3) Analysis & Design Modelling

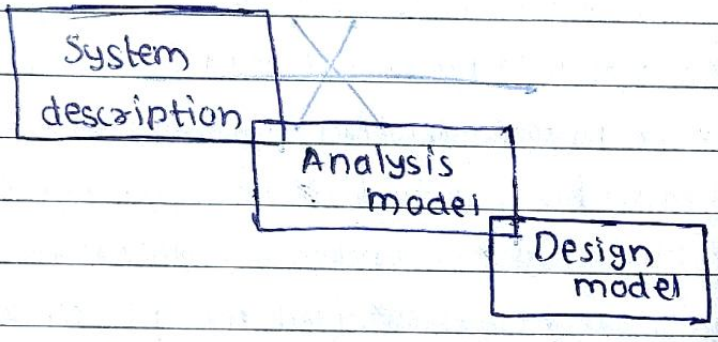
Page No. 28
Date

Analysis modelling :-

Analysis model make use of text and diagrammatic forms in a combination to represent requirements of data, functions and behaviours for easy understanding of overall requirements of the software to be built.

Analysis model validate the software requirements and represents them in multiple dimensions.

Need of analysis modelling :-



① The analysis model acts as a bridge between system description and design model. After considering the requirements at the system level in 'system description', the analysis model focuses on the system requirements

② The information, functions and behaviour of the system defined by analysis model are translated into architectural interface and component level designs in 'design modelling'.

Objectives of Analysis modelling :-

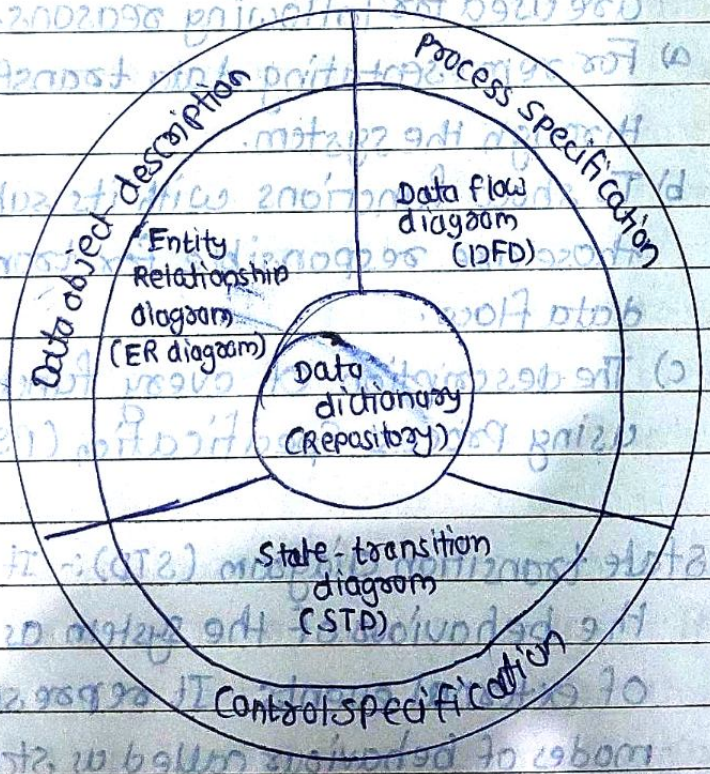
① To state clearly what the customer needs exactly

② To establish the basis of the 'design model'. The information, functions and behaviours of the system defined by 'analysis model' are translated into architectural, interface and component level designs in 'design modelling'.

③ To define a set of requirements, should be validated once the software product is built

Analysis modelling approaches :

Structured Analysis :- (Concept)



At the centre of the model is data dictionary. It is a repository that stores / contains descriptions or information of all data objects used or created by software.

The surrounding area of the core is occupied by different diagrams. Types of diagrams are:

① Entity Relationship diagram (ER diagram):

It is mainly used to represent the relationship between two entities or data objects. The ~~ERD is the~~ ER Diagram is the notation that is used to conduct the data. This data is used to execute data modelling activity.

The additional information about data objects can be given with the help of data object description.

② Data flow diagram (DFD) :- These diagrams are used for following reasons.

a) For representating data transformation through the system.

b) To show functions with its sub-functions those are responsible for transforming the data flow.

c) The description of every function is written using Process Specification (PSPEC)

③ State transition diagram (STD) :- It indicates the behaviour of the system as an outcome of external events. It represents different modes of behaviour called as states of the system. It also shows the fashion in which transitions are made from one state to another state.

Object Oriented Analysis :- (Concept)

The basis of object oriented analysis is classes and member, objects and attributes, wholes and parts. The focus of object oriented analysis is objects.

Classes are the collection of data members and operations to be operated on data members. Objects are run time entities that encapsulate data members and member functions present in the classes.

Principles that can be applied for object oriented analysis are listed below:

1. The information domain has to be modelled.
2. Description of function is done
3. Representation of behaviour is done.
4. All three models called data, functional and behavioural are partioned to understand them in the great depth.
5. Models designed at early stage represent the essense of the problem and models designed on later stage provide implementation details.

The principles listed above forms the basis of object oriented modelling analysis. The objective of object oriented analysis approach is to define all classes that are related to the problem that needs to solved

Date

Domain analysis :- Domain analysis is the part of analysis model by finding common requirements in the project or the system. The previously mentioned domains can be again implemented or reused for several projects within the application domain. Here, in specific application domain common objects, common classes, common frameworks can be identified and reused.

Advantages :-
1. Use of such software domain analysis saves the time.
2. It also reduces development cost.

Elements of analysis model :
1] Scenario based elements
2] Flow oriented elements
3] Class based elements
4] Behavioural elements

1] Scenario based elements :- They represent the system in user point of view.
Scenario based elements are :
1. Use case - Text
2. Use case diagrams
3. Activity diagrams
4. Swim Lane diagrams.

2] Flow oriented elements :- They provide necessary information that how data objects are transformed by processing the functions. Flow oriented elements are:
① Data flow diagram
② Control Flow diagram
③ Processing narratives

3] Class based elements :- They define the objects, attributes and relationships. Class based elements are:
① Class diagram
② Analysis diagram
③ CRC models
④ Collaboration diagrams

4] Behavioral elements :- They show the states and classes and impacts on these three states. Class based elements are:
① State diagrams
② Sequence diagrams.

Technical Domain :- Technical domain of the software is related to the common technical requirements which can be shared by many products. For ex:- mobile applications use common facilities called calling, sending messages, access to internet, etc.

Application domain:- The application domain is the common library that contains the classes that can be used by other products to minimize their work. For ex:- in finance & and banking different types of accounts, fixed deposits and mutual funds, insurance loan etc.

In Hospitality application domain is client or guest, restaurant billing, taxes paid as per bills, In Health care application domain could be patient, tests, specific surgeries etc.

Goals of Domain Analysis

1. Find out common requirements specifications
2. To save the time
3. Reduce the repeated or duplicate work
4. Reduction in complication of the project.
5. To make library of classes available.
6. To enhance the portability.

Input and output of Domain analysis:

Domain analysis takes input like technical literature, surveys done for customer, advice of experts on specific topics, prevailing requirements of that topic etc. This data is then analysed and meaningful information comes out from this.

The output of domain analysis involves reusable classes, reusable standards, domain language, functional models, etc.

Date

Building up the Analysis model :-

Analysis modelling begins with data modelling. Here all data objects that are processed within the system, their attributes and relationships between all data object are specified.

The relationship between different data objects is also considered in terms of number of occurances i-e cardinality

Also the relationship is optional or mandatory is represented by modality.

Partioning is the process of decomposing the problem into smaller parts for the purpose of finding out a total solution to the problem. Theoretically in partitioning we establish a hierarchical representation of function or information.

Data modelling :-

The data model is a collection of three interrelated pieces or parts of information known as data object, the attributes and relationship

① Data objects :- Data objects is representation of any composite information. Composite information means something that has number of different properties or attributes or combination of objects.

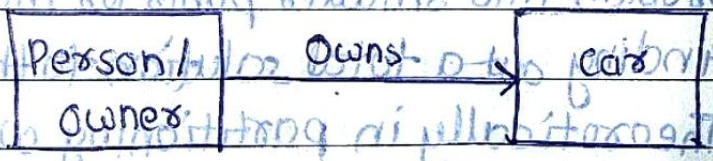
Data object encapsulates data with it self. Data objects do not have direct reference to operation which operates on them.

Date

② Data attributes :- Each data object is having set of attributes. That is data attributes define the properties of data object having following characteristics :-

* Name an instance of data object.
* Describe the instance
* Make reference to another instance in another table

③ Data relationships :- Relationship indicates how data and objects are related to one another

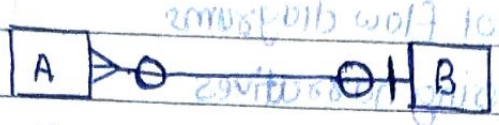


④ Cardinality :- Cardinality represents no. of occurances of one object related to no. of occurances of another object. That is the maximum number of object relationship is represented by cardinality. It can be expressed as :

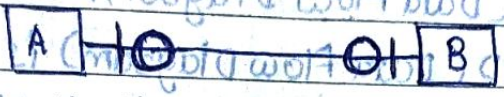
* One to one (1:1)
* One to many (1:n)
* Many to many (m:n)

⑤ Modality :- A modality of relationship is 0 if occurance of relationship is optional and modality of relationship is 1 if occurance of relationship is mandatory.

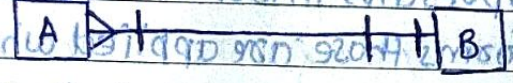
Thus modality specifies the minimum number of relationship.

1] 

Cardinality is n:1 and both roles are optional

2] 

Cardinality is 1:1 and both roles are optional

3] 

Cardinality is n:1 and both roles are mandatory

4] 

Cardinality is m:n and both roles are mandatory

	Cardinality	Modality
1)	Cardinality specifies number of occurances of one object to number of occurances of another object	A modality of relationship is zero if occurance of relationship is optional and one if occurance is mandatory
2)	That is the maximum no of object relationship is represented by cardinality	The modality specifies the minimum number of relationship
3)	Shows different relation-ships like one to one, one to many, many to many	Shows maximum 1 to minimum or compulsory 1

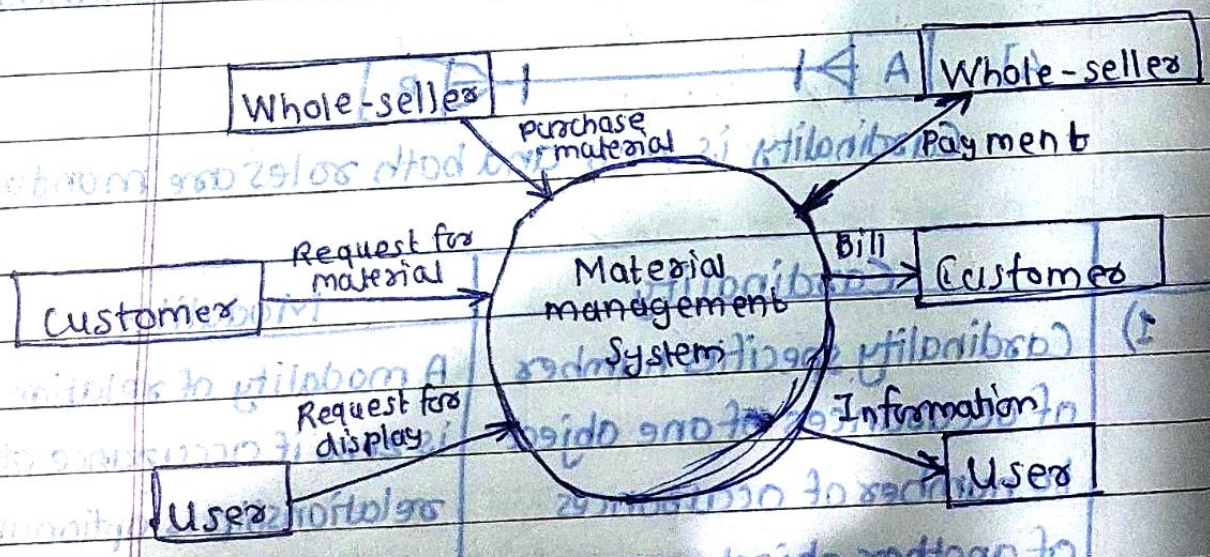
Flow oriented modelling :

They provide necessary information that how data objects are transformed by processing the functions. Flow oriented elements are

* Data Flow diagrams
* Control Flow diagrams
* Processing narratives

① DFD (Data Flow Diagram) :

DFD (Data Flow Diagram) is also called as bubble chart. This is a graphical technique that represents information flow, and transforms those are applied when data moves from input to output.



DFD for medicine material management system

Date

Notations and rules used for DFD :-

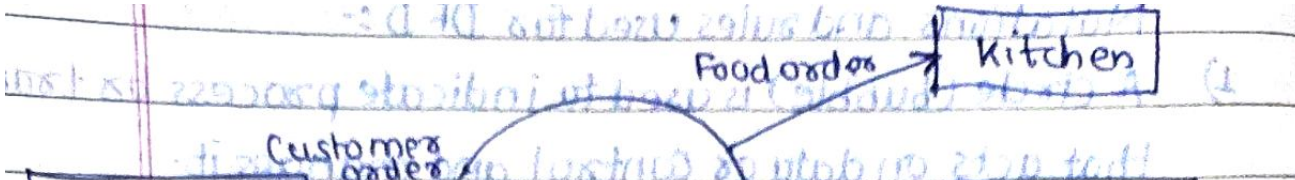
1) A circle (bubble) is used to indicate process or transfor that acts on data or control and changes it.
2) Arrow in the diagrams should be labelled. They show direction of either input or output
3) A double line can be used to represent a data store
4) Entities are represented using Rectangular boxes.

Example diagram

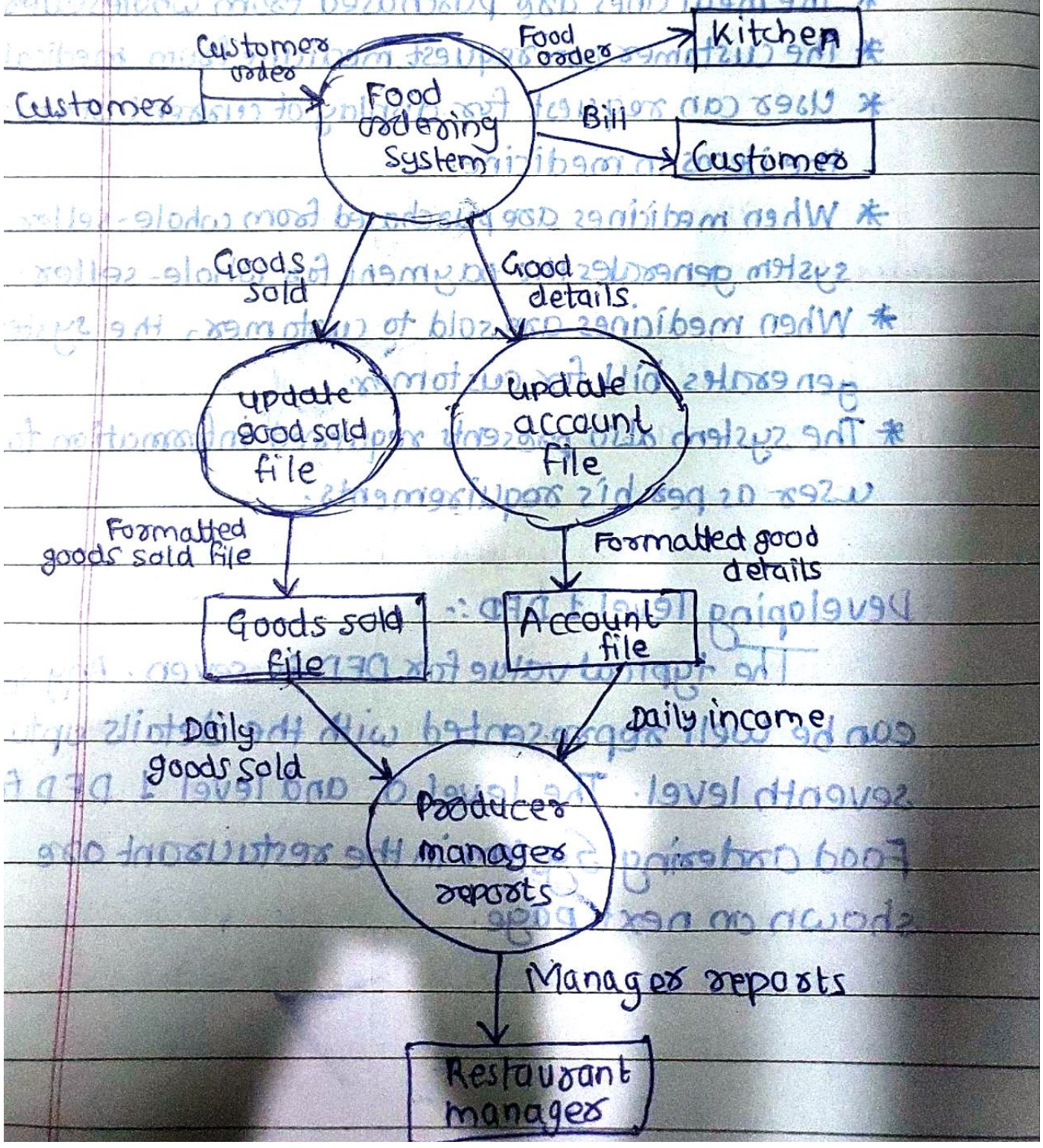
* The medicines are purchased from wholeseller.
* The customer can request medicine from medical store.
* User can request for display of current status of the items in medicine.
* When medicines are purchased from whole-seller, the system generates the payment for whole-seller.
* When medicines are sold to customer, the system generates bill for customer.
* The system also presents required information to the user as per his requirements.

Developing level 1 DFD :-

The typical value for DFD is seven. Any system can be well represented with the details upto seventh level. The level 0, and level 1 DFD for the Food Ordering System in the restaurant are shown on next page.

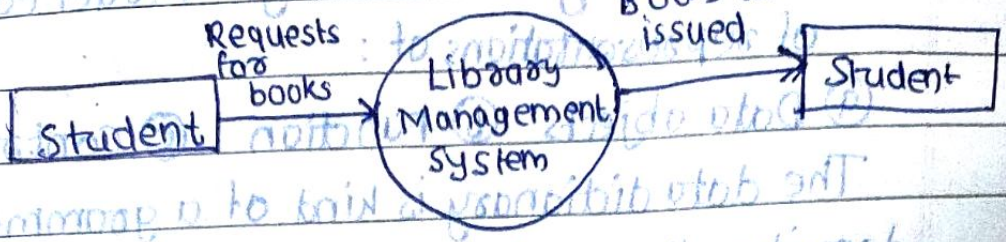

Food order
Kitchen
Customer order
Customer
Food ordering system
Bill
Customer
Management report
Restaurant managers

Level 0 DFD

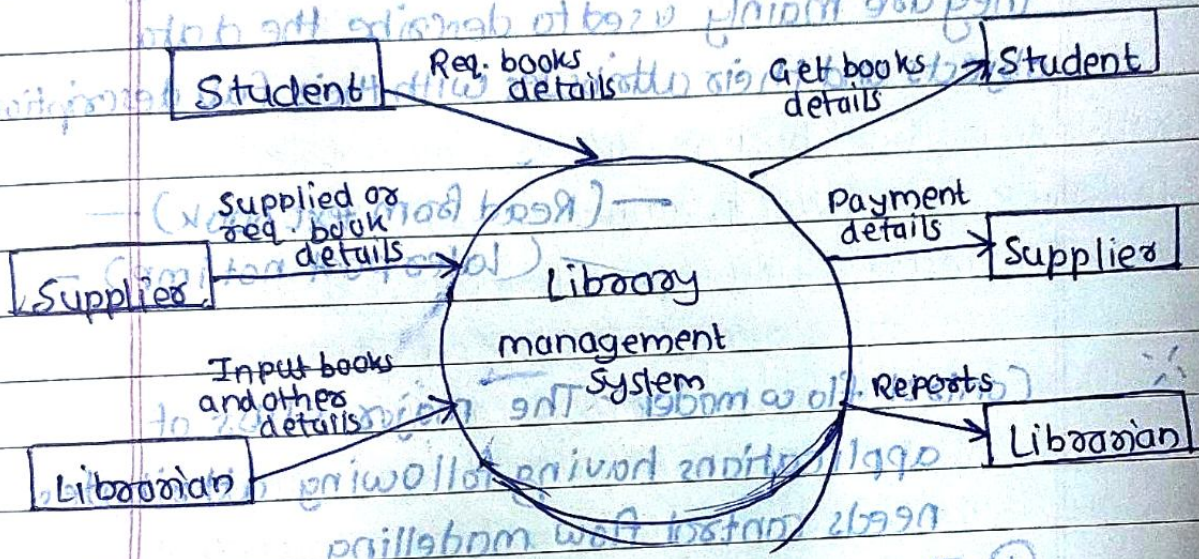

Customer order
Food order
Kitchen
Customer
Food ordering System
Bill
Customers
Goods sold
Good details.
update good sold file
update account File
Formatted goods sold file
Formatted good details
Goods sold file
Account file
Daily goods Sold
Daily income
Producer manager reports
Manager reports
Restaurant managers

DFD for Library Management System

Date

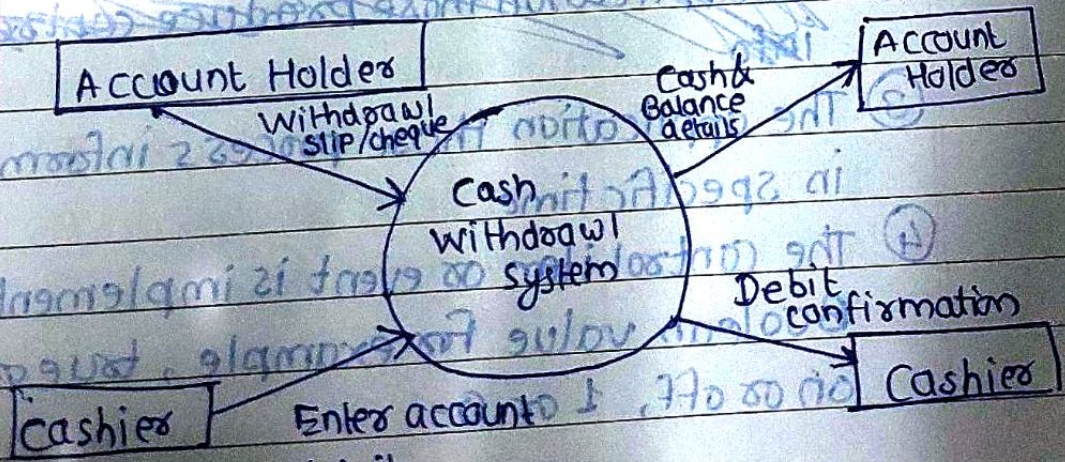


Level 0 DFD



Level 1 DFD

DFD for cash withdrawal system from savings account:



Data dictionary :- The analysis model consist of representations of :

ⓐ Data objects ⓑ Function ⓒ Control

The data dictionary is kind of a grammer that describes the contents of objects in the process of structured analysis. It is used as a part of CASE (Computer Aided Software Engineering) tool. They are mainly used to describe the data objects and their attributes with their full description.

— (Read from text book) —

— (later part not imp) —

Control flow model :- The major class of applications having following attributes needs control flow modelling

① The applications those produce control information rather than reports or displays

② The applications those are driven by the events rather than data

~~③ The application those produce control info~~

③ The application those process information in specific time

④ The control item or event is implemented as Boolean value for example, true or false, on or off, 1 or 0.

Scenario based Modelling:-

Analysis modelling with UML begins with the creation of scenarios.

In scenario based modelling the system is represented in user's point of view. It's elements are:

1. Use case diagrams
2. Activity diagrams
3. Swim lane diagrams

(1) Use case and its purpose (Developing use case):

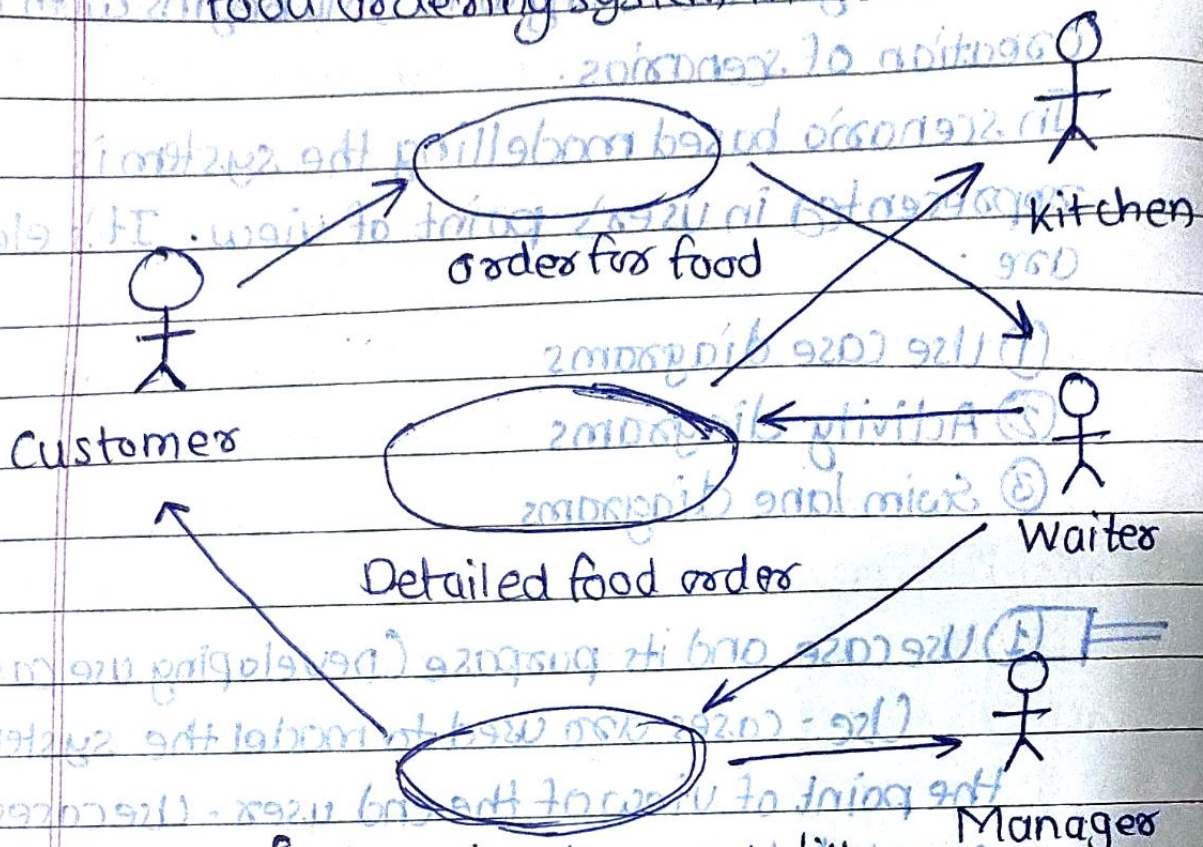
Use-cases are used to model the system from the point of view of the end user. Use cases are created during requirements elicitation. Use cases helps in understanding the exact product requirements in the context of its operation clearly.

(2) Use case diagrams:-

To start developing a set of use cases, the functions or the activities performed by actor are listed. The list of use-cases may be obtained from one of the following resources:

* From customer communication
* End user communication
* By evaluating activity diagrams.

Example: To develop use case diagram for food ordering system in hotel.



Behavioural model:- The behavioural model shows how software will give response when some external event occurs. To create the behavioural model, the analyst must perform following steps:

1. Evaluate all use-cases to fully understand the sequence of interaction within the system.
2. Identify the events and understand how these events relate the specific classes
3. Create the sequence for use cases.
4. Build state diagram for the system
5. Review the model to check its accuracy and consistency.

* State transition diagram :- The state transition diagrams are used to show the behaviour of the system. They show the transition from one state to another on occurance of some external event.

Design Modelling :- Software design plays an important role in software development process. Design is very basis of further development of product. Good design ensures sustainable software product which is highly reliable and standard. Design modelling takes help of different diagrams and creates complete picture of the software before we actually develop it.

Design process :- Throughout the design process, the quality of the evolving design is assessed with a series of formal technical reviews or design walkthrough.

Three characteristics that serve as a guide for the evaluation of a good design are :

① The design must implement all the explicit requirements contained in the analysis model, and it must include all of the implicit requirements desired by the customer.

② The design must be a readable. It should be an understandable guide for those who generate or produce the code and for those who does testing and does maintainence of the software.

③ From the implementation perspective, the design need to give complete idea or picture of software addressing the data, functional, and behavioural domains.

All of these characteristics are actually goals of the design process.

Concept of software design:-

Fundamental collection or set of software design concepts have started evolving before last four to five decades. These concepts are well tested and thus form the basis for the software design. These concepts helps software engineers to ask following questions:

* What criteria will be helpful to partition software into individual components?
* From conceptual representation of software, how to seperate function or data structure.

Design Quality Guidelines:- In order to evalueate the qudlity of a design representation, we must establish technical criteria for good design. We present the following guidelines:

① Software design needs to be modular, Because of modular design, software can be partitioned logically into element.

② A design should depict different representations of components, interfaces, architecture and data.

③ A design has to have appropriate classes and data structure to be implemented. It should be sourced from recognizable data patterns.

④ A design should be done in iterating manner on the basis of information that is received during software requirement analysis.

⑤ A design should be depicted using meaningful and efficient notation for the purpose of knowing and communicating its meaning easily.

Design concepts:-

① Abstraction :- At the topmost level of abstraction, a solution is present in broad terms with the help of the problem environment.
At lower levels of abstraction, a more detailed description of the solution is provided.

② Architecture :- It is overall structure of the software and the methods by which structure allows conceptual integrity for a system.

③ Patterns :- A pattern can be referred as a thing that gives the essence of proven solution to a recurring problem in a particular context.

④ Modularity :- Modularity is represented by software architecture and design patterns. The meaning is, software is divided into uniquely named and addressable components or elements. Sometimes they are called as modules.

⑤ Information Hiding :- ~~Hiding~~ The modules should be specified and designed so that information contained within a module is inaccesible to other modules that have no need for such information.

⑥ Functional independence :- Functional independence can be accomplished by designing the software in such a way that each module addresses a particular sub-function of requirements.

⑦ Refinement :- Top-down design strategy is stepwise refinement. A program is developed by consecutively refining levels of procedural detail.

⑧ Refactoring :- It is a reorganisation technique that makes design component or code simple without making any changes in its behaviour or function.

The design model :- The design model is the collection of representations of data, architecture, interfaces and components.

① Data design elements

Alike different activities of software engineering data design build a model of data and information that is represented at the top level of abstraction.

Data model is then refined into increasingly more implementation specific representation that can be processed by the computer based system.

② Architectural design elements :-

The architectural model is derived from three sources

① Information about the application domain for the software to be built.

② Specific analysis model elements like analysis classes and data flow diagrams or relationships among them and collaboration of current problem.

③ Availability of architectural styles

③ Interface Design Elements :-

There are three important elements of interface design :-

① The user interface (UI).

② External interfaces to other systems, devices, networks, or other producer or consumer of information and,

③ Internal interfaces between various design components.

④ Component level design elements:- The internal detailed information of every software component is completely by component level design.

To accomplish this, the component level design specifies algorithmic details and data structures for all data object that are at local level for all processing that takes place in component and an interface that permits access to all component operations or behaviours.

⑤ Deployment Level Design elements:-

Deployment-level design elements depict how software functionality and subsystems will be placed within the physical computing environment which will support the software.