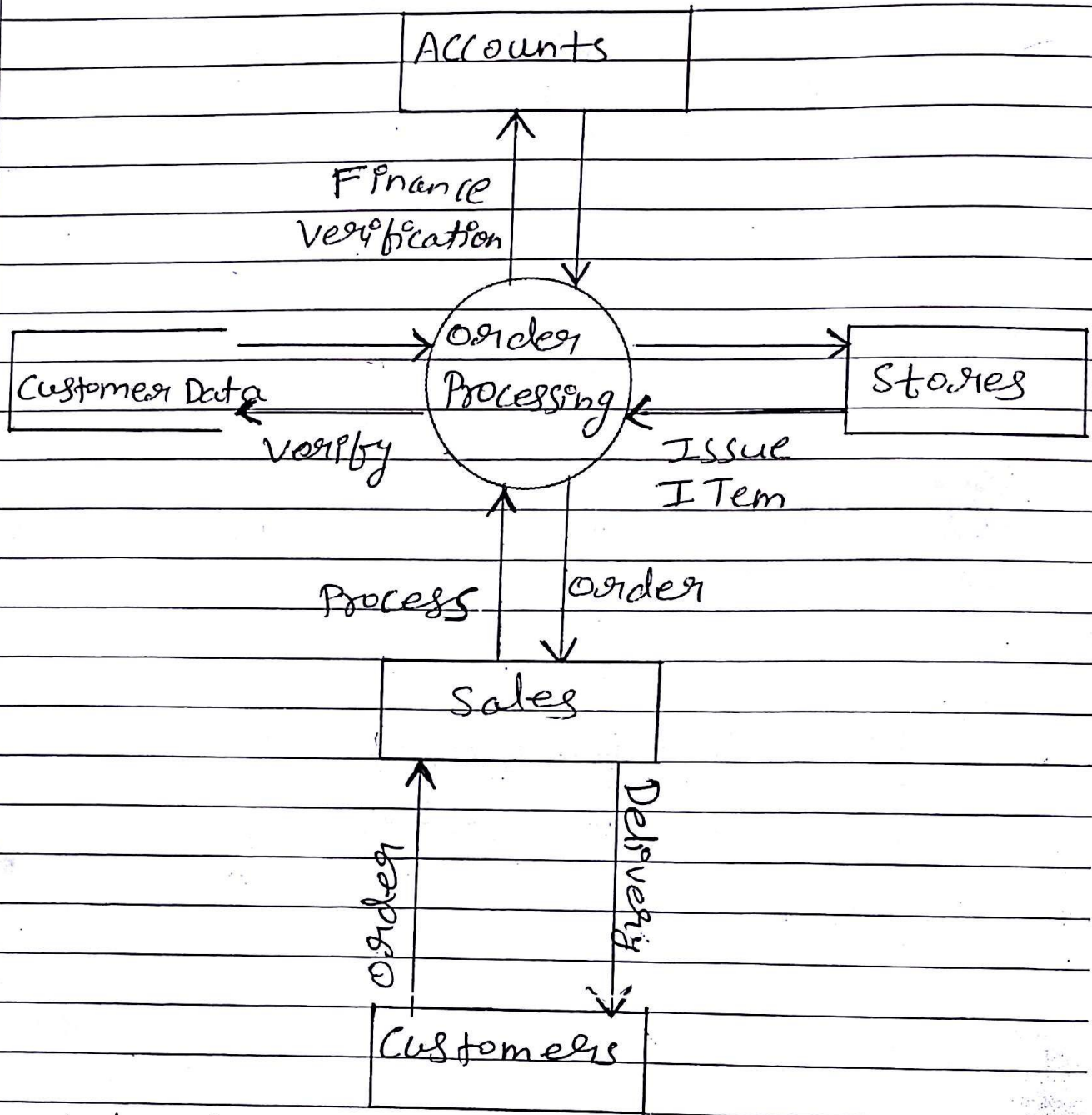
→ Level-1 DFD also mentions basic Processes & Sources of information.

→ It identify external data flows b/w

Name of Lecturer : .....

external entities & Processes.

→ It identify internal data flows b/w Processes & data stores.

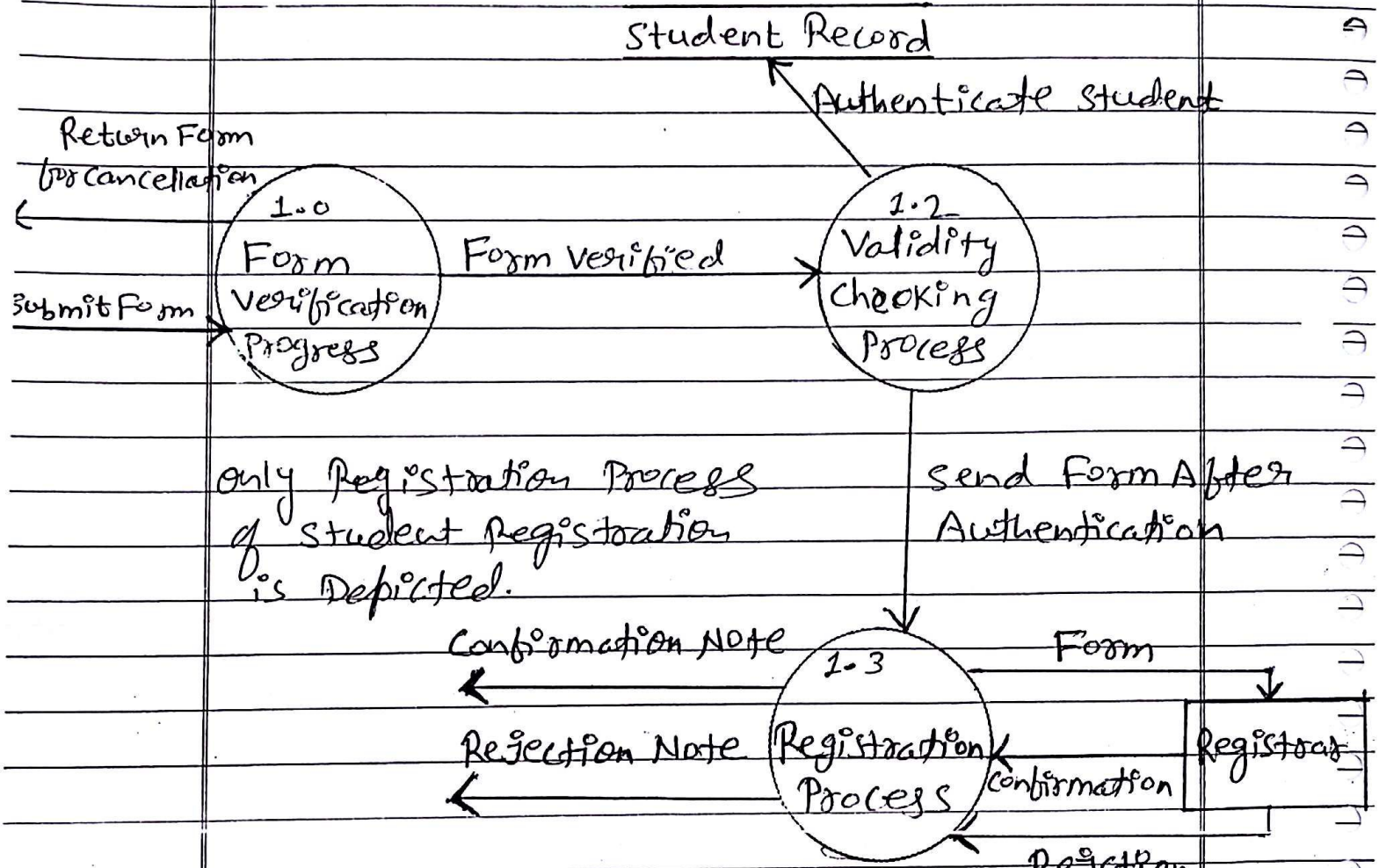


1-Level DFD of Inventory system.

3) 2-Level DFD :-

→ At this level, DFD shows how data flows inside the modules mentioned in level 1.

→ Higher level DFD's can be transformed into more specific lower level DFDs with deeper level of understanding unless the desired level of specification is achieved.



2-Level DFD of Student Registration

Ref :- R2

Name of Lecturer : .....

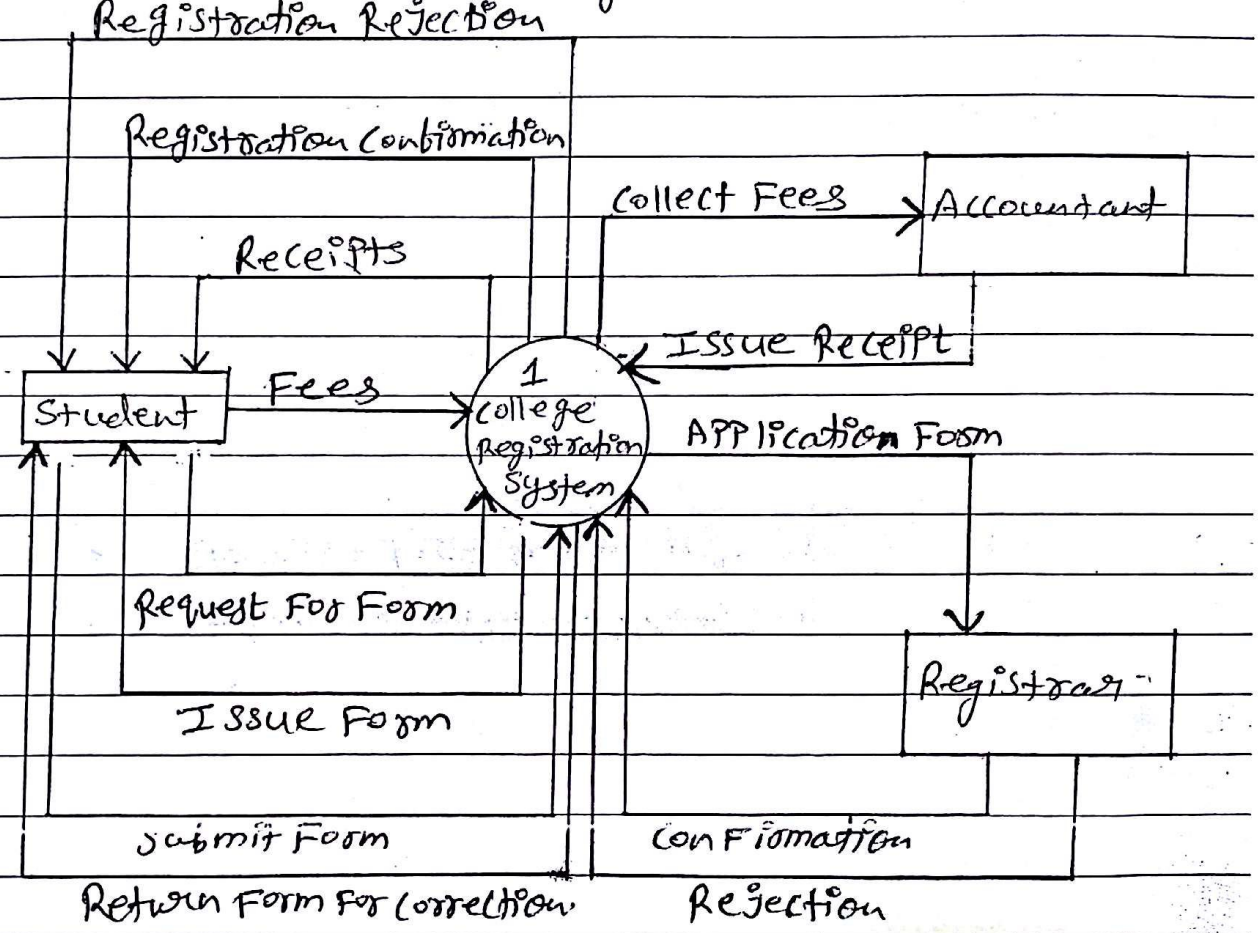
3.36 Case study of DFD :-

→ Case study of college registration system.

→ In this case study Student will registered himself with the help of college registration system.

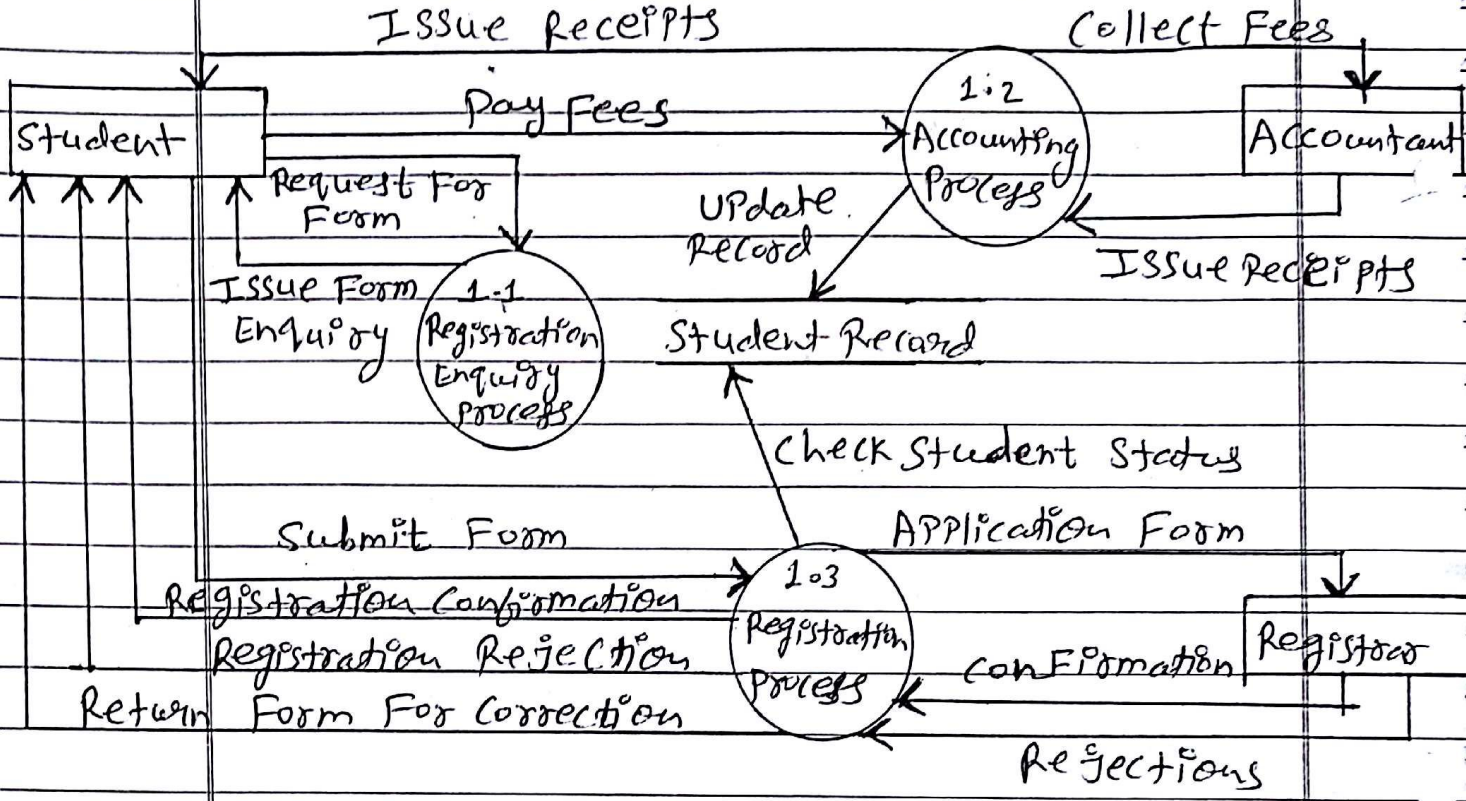
→ Student will enter the fees, get an application form & get signed by registrar (if form is correct) Student get registered in the college.

→ 0-Level DFD of College Registration system.



0-Level DFD For College Registration System.

→ 1-Level DFD of College Registration System.



1-Level of DFD For College Registration System.

→ Above 1-level DFD depicts the relation b/w different Process/Function & data Store & different entity.

→ Processes are:

→ Registration Enquiry process
→ Accounting Process
→ Registration Process

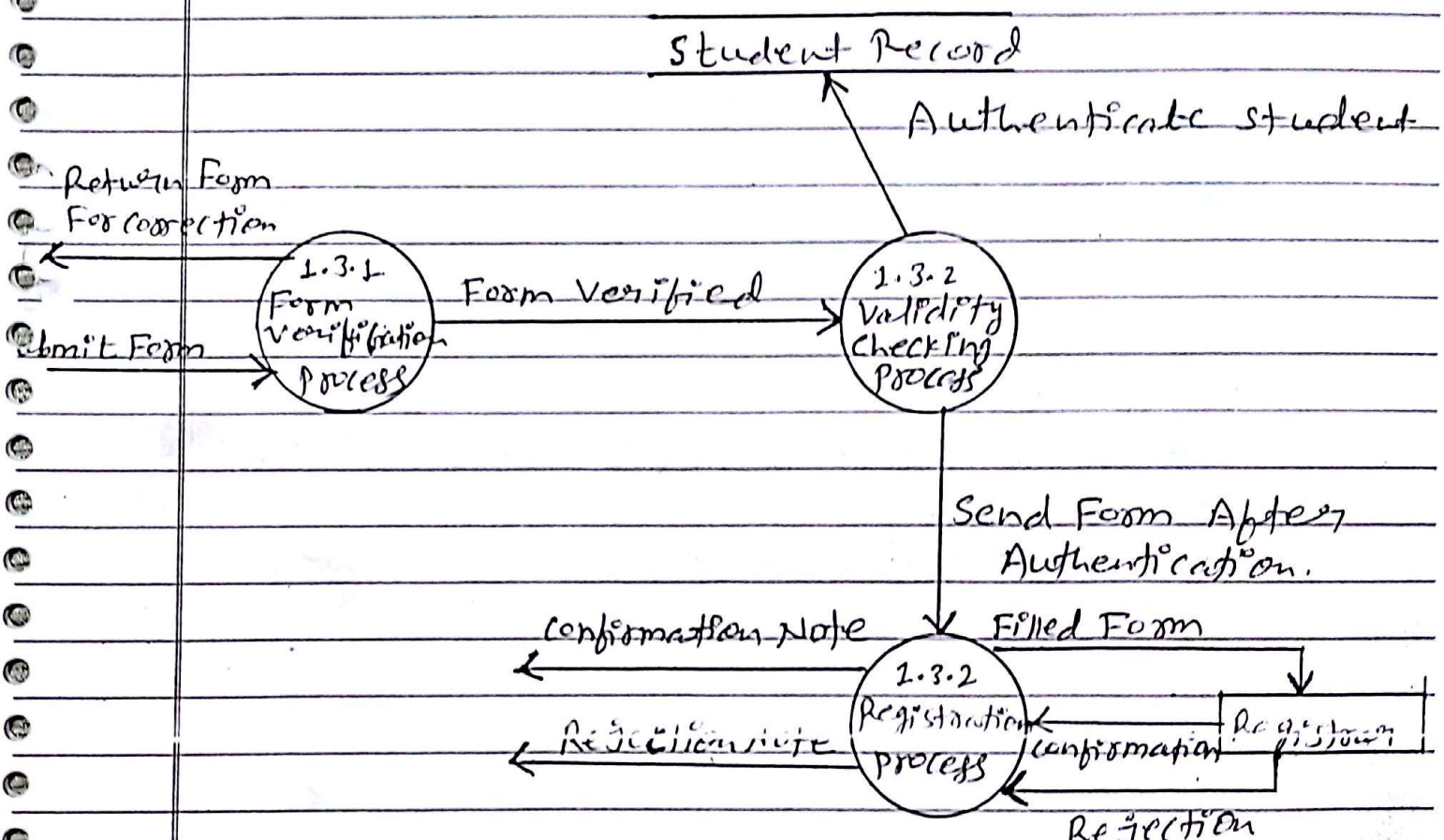
→ Date Store are :

→ Student Record.

→ Entities are :

→ Student
→ Accountant
→ Registrar.

→ 2-Level DFD of College Registration system.



2-Level DFD of Registration Process (1.3)

→ 2-Level DFD depicts the Student Registration Process.

→ It depicts different process in student registration.

→ Form verification process

→ Validity checking process

→ Registration process.

→ It depicts data store

→ Student Record

→ It depicts entity

→ Registrar.

→ The other two processes i.e. Registration enquiry Proces & Accounting Process is scheduled to give as an assignment to students.

3.37 Control Flow Diagram (CFD):-

Definition:-

"A control flow Diagram (CFD) is a diagram to describe the control flow of a business Process or Program."

OR

"A control flow diagram is another name for a program flowchart & will show the flow of control through the different paths of the Program".

It is a graphical notations specially designed to represent event & control flows.

Ref:- R25, R26

3.37.1 Types of control Flow Diagram (CFD):-

There are two types of CFD

1) Process Control Flow Diagram
2) Performance seeking control Flow Diagram.

1) Process control Flow Diagram:-

→ A flow diagram can be developed for Process

control system for each critical activity.

→ It includes text information in Program defined language & Mathematical equations.

2) Performance Seeking Control Flow Diagram:-

→ The figure presents an example of a performance seeking control flow diagram of the algorithm.

→ The control law consists of estimation, modeling, & optimization processes.

3.38 Creating a Control Flow Diagram:-

→ Begin by stripping (removing) all the data flows arrows from the DFD.

→ Solid arrows (for Events) & dashed arrows (for control items) are added to the diagram.

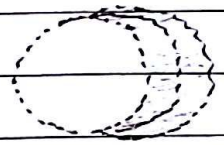
→ Create control specification (CSPEC) for each bubble in final CFD.
→ Contains an STD (State Transition Diagram)

that serves as a sequential specification of the bubble's behavior.

Ward & Mellor Notations of CFD.

→ Solid Arrow for Events

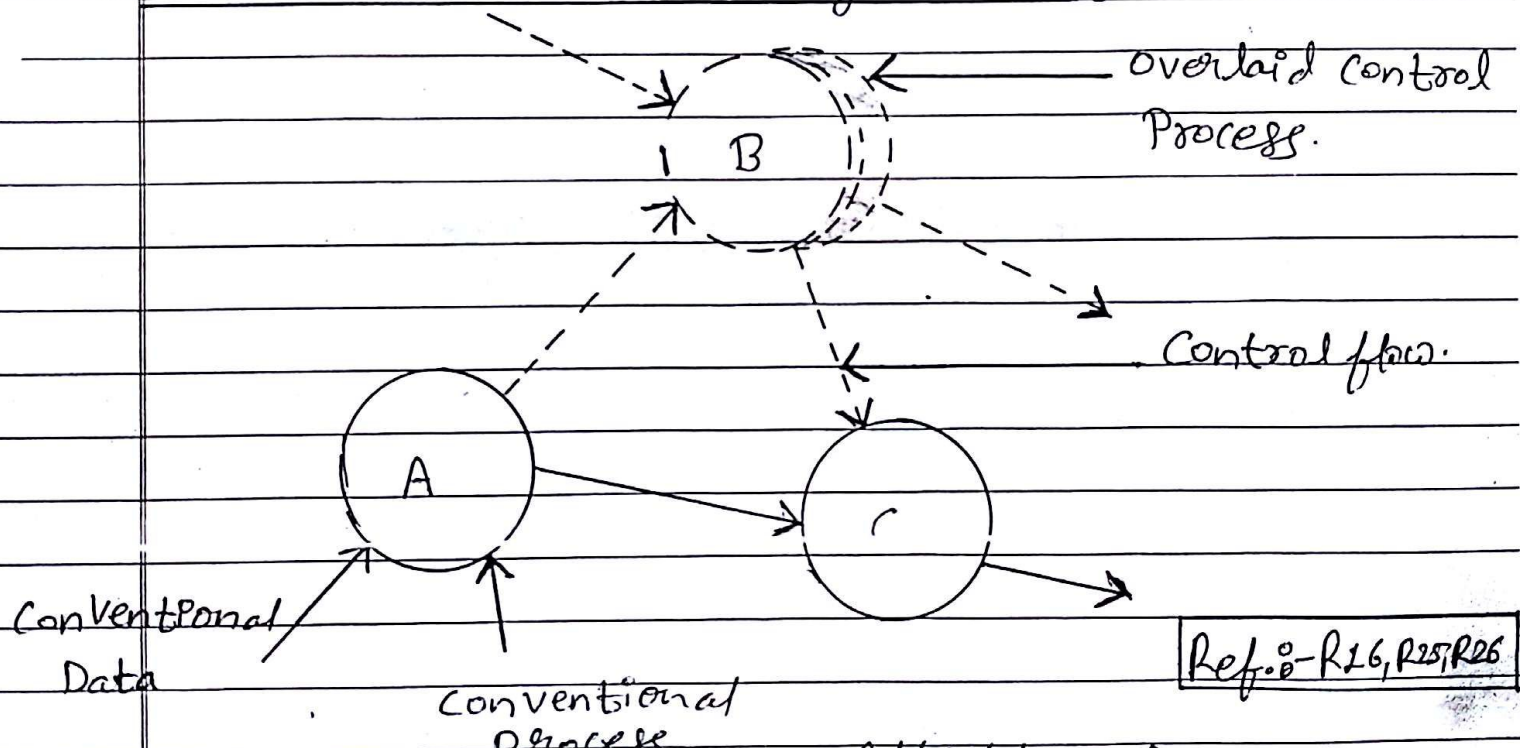
- - - - - - → Dashed Arrow for control Items



multiple instance overlays.

In some cases, multiple of the same control or data transformation occurs; Ward & Mellor notations used to represent such multiple instances simply overlays such process bubbles to indicate multiplicity.

Data & Control Flows Using Ward & Mellor Notations



Ref:- R16, R25 R26

3.39 Behavioral Modeling :-

Definition :- "Behavioral Modeling is an operational principle for all requirements analysis methods".

→ The State transition diagram (STD) represents the behavior of a system by depicting its state.

→ Events that cause system to change state.

→ The STD indicated what actions are taken as a consequence of particular event.

→ A state, is any observable mode of behavior.

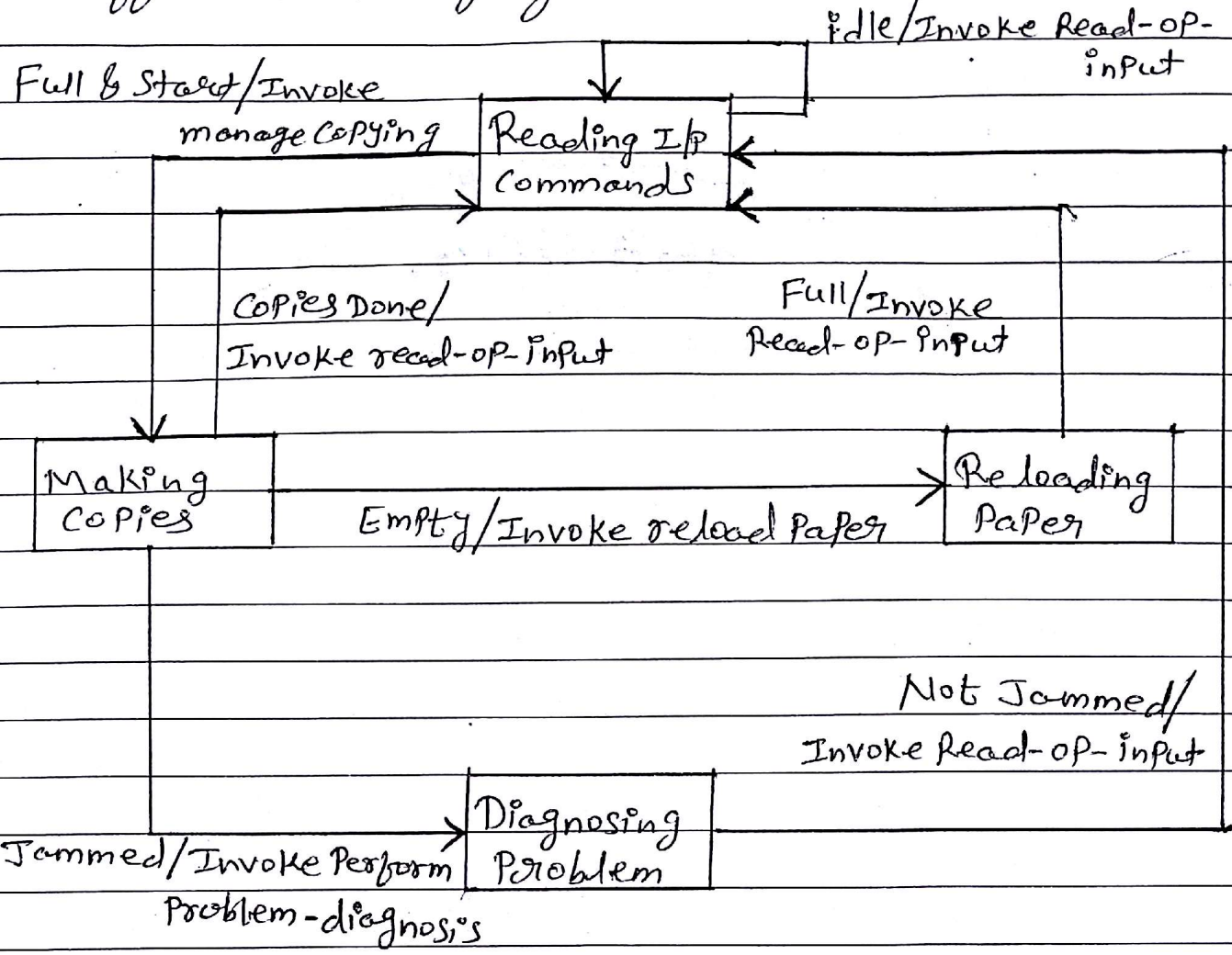
→ State transition is the movement from one state to another. state.

→ An Event, is an occurence that causes the system to exhibit some predictable form of behavior.

→ An Action is a process that occurs as a consequence of making a transition.

Name of Lecturer :

Ex:- Consider software embedded within an office Photocopying Machine.



State Transition Diagram for Photocopier Software

→ In the above STD the rectangles represent system states & the arrows represent transitions b/w states.

→ Each arrow is labeled with a ruled expression

→ The top value indicates the event(s) that

Name of Lecturer : Abhishek Jain

cause the transition to occur.

The bottom value indicates the action that occurs as a consequence of an event during transition.

When the paper tray is full & the start button is pressed, the system moves from the reading command/s state to the making copies state.

Photocopies s/w will make photocopies only when it is not in jammed state & exists in start state.

If it is in jammed state, then the problem is diagnosed & removed.

Note that states do not necessarily correspond to processes on a one-to-one basis.
for eg:- the state making copies would encompass both the manage copying & Produce user displays processes.

Ref. of R19

3.40 **Various Tools of Structured Analysis :-**

Various tools of structured analysis are

1) **Data Flow Diagram :-**

Definition :- "A data flow diagram is a graphical representation to depict the flow of data through a system & the work or processing performed by that system".

Rest of the details discussed in 3.31

Ref. :- R20

2) **Data Dictionary :-**

Definition :- "The data dictionary is an organized listing of all data elements that are used to build a system, with precise rigorous definitions so that both user & system analyst (Programmer) will have a common understanding of i/p's, o/p's, processing & storation".

Ref. :- R3, R17

Rest of the details discussed in 3.21

Name of Lecturer : .....

3) Entity Relationship Diagram:-

→ It is a graphical tool; easy to ready by analysts.

→ Commonly used; well understood.

→ Data objects & relationships are portrayed independently from the processes.

→ Can be used to design database architecture.

→ Effective tool to communicate with DBAs.

Ref:- R27

4) State Transition Diagram (STD):-

It models a real time behavior of any S/W System.

STD represents the behavior of any System by depicting its state.

Events cause System to change state.

Ref:- R19

Name of Lecturer : .....

5) Structure Charts :-

→ In this system maintainability with the help of modularity.

→ It provides a means for transition from analysis to design.

→ It provides a synchronous hierarchy of modules.

Ref. :- R27

6) Event List :-

→ Provides guidence for functionality.

→ Provides a list of system i/p's & o/p's.

→ Means of Req's summarization.

→ can be used to define test cases.

Ref. :- R27

Name of Lecturer : Abhishek Jain