Unit 4: Data Flow Analysis

AGENDA

- Data Flow Analysis,
- Developing Logical Model Of the System Using Data Flow Diagram,
- Data Dictionary,
- HIPO Chart, Visual Table of Content,
- System Flow Chart,
- Data Structure Diagram.

Data Flow Analysis

- **Data-flow analysis** is a technique used by software engineers to analyze how data is input into the system, stored, processed, and outputted.
- This helps designers to understand how information flows through the system and to identify any potential bottlenecks, inefficiencies, and security vulnerabilities or areas where data may be lost or corrupted.
- The data gained from this process may also be used for optimizing or debugging the software.
- Data flow analysis involves identifying the different data sources, data processes, data stores, and data destinations within the system.

Data Flow Analysis

- It also involves documenting the flow of data between these components and analyzing how data is transformed as it moves through the system.
- This information can then be used to optimize the system design, improve data processing efficiency, and enhance data security.
- Data flow analysis can be done using a variety of techniques, such as **data flow diagrams**, **entity-relationship diagrams**, and **data dictionaries**.
- These tools help designers to visualize the flow of data and to identify relationships between different parts of the system.

Data Flow Analysis

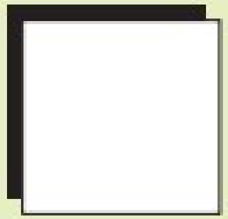
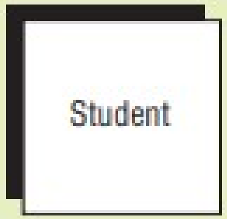
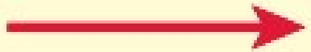
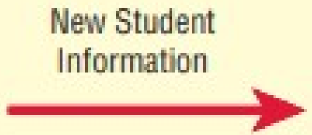
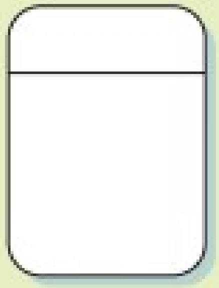
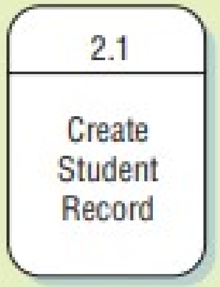
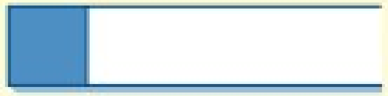
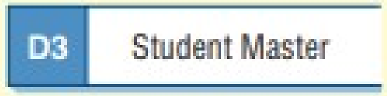
- Overall, **data flow analysis** is an important part of system analysis and design, as it helps to ensure that the system is designed in a way that allows for efficient and accurate flow of data.
- By understanding how data moves through the system, designers can create a system that meets the needs of its users and stakeholders.

Developing Logical Model Of the System Using Data Flow Diagram

- A Data Flow Diagram (DFD) is a graphical representation of the flow of data within a system.
- It shows the processes that transform **input** data into **output** data and the data **stores** and **flows** that connect them.
- A Data Flow Diagram (DFD) is a powerful tool used in system analysis and design to model the processes, data stores, data sources, and data destinations within a system, as well as the flow of data between them.
- DFDs are typically hierarchical, with multiple levels of diagrams representing increasing levels of detail.

Developing Logical Model Of the System Using Data Flow Diagram

- The highest-level DFD, known as a **context diagram**, provides an overview of the entire system and shows the interactions between the system and its external entities.
- Lower-level DFDs provide more detailed views of specific processes within the system.
- DFD helps in understanding the system's data flow, identifying potential issues or bottlenecks, and designing more efficient and secure systems.
- DFDs are a valuable tool for communication between stakeholders, as they provide a clear and intuitive representation of the system's data flow.
- In a DFD, four symbols are used in drawing data flow diagrams.

Symbol	Meaning	Example
	Entity	
	Data Flow	
	Process	
	Data Store	

Developing Logical Model Of the System Using Data Flow Diagram

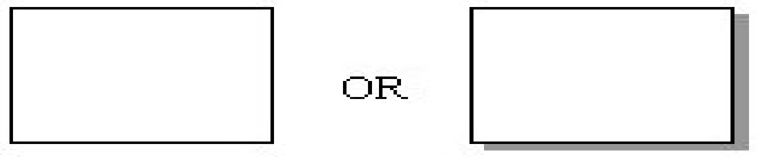
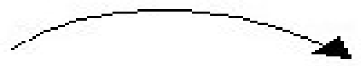
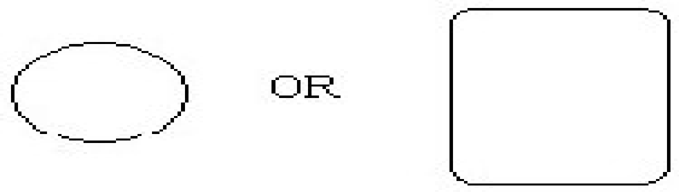
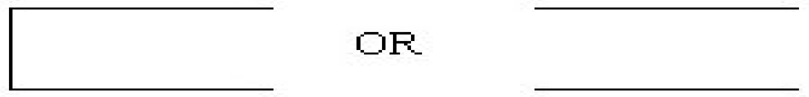
- **Processes** are represented as rectangles, **data stores** are represented as parallel lines, **data sources** and **destinations** are represented as ovals, and **data flows** are represented as arrows.
- The direction of the arrows indicates the flow of data between the different components of the system.
- A **circle/bubble** is used to depict a process. A **process** describes what a system does. Both inputs and outputs to a process are data flows.
- Processes may store or retrieve data from a **data store**. A data store is depicted by two parallel lines. write in or read from the store.

Symbols Used in DFDs

- If an arrows points to the store, it indicates operation of writing in the store and if it points away from the store it indicates operation of reading from the store.
- If both arrows are there, it indicates that one may write in or read from the store.
- Data flows are represented by a line with an arrow. The arrow shows the direction of flow of data.
- The name of the data appears below the line. In the above diagram the data flows are labelled : Stores demand slip, Delivery slip and Issue advice.

SYMBOLS USED IN DFDs

- External entities are represented by rectangles, and are outside the system, such as vendors or customers with whom the system interacts.
- The designer has no control over them. They either supply or consume data.
- Entities supplying data are known as sources and those that consume data are called sinks.

Symbol Name	**Symbol**	**Symbol Meaning**	**Comments**
Square		Source or destination of data	May be one customer or a number of customers with customer orders
Arrow		Data flow	May be physically contained in a purchase order, invoice, etc.
Circle		Process that transforms that data flow	May be an accountant calculating discounts and preparing invoices
Open Rectangle		Data store	Can be a file, magnetic tape, a database on disk, etc.

SYMBOLS USED IN DFDs

- Data flow can take place between processes, from:
  - (i) a data store to a process,
  - (ii) a process to a data stores,
  - (iii) an external entity to a process, or
  - (iv) from a process to an external entity.
- Data flows cannot take place between two data stores as they are passive or between two external entities.
- Besides the symbols we have used, other variations exist as there is no universal standard.
- In several books a process is depicted by a rectangle with round edges and a data store by a long open rectangle as shown below.

Developing DFDs

- Data flow diagrams can and should be drawn systematically.
- First, the systems analyst needs to conceptualize data flows from a top-down perspective.
- To begin a data flow diagram, collapse the organization’s system narrative (or story) into a list with the four categories of external entity, data flow, process, and data store.
- This list in turn helps determine the boundaries of the system you will be describing.
- Once a basic list of data elements has been compiled, begin drawing a context diagram. Here are a few basic rules to follow:

Developing DFDs

1. The data flow diagram must have at least one process, and must not have any freestanding objects or objects connected to themselves.
2. A process must receive at least one data flow coming into the process and create at least one data flow leaving from the process.
3. A data store should be connected to at least one process.
4. External entities should not be connected to each other. Although they communicate independently, that communication is not part of the system we design using DFDs.
5. DFDs must be developed top down with lower levels giving more details.

Developing DFDs

6. Arrows should not cross each other. Squares, circles, and files must bear names.
7. Decomposed data flows must be balanced (all data flows on the decomposed diagram must reflect flows in the original diagram).
8. No two data flow, squares, or circles can have the same name. Draw all data flows around the outside of the diagram.
9. Choose meaningful names for data flows, processes, and data stores. For instance, a name such as “data A” is not meaningful, whereas a name such as “requisition for items” is meaningful.
10. Use strong verbs followed by nouns. Control information such as record counts, passwords, and validation requirements are not pertinent to a data-flow diagram.

How Detailed a DFD should be?

- The DFD is designed to aid communication. If it contains dozens of processes and data stores, it gets too unwieldy.
- The rule of thumb is to explode the DFD to a functional level, so that the next sublevel does not exceed 10 processes.
- Beyond that, it is best to take each function separately and expand it to show the explosion of the single process.
- If a user wants to know what happens within a given process, then the detailed explosion of that process may be shown.
- A DFD typically shows the minimum contents of data stores. Each data store should contain all the data elements that flow in and out.

How Detailed a DFD should be?

- All discrepancies, missing interfaces, redundancies, and the like are then accounted for- often through interviews.
- The DFD methodology is quite effective, especially when the required design is unclear and the user and the analyst need a notational language for communication.
- The DFD is easy to understand after a brief orientation.

How Detailed a DFD should be?

- The two types of data flows are physical DFD and logical DFD.
- **Physical DFD:**
- Physical data flow diagram shows how the data flow is actually implemented in the system. Physical DFD is more specific and close to implementation.
- A physical DFD is an implementation dependent view of the current system showing what functions are performed.
- A physical diagram provides details about hardware, software, files and people involved in the implementation of the system.
- Physical characteristics include names of the people, names of the departments, names of files, names of hardware, locations, names of procedures, form names, document names, etc.

How Detailed a DFD should be?

- **Logical DFD**: is an implementation independent view of a system that focuses only on the flow of data between different processes or activities.
- Logical diagrams show how the business operates. They do not show how the system can be implemented.
- They explain the events of the system and the data required by each event of the system.

How Detailed a DFD should be?

- **Physical DFD differs from the logical DFD in the following ways:**
  - Physical DFD is implementation dependent whereas logical DFD is implementation independent.
  - Physical diagrams in the physical DFDs provide low level details, such as hardware and software requirements of a system whereas logical diagrams in the logical DFDs explain only the events involved in the system and the data required to implement each event of the system.

How Detailed a DFD should be?

- Context diagrams — context diagram DFDs are diagrams that present an overview of the system and its interaction with the rest of the “world”.
- Level 1 data-flow diagrams — Level 1 DFDs present a more detailed view of the system than context diagrams, by showing the main sub-processes and stores of data that make up the system as a whole.
- Level 2 (and lower) data-flow diagrams — a major advantage of the data-flow modelling technique is that, through a technique called “levelling”, the detailed complexity of real world systems can be managed and modelled in a hierarchy of abstractions.
- Certain elements of any dataflow diagram can be decomposed (“exploded”) into a more detailed model a level lower in the hierarchy.

Context Level Diagram

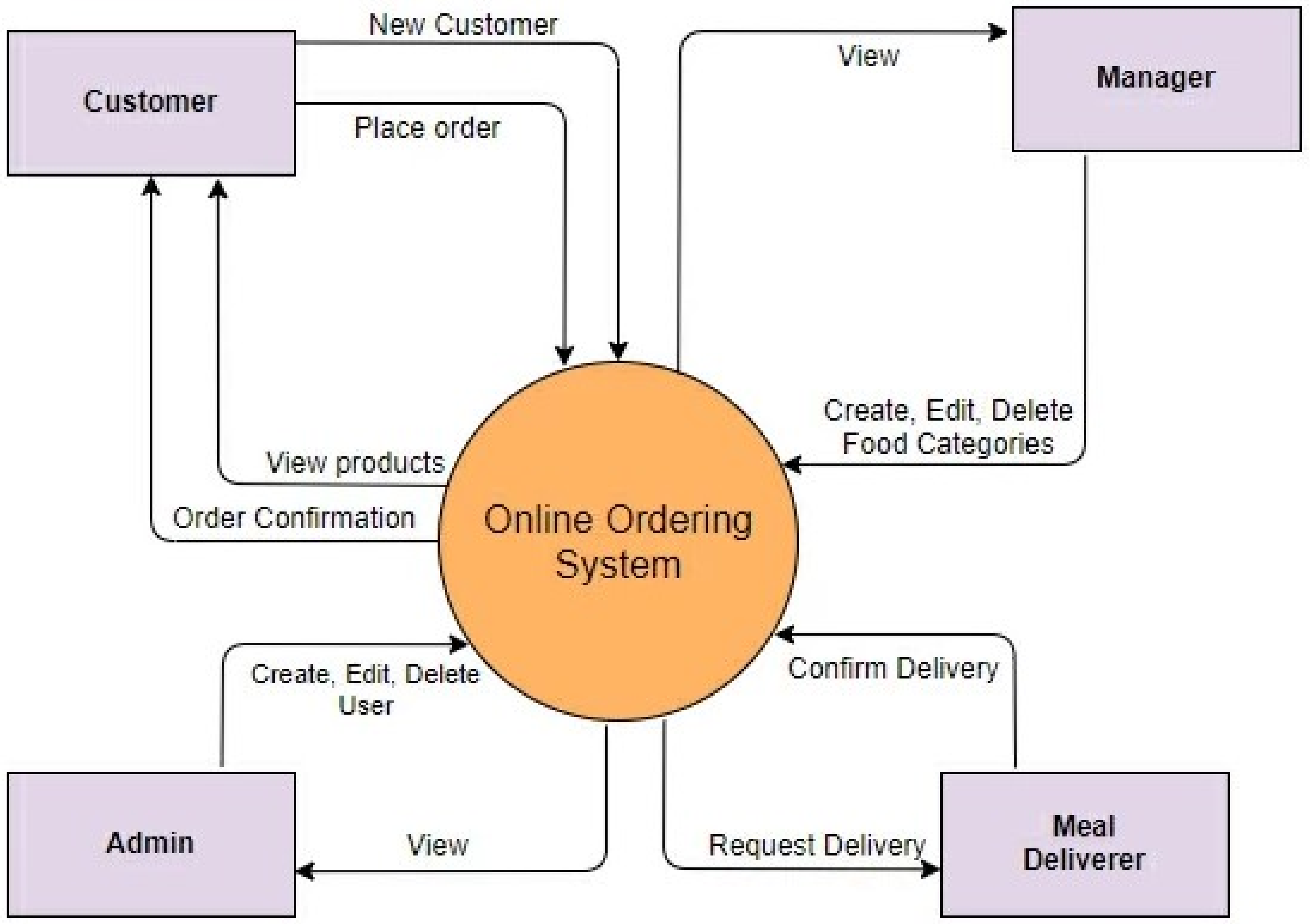
- The **context diagram** is used to establish the context and boundaries of the system to be modelled: <u>which things are inside and outside of the system being modelled, and what is the relationship of the system with these external entities</u>.
- Anything that is not inside the process of the context diagram will not be part of the system study.
- Context diagram helps in understanding the general characteristics of the system under study.
- It contains a single process and defines the whole system that will be studied.
- The inputs and outputs specified at this level remain constant for the other lower levels as well.

Context Level Diagram

- **Context diagrams**
- The context diagram is used to establish the context and boundaries of the system to be modelled: which things are inside and outside of the system being modelled, and what is the relationship of the system with these external entities.
- A context diagram, sometimes called a level 0 data-flow diagram, is drawn in order to define and clarify the boundaries of the software system.
- It identifies the flows of information between the system and external entities. The entire software system is shown as a single process

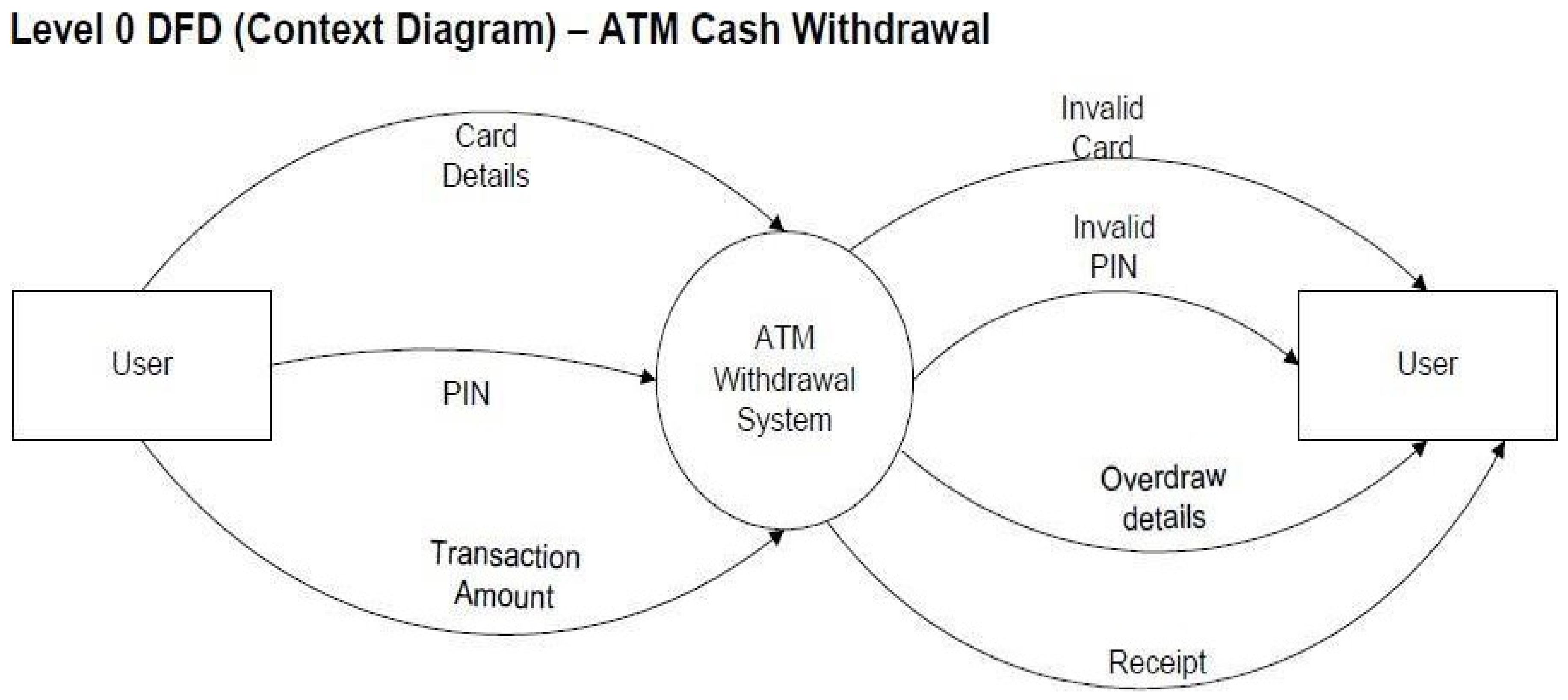
Context Level Diagram

- Context Level DFD for **Online Ordering System**



Context Level Diagram

- **Context diagrams**



Level 1 DFD

- Level 1 DFD provides more detail than the context diagrams. By dividing context.
- diagram processes into sub-processes, the systems analyst begins to fill in the details about the flow of data within the system. Each sub-process in the Level 1
- DFD is numbered with an integer. Generally, this number starts from the upper left corner and proceeds towards the lower right corner of the Level 1 DFD.
- Following figure shows the Level 1 DFD for customer order and credit verification system.
- Each process in the Level 1 DFD may, in turn, be divided to create a more detailed child diagram.

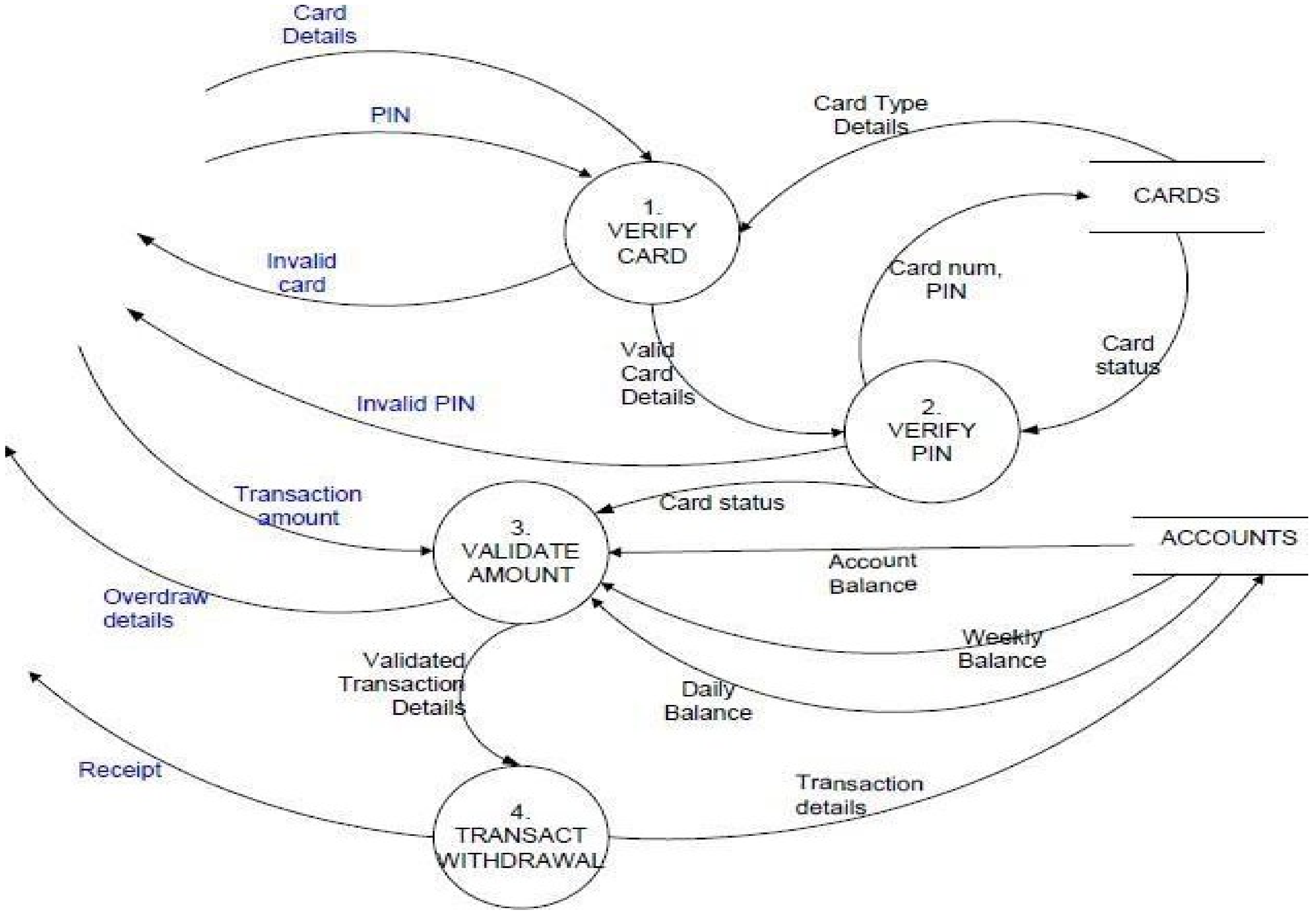
Level 1 DFD

- **Level 1 data-flow diagrams**
- As described previously, context diagrams (level 0 DFDs) are diagrams where the whole system is represented as a single process.
- A level 1 DFD notates each of the main sub-processes that together form the complete system. We can think of a level 1 DFD as an “exploded view” of the context diagram.

Level 1 DFD

- The process of level 1 DFD that is divided is called the 'parent process' and the resultant diagram is called the 'child diagram.'
- The primary rule while creating child diagrams is that the child diagram cannot receive input or produce output that is not produced or received by its parent process.
- That means any data flowing in or out of the parent process should be shown flowing into or out of the child diagram.
- The processes in the child diagram are numbered by using the parent process number, a decimal point and a unique number assigned for each child process of the child diagram.
- Following figure shows a first Level DFD elaborating customer order processing and shipment system.

Level-1 for ATM cash withdrawal



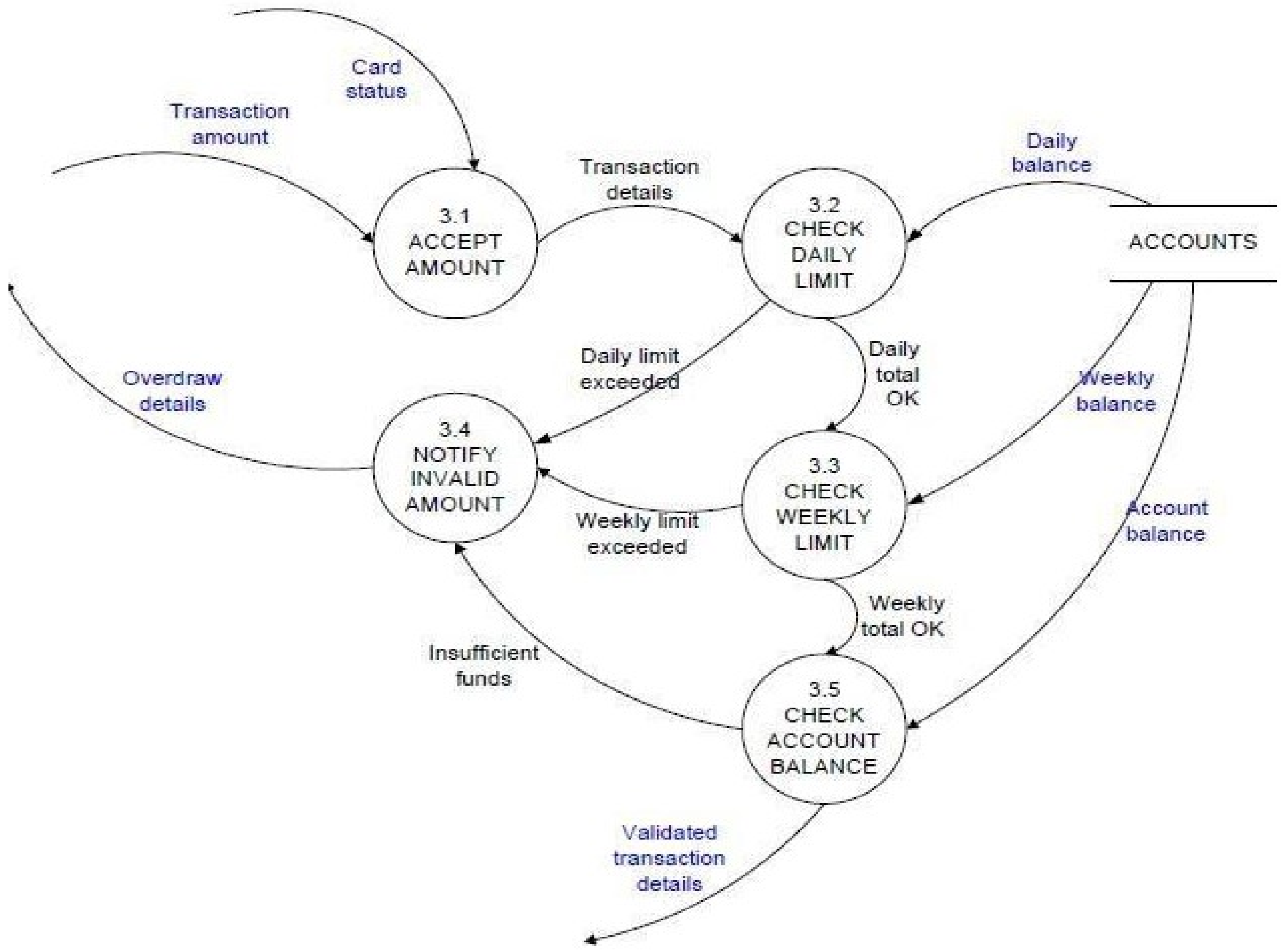
Decomposing diagrams into level 2

- Decomposing diagrams into level 2 and lower hierarchical levels
- We have already seen how a level 0 context diagram can be decomposed (exploded) into a level 1 DFD.
- In DFD modelling terms we talk of the context diagram as the “parent” and the level 1 diagram as the “child”.
- This same process can be applied to each process appearing within a level 1 DFD.
- A DFD that represents a decomposed level 1 DFD process is called a level 2 DFD.
- There can be a level 2 DFD for each process that appears in the level 1 DFD.

Decomposing diagrams into level 2

- Decomposition is applied to each process on the level 1 diagram for which there is enough detail hidden within the process.
- Each process on the level 2 diagrams also needs to be checked for possible decomposition, and so on.
- Each process on the level 1 diagram is investigated in more detail, to give a greater understanding of the activities and data-flows.

Decomposing diagrams into level 2



Level 2 DFD – ATM Cash Withdrawal (Validate Amount)

Numbering in DFDs

- Numbering in a levelled set of diagrams is important, as the numbers help you to find your way around the levels.
- It is easily described by example.
- Suppose Receive Order is the process numbered 3 on the level one diagram (remember, numbers do not indicate any order, they are simply labels) and this is expanded to a level two diagram.
- The process numbers on the level two diagram will be 3.1, 3.2, 3.3 and so on. Suppose now that process 3.4 on the level two diagram is Register New Customer and needs further expansion to a level three diagram.
- The process numbers on this diagram will be 3.4.1, 3.4.2, 3.4.3 and so on.
- The rule used here is this: if X is the number of the process you wish to expand, then the numbers on the next level are X.1, X.2, X.3...

Data Dictionaries

- In our data flow diagrams, we give names to data flows, processes and data stores.
- Although the names are descriptive of the data, they do not give details.
- So following the DFD our interest is to build some structured pace to keep details of the contents of data flows, processes and data store.
- A data dictionary is a structured repository of data. It is a set of rigorous definitions of all DFD data elements and data structure encountered during the analysis and design of a new system.

Data Dictionaries

- Data elements can describe files, data flows, or process.
- For example, suppose you want to print the vendor's name and address at the bottom of a cheque.
- The data dictionary might define vendor's name and address as follows:

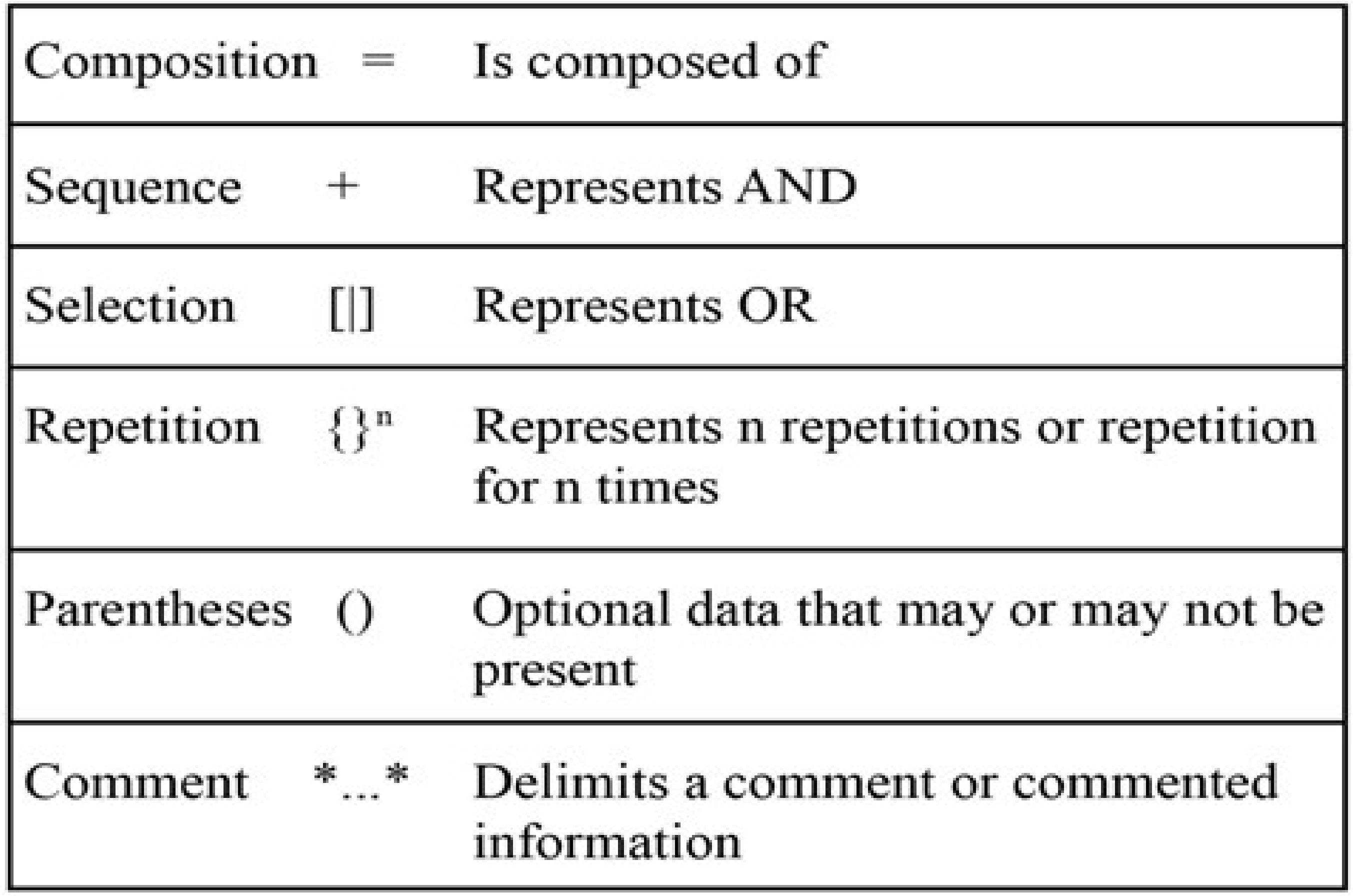
  Vendor name and address = Vendor name + Street + City + State + Pin + Phone + Fax + e-mail
- The definition becomes a part of the data dictionary that ultimately will list all key terms used to describe various data flows and files.

Data Dictionaries

- A data dictionary provides details about things like:
  - Meanings of objects
  - Relationship between the data objects
  - Views
  - Origin of the data elements
  - Format of the data elements
- The data dictionary concept is used in various domains like DBMS (database management systems), software engineering, etc.
- In DBMS, for example, in oracle the content of the data dictionary is metadata about structures and blueprint of tables:
  - Name of the tables
  - Definition of each column of every table
  - Constraints like key relationships
  - Ownership and privileges granted on the tables and other database objects
  - Statistics like last accessed or last updated
  - Indexes used by the optimizer

Data Dictionaries

- **Major Symbols**
- A data dictionary uses the following major symbols:



Composition	=	Is composed of
Sequence	+	Represents AND
Selection	$[\vert]$	Represents OR
Repetition	$\{\}^n$	Represents n repetitions or repetition for n times
Parentheses	()	Optional data that may or may not be present
Comment	*...*	Delimits a comment or commented information

Data Dictionaries

- **Four Rules** govern the construction of data dictionary entries;

1. Words should be defined to stand for what they mean and not the variable names by which they may be described in the program; use CLIENT_NAME not ABCPQ or CODE06. Capitalization of words helps them to stand out and may be of assistance.
2. Each word must be unique; we cannot have two definitions of the same client name.
3. Aliases, or synonyms, are allowed when two or more entries show the same meaning; a vendor number may also be called a customer number. However, aliases should be used only when absolutely necessary.
4. Self-defining words should not be decomposed. We can even decompose a dictionary definition. For instance, we might write:
   Vendor name = Company name, Individual's name

Data Dictionaries

- Which we might further decompose to:

Company name	=	*(Contact) + Business name*
Individual's name	=	*Last name + First name + (Middle initial)*

- After defining a term, say VENDOR NUMBER, we list any aliases or synonyms, describe the term verbally, specify its length and data type, and list the data stores where the terms is found.
- Some terms may have no aliases, may be found in many files, or may be limited to specific values. Some self-defining or obvious words and terms may not require inclusion in the data dictionary.
- For example, we all know what a PIN code and a middle initial are. Data dictionaries seldom include information such as passwords users must enter to gain access to sensitive data.
- Rather, data dictionaries offer definitions of words and terms relevant to a system, not statistical facts about the system.

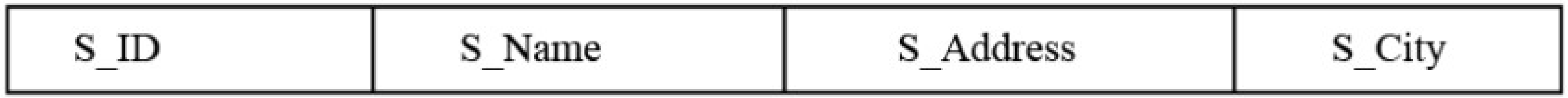
Data Dictionaries

- Data dictionaries allowed analysts to define precisely what they mean by a particular file, data flow, or process.
- Some commercial software packages, usually called Data Dictionary systems (or DDS), help analyst maintain their dictionaries with the help of the computer.
- These systems keep track of each term, its definition, which systems or programs use the term, aliases, the number of times a particular term is used and the size of the term can be tied to commercial data managers.

Data Dictionaries

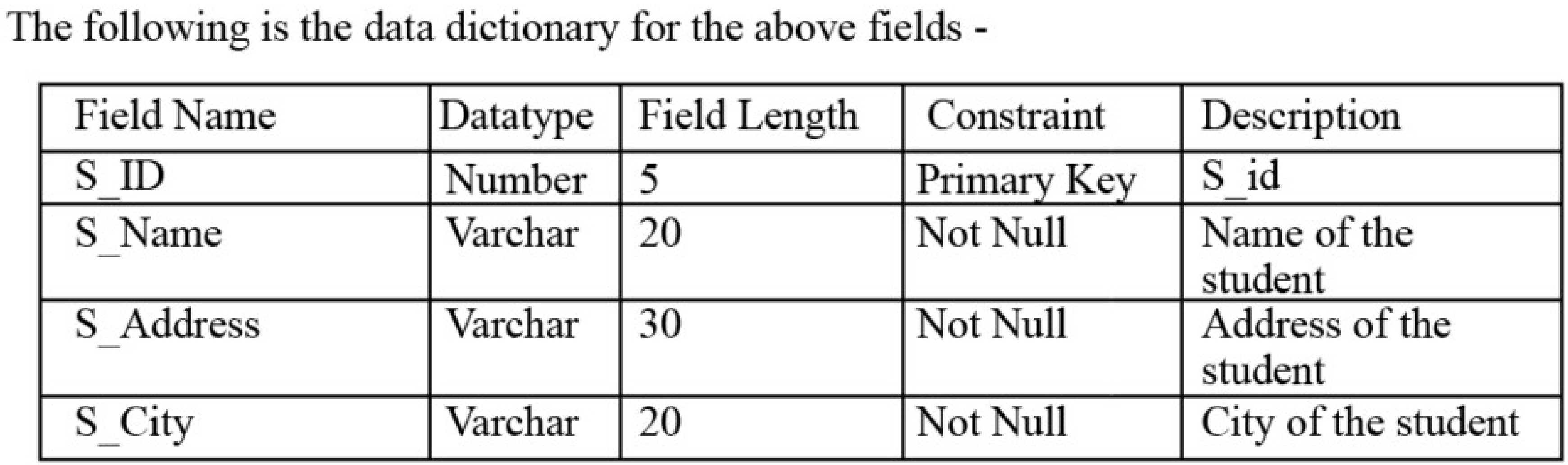
- **Example**

Fields of relation Student



S_ID	S_Name	S_Address	S_City

The Data Dictionary for the above fields is given below.



The following is the data dictionary for the above fields -

Field Name	Datatype	Field Length	Constraint	Description
S_ID	Number	5	Primary Key	S_id
S_Name	Varchar	20	Not Null	Name of the student
S_Address	Varchar	30	Not Null	Address of the student
S_City	Varchar	20	Not Null	City of the student

Data dictionary

Data Dictionaries

- **Data Dictionary Types**
- Figure 4.5 illustrates the different types of data dictionaries and the functions of each address.
- There are two kinds of data dictionaries;
  - (i) Integrated and
  - (ii) Stand-alone.
- **The integrated dictionary** is related to one database management system (DBMS). To the extent the organization data is under this DBMS it is global or organization wide.
- However, very few enterprises have all their data eggs in one basket, so the dictionary documentation (metadata) can be considered as local and fragmented.

Data Dictionaries

- **The stand-alone dictionary** is not tied to any DBMS, although it may have special advantages for one DBMS, such as the IBM DB-DC Data Dictionary, which has special features related to the IBM IMS DBMS but is still a stand-alone variety of dictionary.
- **Data Dictionary Functions**
- Both these types of dictionaries can be identified by functions as either **passive and active.**
- Viewed either way, by type or function, the differences are striking. Passive and active dictionaries differ functionally as follows:

Data Dictionaries

- **Active data dictionary (integrated)**
- If the fields of the database change in between the running system, then the update should be reflected in the data dictionary also.
- Active data dictionary updates automatically through background running tasks.
- Because of that, we do not use any external software for managing the data dictionary.
- **Passive data dictionary**
- Managed by the users and is modified manually when the database structure change.
- The file is stored separately from the database.
- The passive data dictionary is not a part of a database management system. This type of data dictionary is not directly associated with any database or server.

HIPO Charts

- The HIPO (Hierarchy Process Input Output) chart is a tool used to analyse a problem and visualize a solution using top-down design approach.
- Hierarchy Process Input Output (HIPO) diagrams were developed by International Business Machines (IBM) in the 1970s as a design Notation for representing system design.
- These diagrams further became an excellent input for detailed program design.
- In addition, HIPO diagrams have been used by systems analysts to present a high level view of the functions performed by a system and decomposition of the functions into sub-functions and so on.

HIPO Charts

- The Input represents the resources, materials, or information needed to start the process, the Process is the steps or activities involved in transforming the inputs into outputs, and the Output is the end result or product of the process.
- The HIPO chart can help organizations improve efficiency, identify bottlenecks, and streamline processes.
- Thus, it can be said that the HIPO diagram is as useful to analysts as the program design language is to programmers.

HIPO Charts

- A HIPO diagram or chart generally consists of:
  - A set of overview diagrams and a set of detailed diagrams.
  - **A Visual Table Of Contents (VTOC)** consists of tree or graph structured directory, summary of contents in each overview diagram and a legend of symbol definitions.
  - **Input Process Output (IPO)**
- Note that in some user environments HIPO diagrams can be used as a modelling tool as they are similar to the organization chart that describes the hierarchy of managers, sub-managers, and so on.

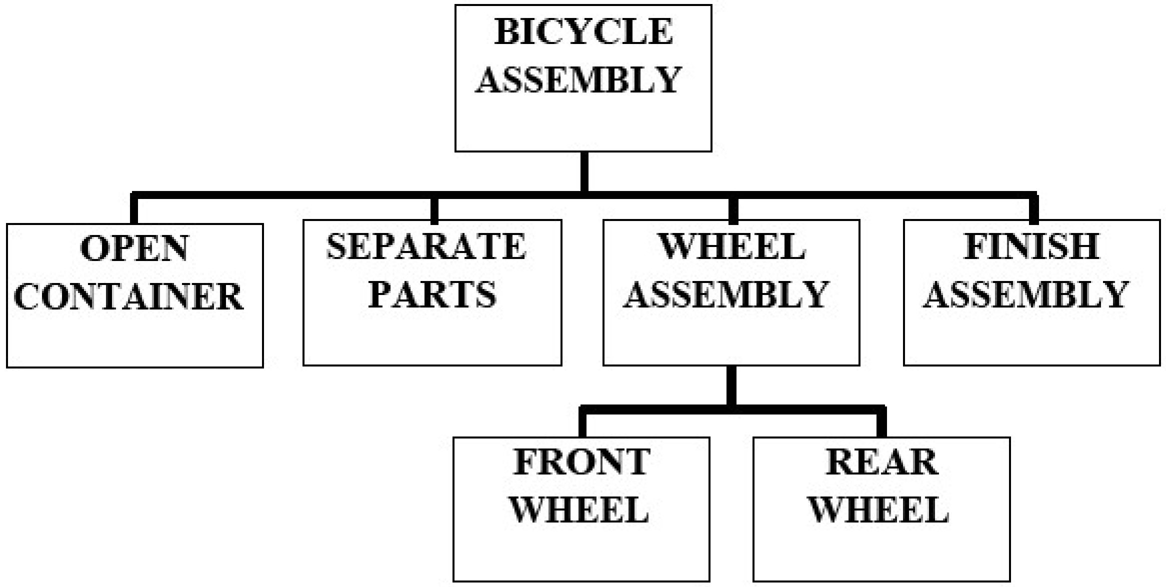
HIPO Charts

- Together these diagrams assist in designing programs and their functions.
- **VTOC (Visual Table of Content) diagram**
- Following the structured approach that begins with generalities and descends to details, VTOC diagram breaks a system down into increasingly detailed levels.
- Therefore, the name of the system appears at the top of the VTOC, the names of the major functions within the system lie on the second level and even smaller sub-functions lie on the third and succeeding levels.
- When used to diagram a program, the VTOC arranges the program modules in order of priority, and it reads from the top down and from left to right.

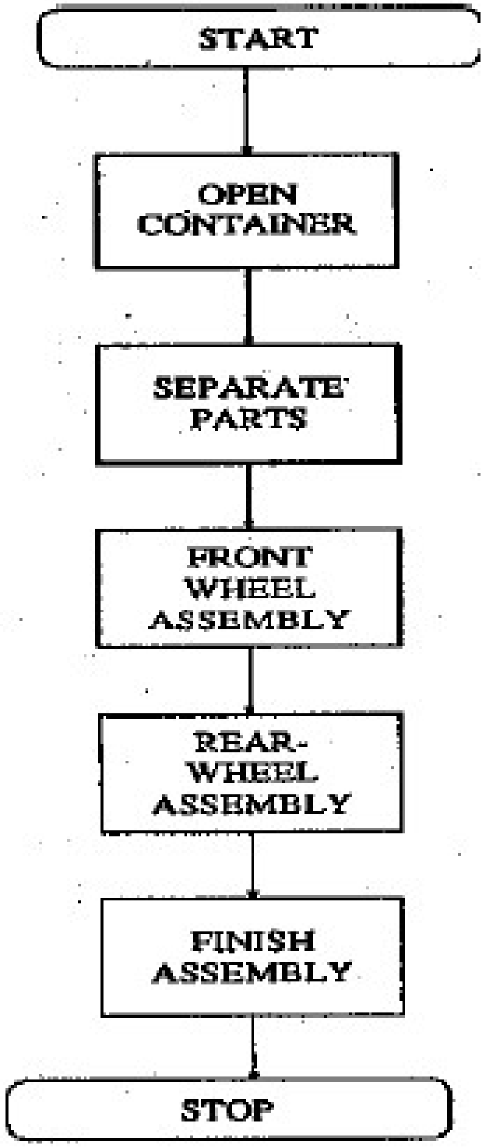
HIPO Charts

- Each module of the program appears as a rectangle which contains a brief description of the module's purpose (two to four words, beginning with a verb followed by an object i.e. “compute net pay”).
- The VTOC for the correct assembly of a bicycle might include five major tasks.
    - (i) open the carton
    - (ii) remove the parts (that is separate them)
    - (iii) group similar parts
    - (iv) assemble the wheels
    - (v) finish assembling the bicycle

HIPO Charts



VTOC for the modules to assemble a bicycle



Flowchart of Bicycle Assembly

HIPO Charts

- Both (HIPO and flow chart) indicate hierarchy but the VTOC offers a more complete picture of the overall process.
- When assembling something, no matter how clear the instructions are, it helps to refer to a picture of the finished product.
- Within the VTOC, each task must be performed in the order specified, and each task may involve several subtasks.
- For example, wheel assembly involves the separate subtasks of front-and rear-wheel assembly.
- An IPO chart defines the inputs, processing, and outputs for each module in the program.

HIPO Charts

- We name modules according to their function so the reader can tell exactly the purpose of the module (again using the verb-object format).
- After naming, we choose a number for the system, we give each module a sub or level number, beginning with 1 for the most general or highest level function.
- Therefore, we would identify the overall system as 1.0, the accounts payable check module as 1.1, the numeric vendor list as 1.2 and so on.
- Such a number system clarifies the relationships between modules, and allows anyone reading it easily to locate detailed IPO charts with corresponding numbers.

HIPO Charts

- After assigning level numbers to each module, we can consider whether further decomposition is necessary.

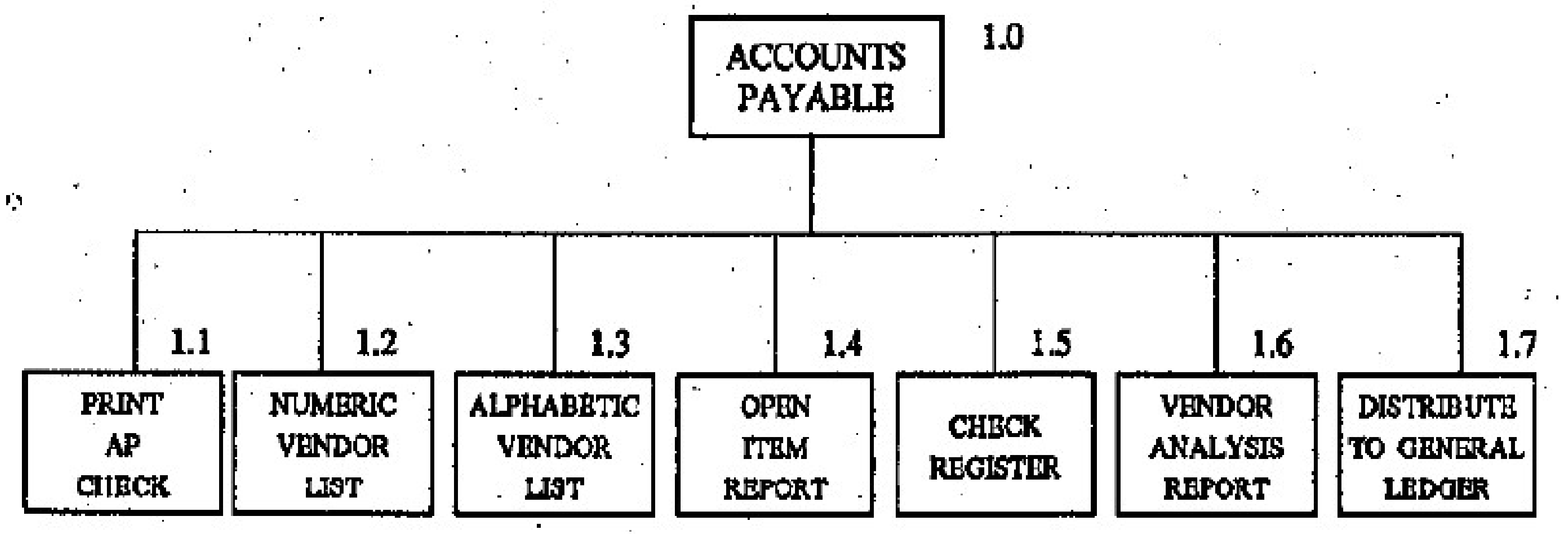
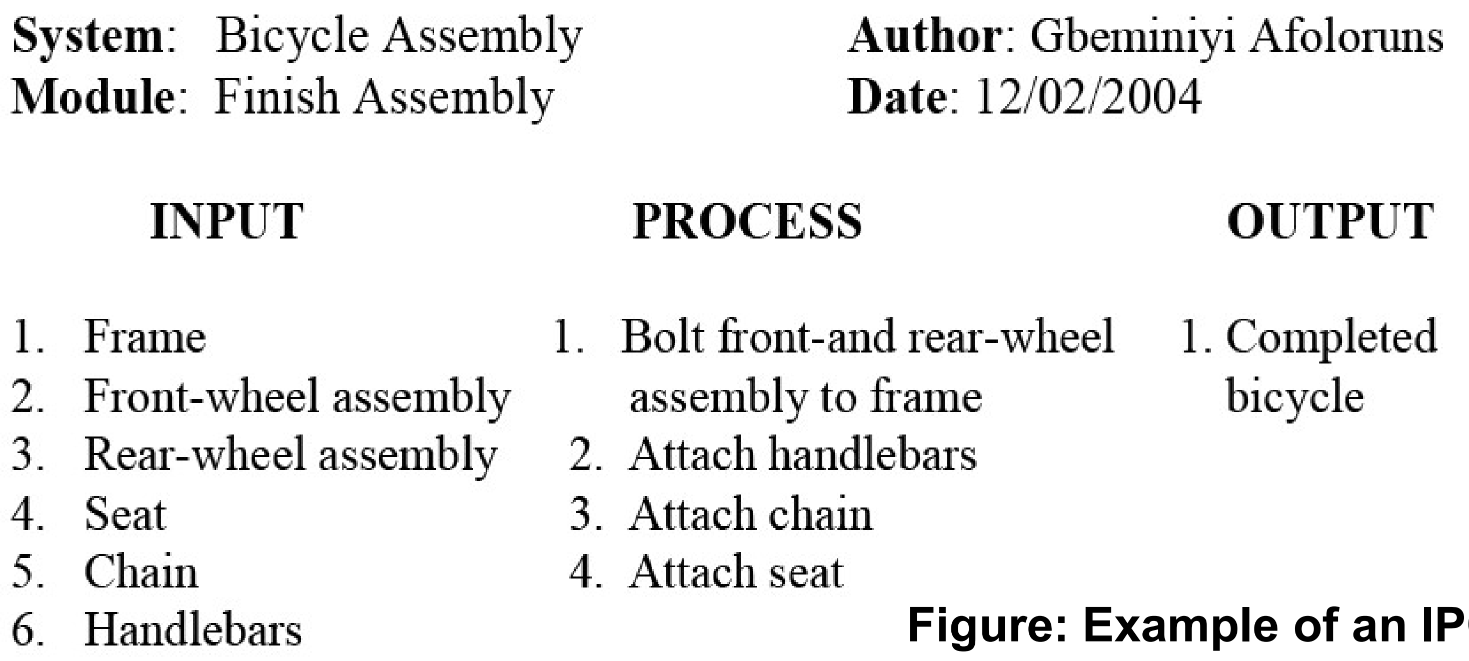


Fig. 3.13: The VTOC for the AP check-printing system with level number.

HIPO Charts

- **IPO(Input Process Output) Chart**
- Let us add the second part of the HIPO diagramming system, the IPO chart.
- Figure below is an IPO for the “finish assembly” module, inputs for which are frame, front-wheel assembly, rear-wheel assembly, seat, handlebars and chain.
- The processing requirements are bolting wheel assemblies to the frame, and attaching handlebars, chain, and seat. The output is the completed bicycle.



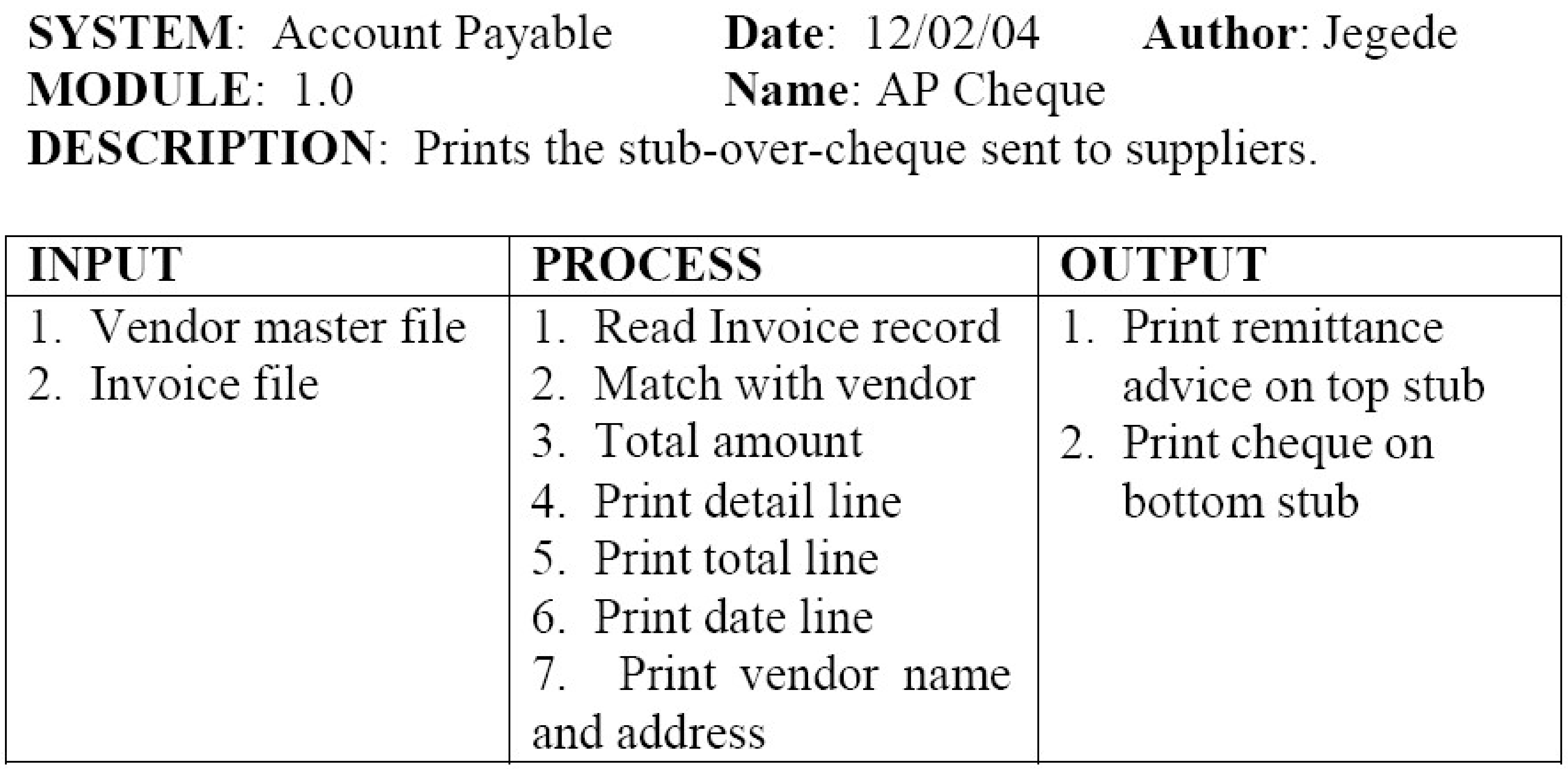
System: Bicycle Assembly **Author**: Gbeminiyi Afoloruns
Module: Finish Assembly **Date**: 12/02/2004

INPUT	PROCESS	OUTPUT
1. Frame 2. Front-wheel assembly 3. Rear-wheel assembly 4. Seat 5. Chain 6. Handlebars	1. Bolt front-and rear-wheel assembly to frame 2. Attach handlebars 3. Attach chain 4. Attach seat	1. Completed bicycle

Figure: Example of an IPO

HIPO Charts

- Whereas the VTOC diagram graphically shows an overview of the system, the IPO charts depict program logic, illustrating the steps required to produce desired purposes.



SYSTEM: Account Payable **Date**: 12/02/04 **Author**: Jegede
MODULE: 1.0 **Name**: AP Cheque
DESCRIPTION: Prints the stub-over-cheque sent to suppliers.

INPUT	PROCESS	OUTPUT
1. Vendor master file 2. Invoice file	1. Read Invoice record 2. Match with vendor 3. Total amount 4. Print detail line 5. Print total line 6. Print date line 7. Print vendor name and address	1. Print remittance advice on top stub 2. Print cheque on bottom stub

Fig. 3.16: The IPO chart detail or program logic

System Flow Charts

- A flow chart is a graph that shows process flow, decisions, and outcomes. Let us take an example of system flowchart of an online air ticket booking system.
- For this, there would be basically two master files which will be maintained for successful system implementation.
- One of the files is maintained for admin side and other one is maintained for travellers.

System Flow Charts

- For the above system design, analysts will go through a basic questionnaire as follows:
  - How many flight numbers will be taken for landing and taking off?
  - Travelling would be taken as round trip or one way and from where (source) to where (destination)?
  - The classes and choices of meals of travellers will be recorded at which stage in system flowchart?
  - How the whole system transactions will run successfully?
  - What type of front end (applications) and back end support (database server) will be implemented to design the system?
  - What total cost will come in the whole transaction during system implementation?

System Flow Charts

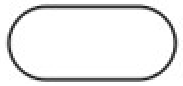
- System flowchart describes the data flow for a data processing system and also provides a logical diagram of how the system operates.
- It represents the flow of documents and the operations performed in data processing system.
- It also reflects the relationship between inputs, processing and outputs.
- A system flowchart, also known as data flowchart, is used to describe the flow of data through a complete data processing system.
- **Different graphic symbols represent the clerical operations involved and the different input, storage and output equipment System Development required.**
- Although the flowchart may indicate the specific programs used but no details are given of how the programs process the data.

System Flow Charts

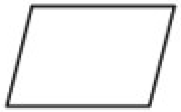
- As you may have already known, a flow chart is a graph that shows process flow, decisions, and outcomes. They are common tools of quality control that are utilized in many fields. There are four basic categories of flowcharts:
- Document flowcharts show you the flow of documents from one business unit to another.
- Data flowcharts let you see the overall data flow in a system.
- Program flowcharts show you a program's control in a system. They are also one of the essential tools in programming. We have already covered it in an article, so check it out on our website!
- **System flowcharts** are the diagram type that shows you the flow of data and how decisions can affect the events surrounding it.

System Flow Charts

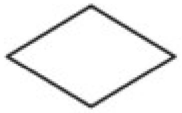
- **System Flow chart symbols**



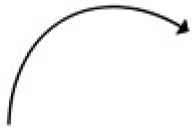
- A system flowchart begins and ends with an oval symbol. This is also called the ‘terminator’ and indicates the start and end of the processes mentioned.



- A rectangle then represents a process or an activity.



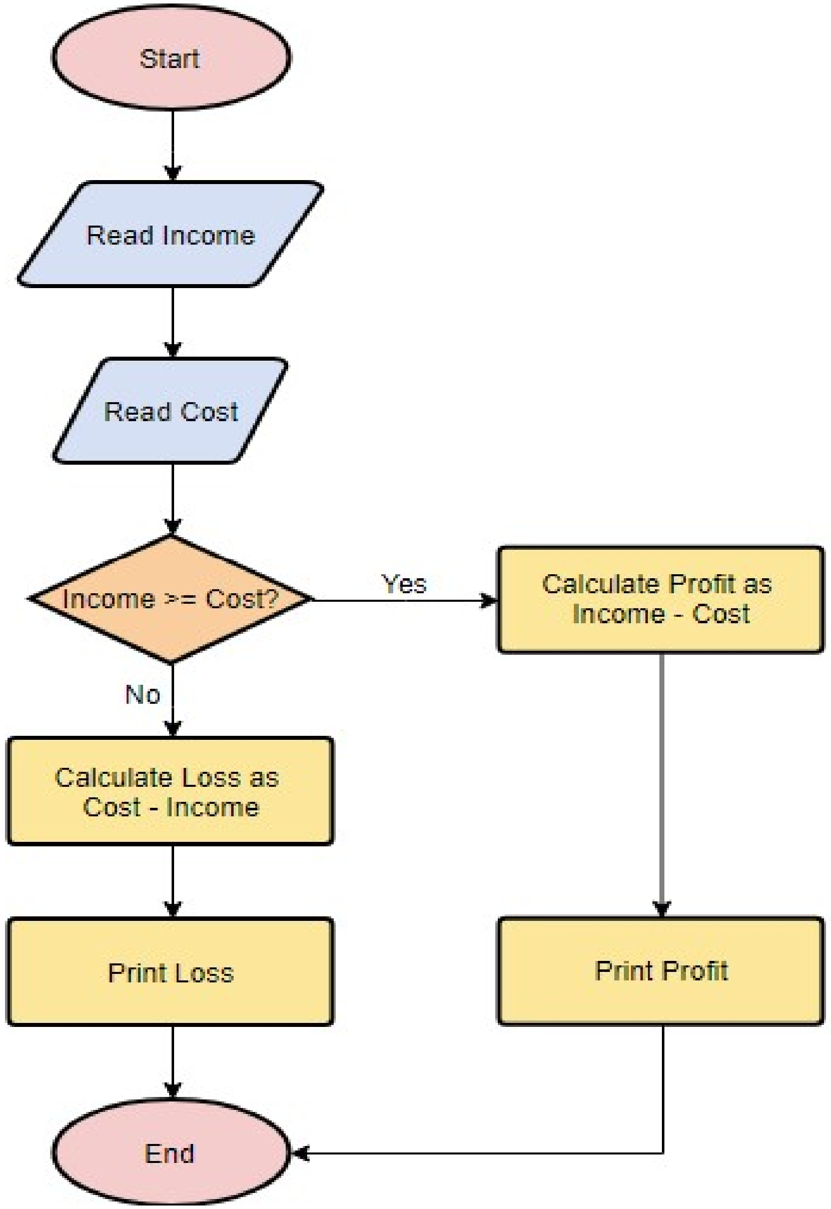
- One of the major symbols used in system flowcharts is the decision boxes represented by a diamond shape. They represent the decisions made, and all arrows coming out of these decision boxes show alternate pathways for data.



- Lastly, all these shapes are connected through arrows showing the direction of the flow and how data moves along the entire system.

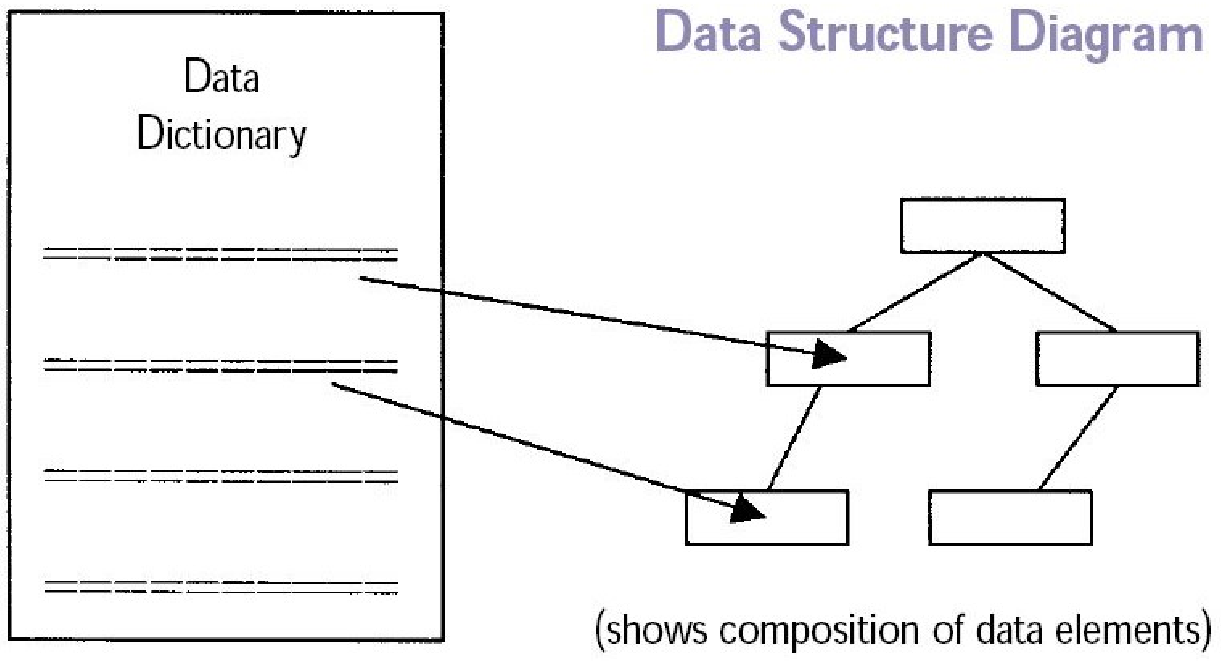
System Flow Charts

- Example: **Profit/Loss Calculating System**
- Profit is one of the important aspects that business runners often care about. It is one of the duties of finance teams.
- Usually, if the production cost is higher than the income, it is a loss. If the income is higher than the cost, it means you have profit.
- The system is simple. It will compare the cost and the income to determine your profit or loss.
- The detailed flow chart of this system will be like this:



Data Structure Diagram

- A **Data Structure Diagram** or **Data Structure Design** is a diagram type that is used to depict the structure of data elements in the data dictionary.
- The data structure diagram is a graphical alternative to the composition specifications within such data dictionary entries.
- IT is a predecessor of the entity–relationship model (E–R model).



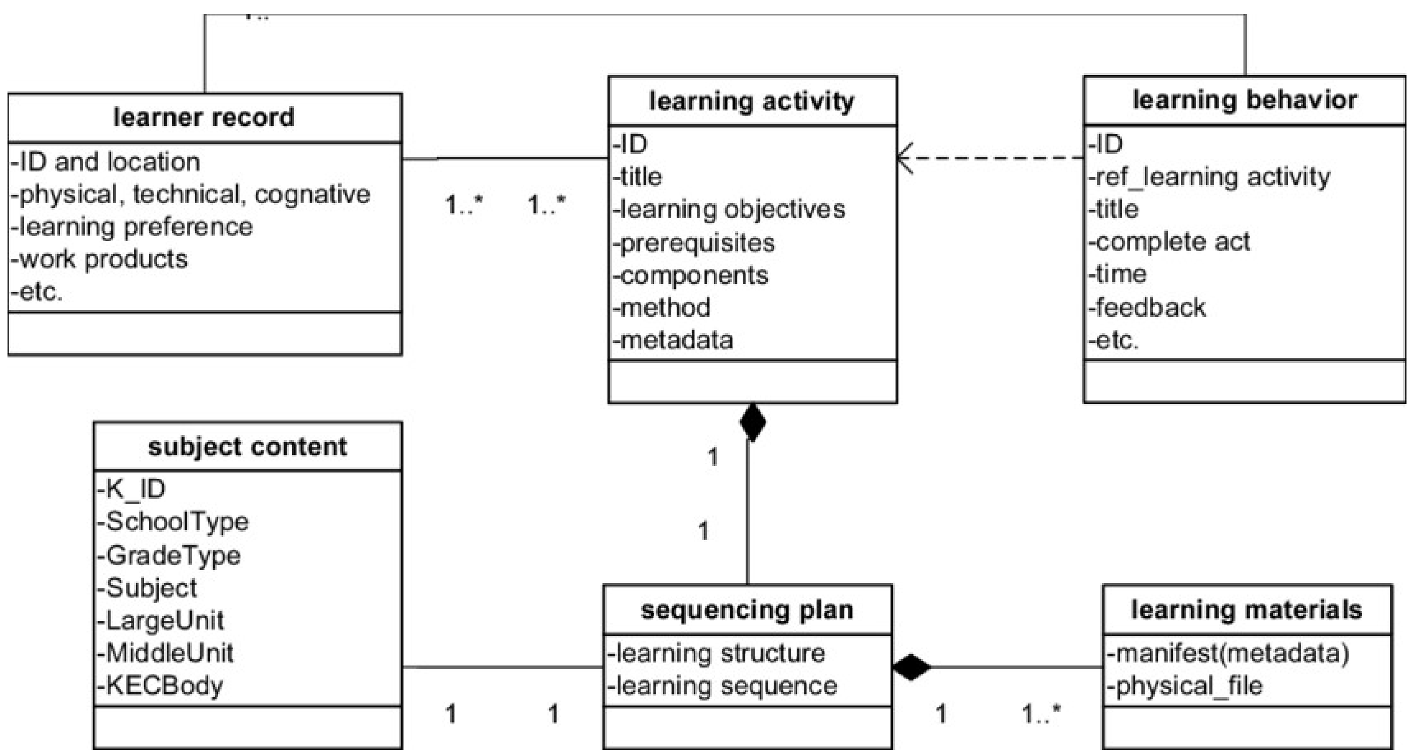
Data Structure Diagram

- DSDs differ from the E–R model in that the E–R model focuses on the relationships between different entities, whereas DSDs focus on the relationships of the elements within an entity.
- There are several styles for representing data structure diagrams, with the notable difference in the manner of defining cardinality.
- The choices are between arrow heads, inverted arrow heads (crow's feet), or numerical representation of the cardinality.

Data Structure Diagram

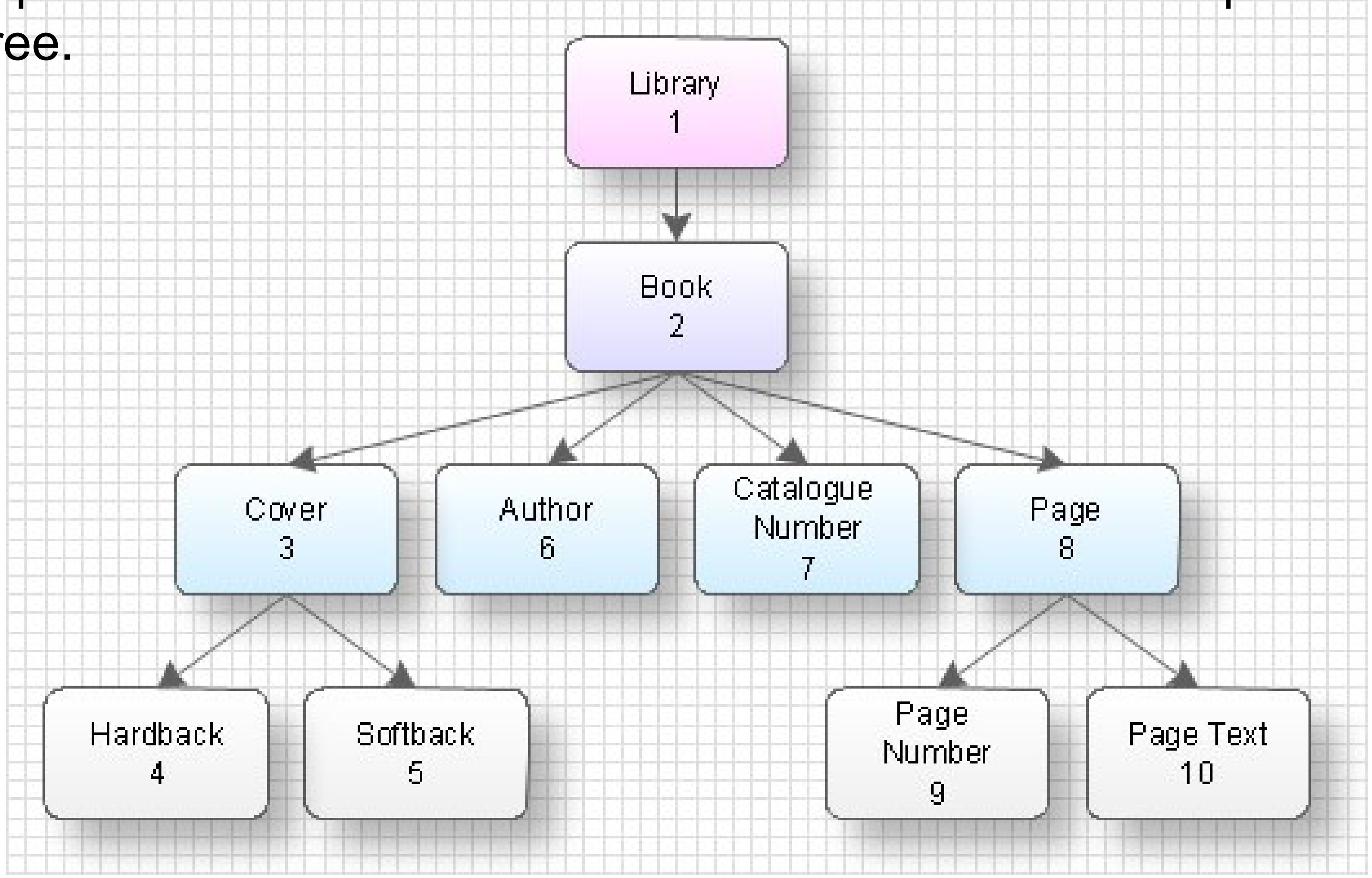
- DSDs typically use graphical symbols and notation to depict data elements and the relationships between them.
- For example, boxes or rectangles may be used to represent data classes or data structures, while arrows or lines may be used to represent relationships between data elements, such as inheritance or aggregation.
- DSDs can be used to illustrate the structure of various types of data structures, such as arrays, linked lists, trees, and graphs.
- They are useful for clarifying the relationships among data elements and for communicating the design of a data structure to others.

Data Structure Diagram



Data Structure Diagram

- DSD are also used for hierarchical presentation of the complicated entity.
- Such data structure diagram is usually created in the result of entity decomposition to the set level of detailed elaboration and presented as a tree.



Library 1
Book 2
Cover 3
Author 6
Catalogue Number 7
Page 8
Hardback 4
Softback 5
Page Number 9
Page Text 10