Chapter 4

Functional Modeling

Scanned by TapScanner

Scanned by TapScanner

4.0 Introduction

The functional model describes computations and specifies those aspects of the system concerned with transformations of values - functions, mappings, constraints, and functional dependencies. The functional model captures what the system does, without regard to how or when it is done.

The functional model uses a hierarchy of data flow diagrams (DFDs) in a similar fashion to the Yourdon method. However, the DFDs are not very well integrated with the object and dynamic models and it is difficult to imagine many projects making extensive use of the functional model.

Some aspects of the functional model are, however, quite useful. For example, the context diagram is vital for defining the scope of the system. Also, it may be possible to strengthen the real-time aspects of the method by introducing the concept of a processor and task model using DFDs.

Data Flow Diagrams are commonly used during problem analysis. They are quite general and are not limited to problem analysis for software requirements specification. DFDs are very useful in understanding a system and can be effectively used during analysis.

4.1 Objectives

In this chapter, you will learn what is functional model and DFDs. The chapter discusses how to draw a DFD, what is data dictionary and metadata. It also introduces concepts of candidate keys.

4.2 Presentation of Contents

4.2.1 Functional Modeling

The functional model describes computations and specifies those aspects of the system concerned with transformations of values - functions, mappings, constraints, and functional dependencies. The functional model captures what the system does, without regard to how or when it is done.

Scanned by TapScanner
Scanned by TapScanner

The functional model is represented graphically with multiple data flow diagrams, which show the flow of values from external inputs, through operations and internal data stores, to external outputs. Data flow diagrams show the dependencies between values and the computation of output values from input values and functions. Functions are invoked as actions in the dynamic model and are shown as operations on objects in the object model. The data flow diagram should adhere to OMT's notation and exploit the capabilities of OMT, such as nesting, control flows, and constraints.

4.2.1.1 Data Flow Diagram (DFD)

Data Flow Diagrams are commonly used during problem analysis. They are quite general and are not limited to problem analysis for software requirements specification. DFDs are very useful in understanding a system and can be effectively used during analysis.

A DFD shows the flow of data through a system. It views a system as a function that transforms the inputs into desired outputs. Any complex system will not perform this transformation in a "single step", and a data will typically undergo a series of transformations before it becomes the output. The DFD aims to capture the transformations that take place within a system to the input data so that eventually the output data is produced. The agent that performs the transformation of data from one state to another is called a process. So a DFD shows the movement of data through the different transformation or process in the system.

DFDs are basically of 2 types: Physical and logical ones. Physical DFDs are used in the analysis phase to study the functioning of the current system. Logical DFDs are used in the design phase for depicting the flow of data in a proposed system.

In a nutshell, a data flow diagram is a graph showing the flow of data values from their sources in objects through processes that transform them to their destinations in other objects.

Scanned by TapScanner
Scanned by TapScanner

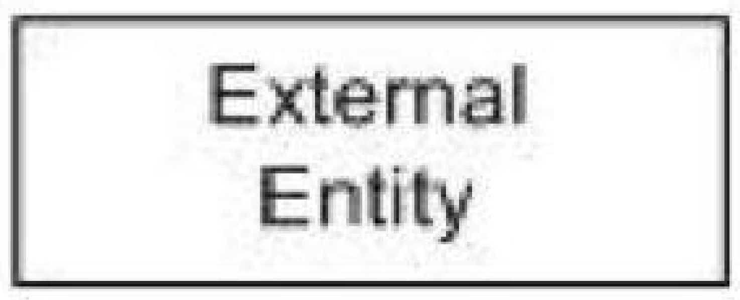
Data Flow Diagrams are composed of the four basic symbols – external entities, processes, data stores and data flow - as discussed below.

- The External Entity symbol represents sources of data to the system or destinations of data from the system.
- The Process symbol represents an activity that transforms or manipulates the data (combines, reorders, converts, etc.).
- The Data Store symbol represents data that is not moving (delayed data at rest).
- The Data Flow symbol represents movement of data.

Any system can be represented at any level of detail by these four symbols. Now, these four elements of DFD are discussed in detail.

4.2.1.1.1 External Entity

The External Entity symbol represents sources of data to the system or destinations of data from the system. They determine the system boundary. They are external to the system being studied. They are often beyond the area of influence of the developer. They can represent another system or subsystem. These go on margins/edges of data flow diagram. They are represented by a rectangle symbol and are named with appropriate name as shown in Fig below.



Some authors call them actors as they are active objects that drive the data flow diagram by producing or consuming values. Actors are attached to the inputs and outputs of a data flow diagram. Actors are also called as terminators as they act as source and sink for data.

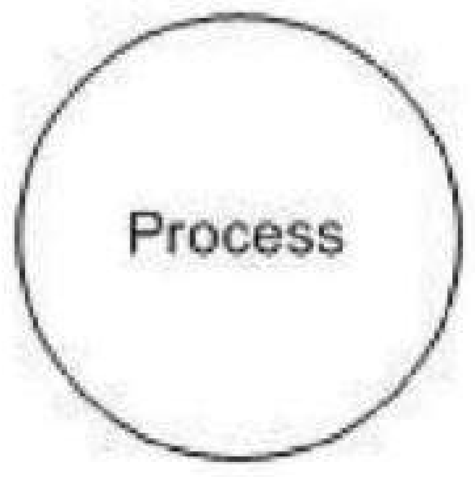
4.2.1.1.2 Process

Processes are work or actions performed on incoming data flows to produce outgoing data flows. These show data transformation or change. Data coming into a

Scanned by TapScanner
Scanned by TapScanner

process must be "worked on" or transformed in some way. Thus, all processes must have inputs and outputs. In some cases, data inputs or outputs will only be shown at more detailed levels of the diagrams. Each process is always "running" and ready to accept data. Major functions of processes are computations and making decisions. Each process may have dramatically different timing: yearly, weekly, daily etc.

A process is depicted by a circle as shown below:

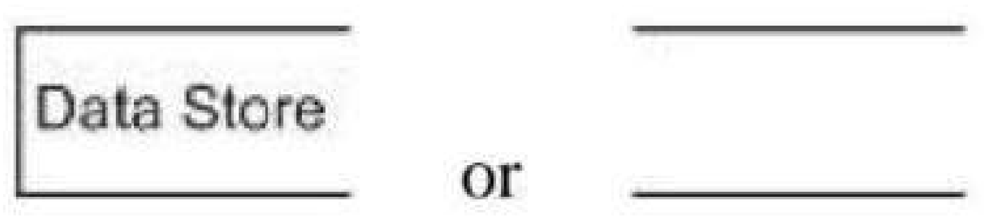


Every process is named. Processes are named with one carefully chosen verb and an object of the verb. There is no subject. Name is not to include the word "process". Each process should represent one function or action. If there is an "and" in the name, you likely have more than one function (and process). For example, get invoice, update customer and create Order. Processes are numbered within the diagram as convenient. Levels of detail are shown by decimal notation. For example, top level process would be Process 4, next level of detail Processes 4.1, and so on. Processes should generally move from top to bottom and left to right.

4.2.1.1.3 Data Store

Data stores are repository for data that are temporarily or permanently recorded within the system. It is an "inventory" of data. These are common link between data and process models. Only processes may connect with data stores.

There can be two or more systems that share a data store. This can occur in the case of one system updating the data store, while the other system only accesses the data. Data stores are represented by open rectangle or two parallel lines as shown below.



Scanned by TapScanner
Scanned by TapScanner

Data stores are named with an appropriate name, not to include the word "file", Names should consist of plural nouns describing the collection of data. Like customers, orders, and products. These may be duplicated.

They store data for later use. They do not generate any operation on its own but can respond to request. That is why they are passive objects in a data flow diagram.

4.2.1.1.4 Data Flow

Data flow represents the input (or output) of data to (or from) a process, data store or an actor. Data flow only data, not control. Represent the minimum essential data the process needs. Using only the minimum essential data reduces the dependence between processes. Data flows must begin and/or end at a process.

Data flows are always named. Name is not to include the word "data". It should be given unique names. Names should be some identifying noun. For example, marks, order, payment, complaint, registration no.

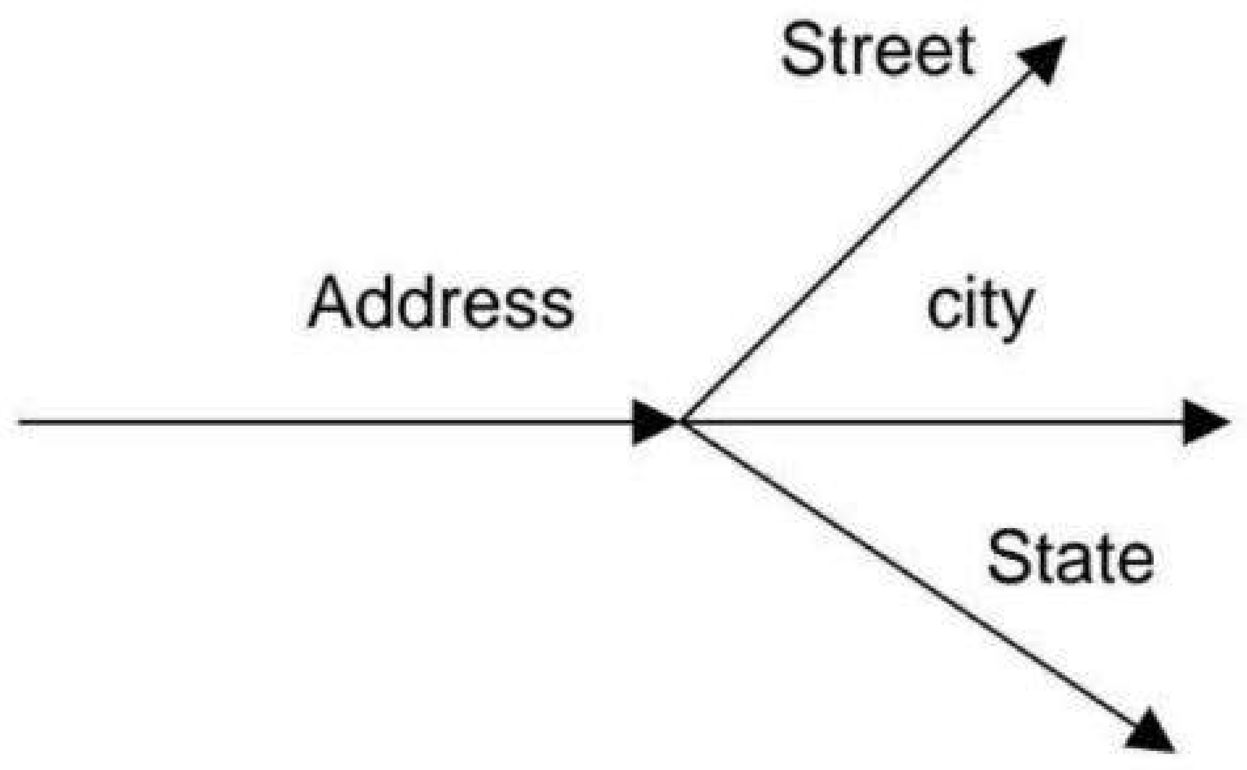
A data flow is represented by an arrow as shown in Fig below.

An arrow between the producer and the consumer of the data value represents a data flow. Arrow is labeled with description of data. Data can be elementary or aggregate. Input arrow indicates storing data in the data store and output arrow indicates accessing of data from data store.

Elementary data can not be decomposed into its meaningful constituents. For example, roll no, pin code, and quantity. Aggregate data can be decomposed into its meaningful constituents. For example, name can be decomposed into first name, middle name and last name. Sometimes an aggregate value is split into its constituents, each of which goes to a different process. A fork in the path as shown below is used to do this. Reverse can also be done. That is elementary data coming

Scanned by TapScanner
Scanned by TapScanner

from different sources can be aggregated. This is done by reversing the arrows in the diagram below.



4.2.1.2 Control Flow

Control flow is a Boolean value in the DFD that affects whether a process is evaluated or not. The control flow is not an input value to the process. It is represented by a dotted line from a process originating the Boolean value to the process being controlled as shown in fig below. This DFD is for a withdrawal from a bank account. The customer supplies a password and an amount. The update (withdrawal) can occur only when password is OK, which is shown as control flow in the diagram.

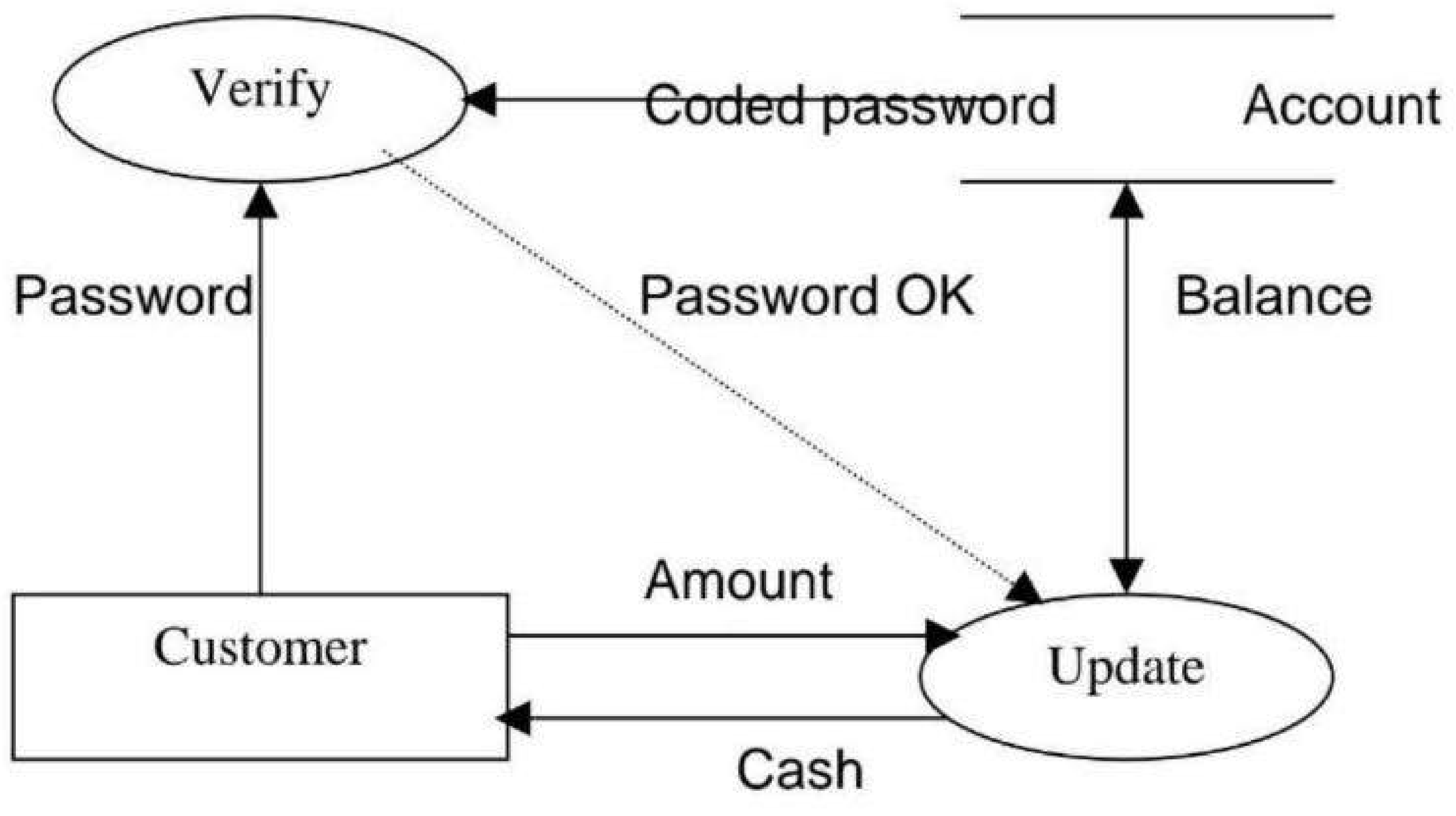


Figure: Control flow

Scanned by TapScanner
Scanned by TapScanner

4.2.3 Examples of DFDs

Example1: An example of a Data Flow Diagram - DFD for a system that pays workers is shown in the figure below. In this DFD there is one basic input data flow, the weekly time sheet, which originates from the source worker. The basic output is the pay check, the sink for which is also the worker. In this system, first the employee's record is retrieved, using the employee ID, which is contained in the time sheet. From the employee record, the rate of payment and overtime are obtained.

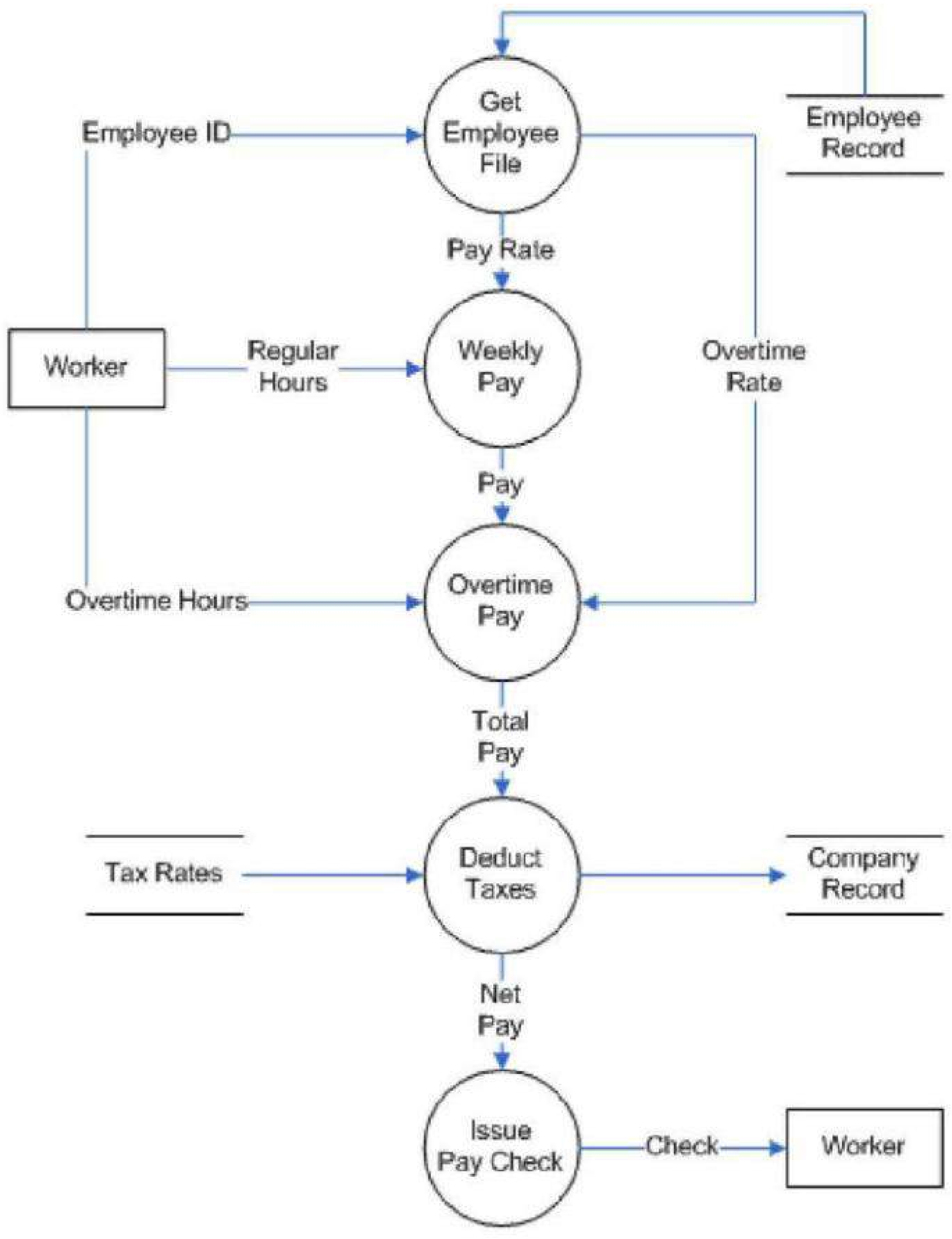


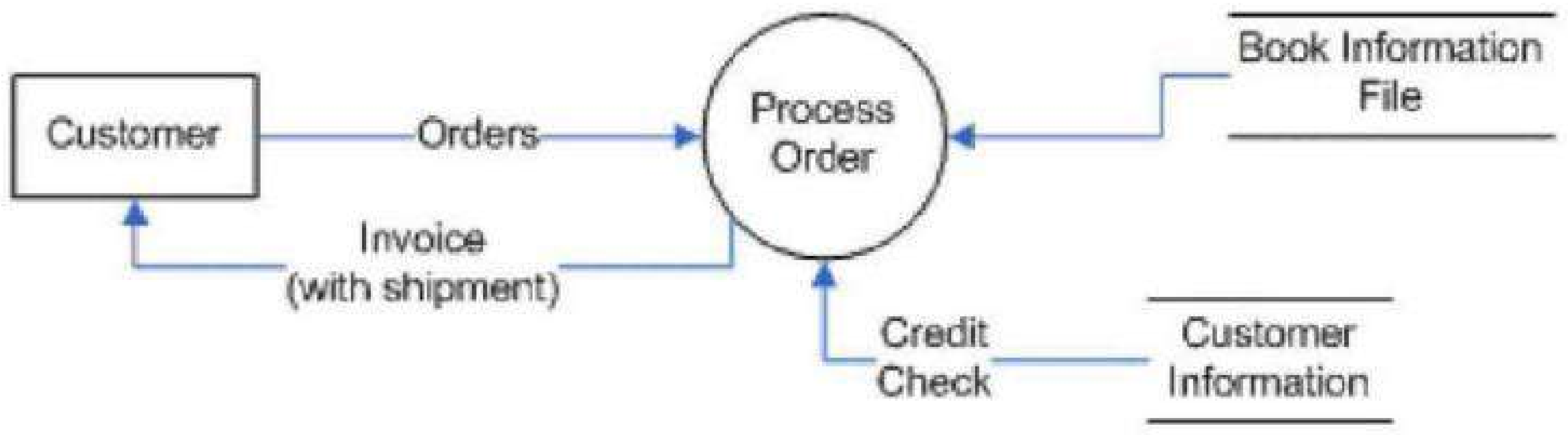
Figure: DFD of a system that pays workers.

These rates and the regular and overtime hours (from the time sheet) are used to complete the payment. After total payment is determined, taxes are deducted. To computer the tax deduction, information from the tax rate file is used. The amount of tax deducted is recorded in the employee and company records. Finally, the

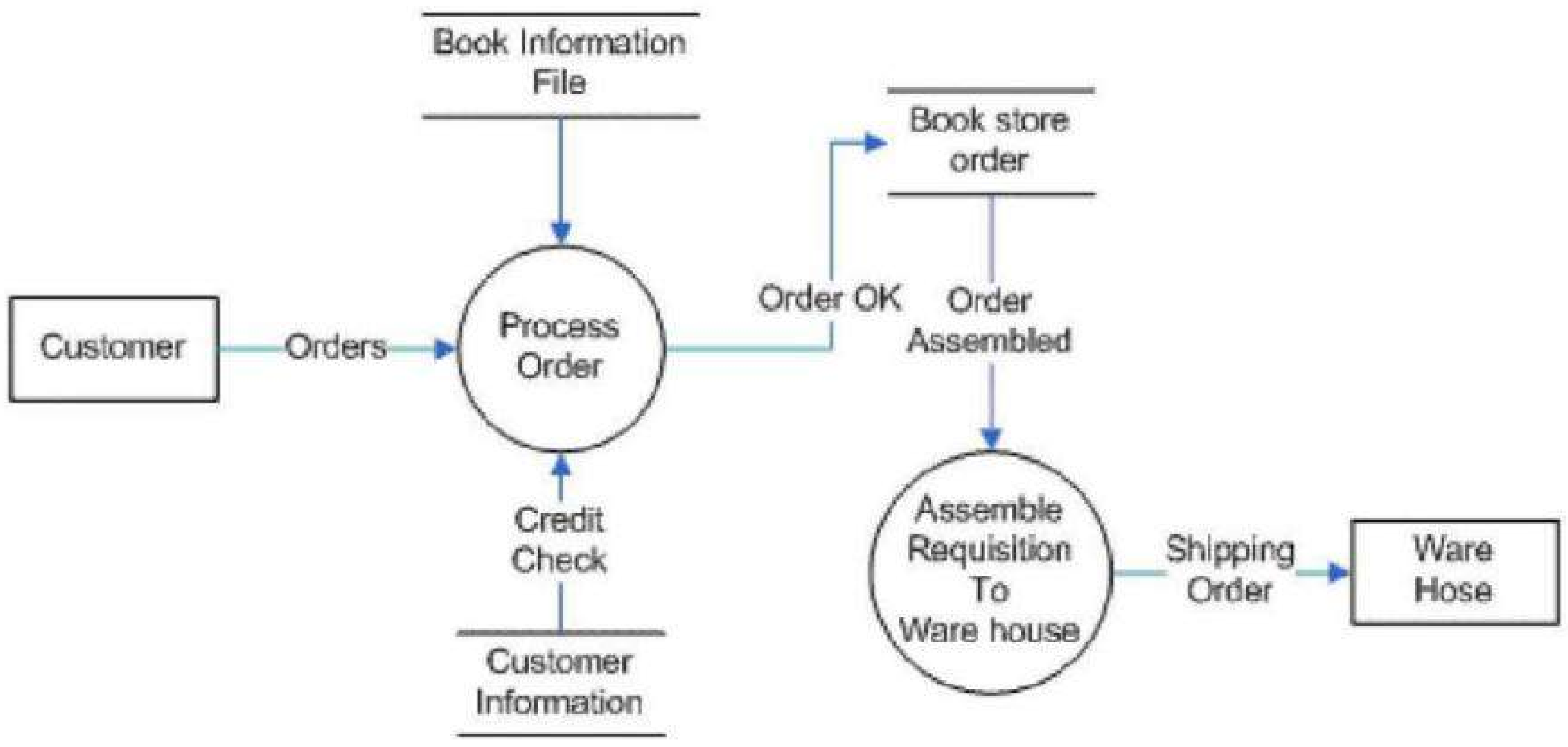
Scanned by TapScanner
Scanned by TapScanner

paycheck is issued for the net pay. The amount paid is also recorded in company records.

Following are the set of DFDs drawn for the General model of publisher's present ordering system.

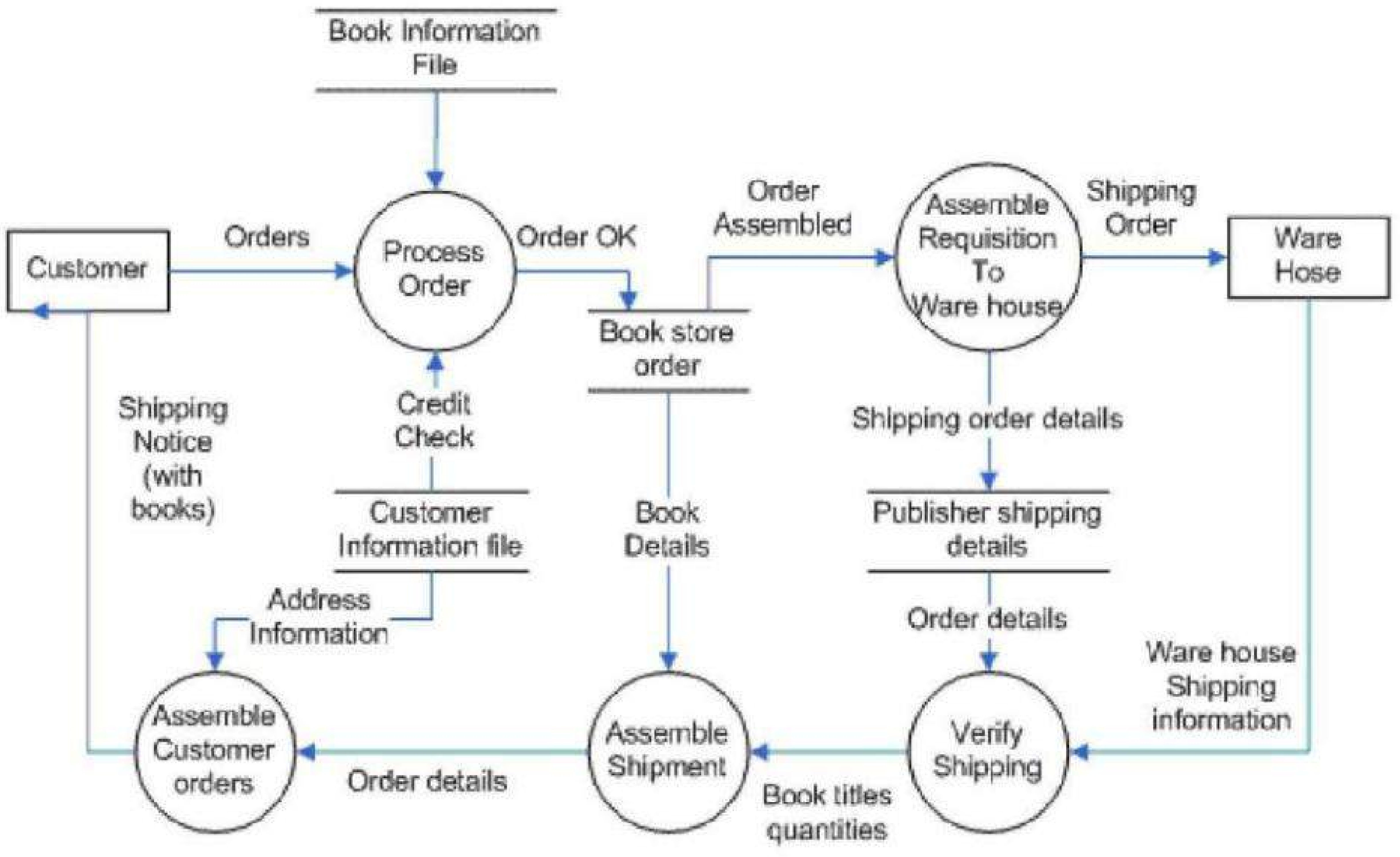


First Level DFD

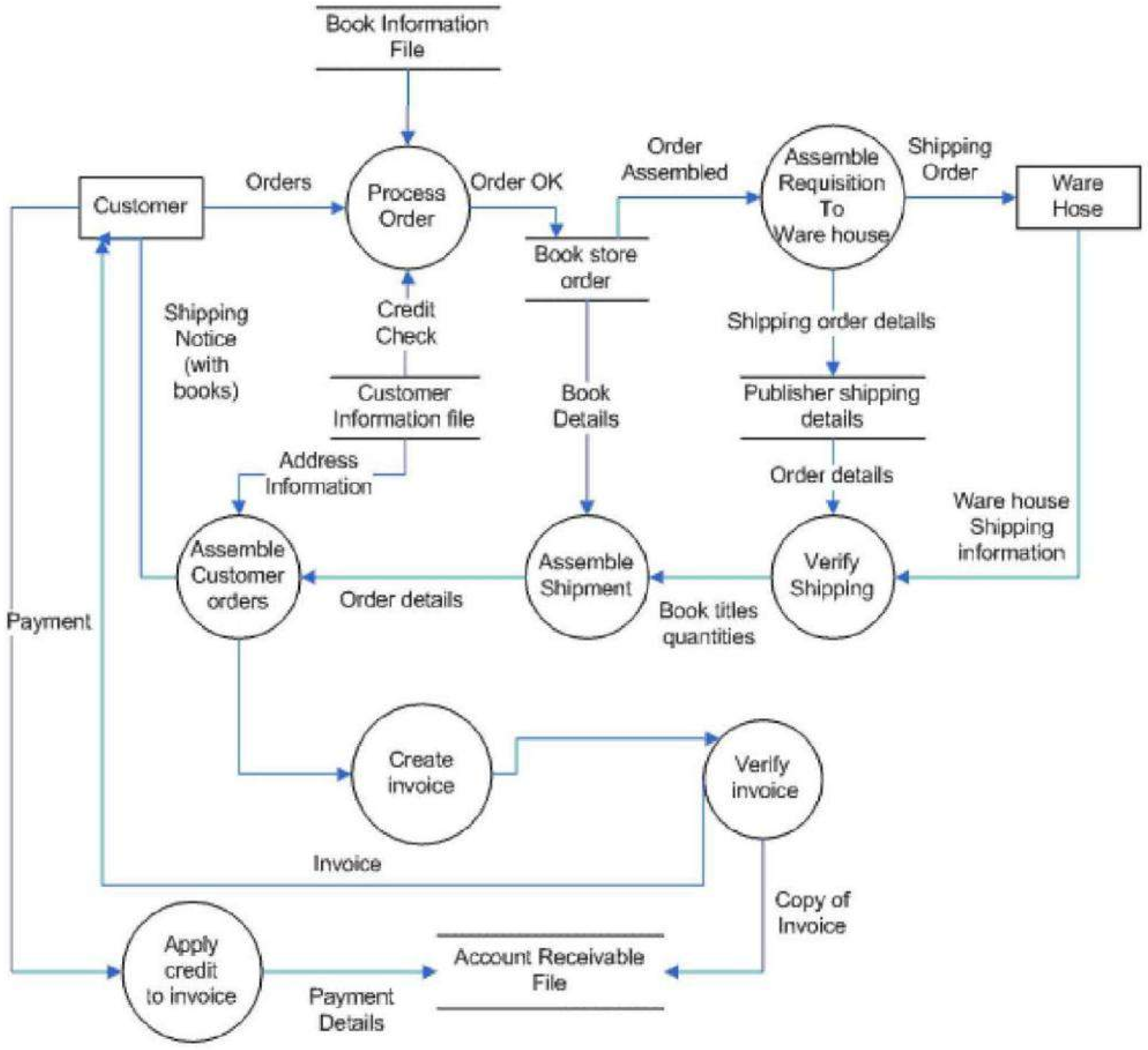


Second Level DFD - Showing Order Verification & credit check

Scanned by TapScanner
Scanned by TapScanner



Third Level DFD - Elaborating an order processing & shipping



Fourth level DFD: Completed DFD, Showing Account Receivable Routine.

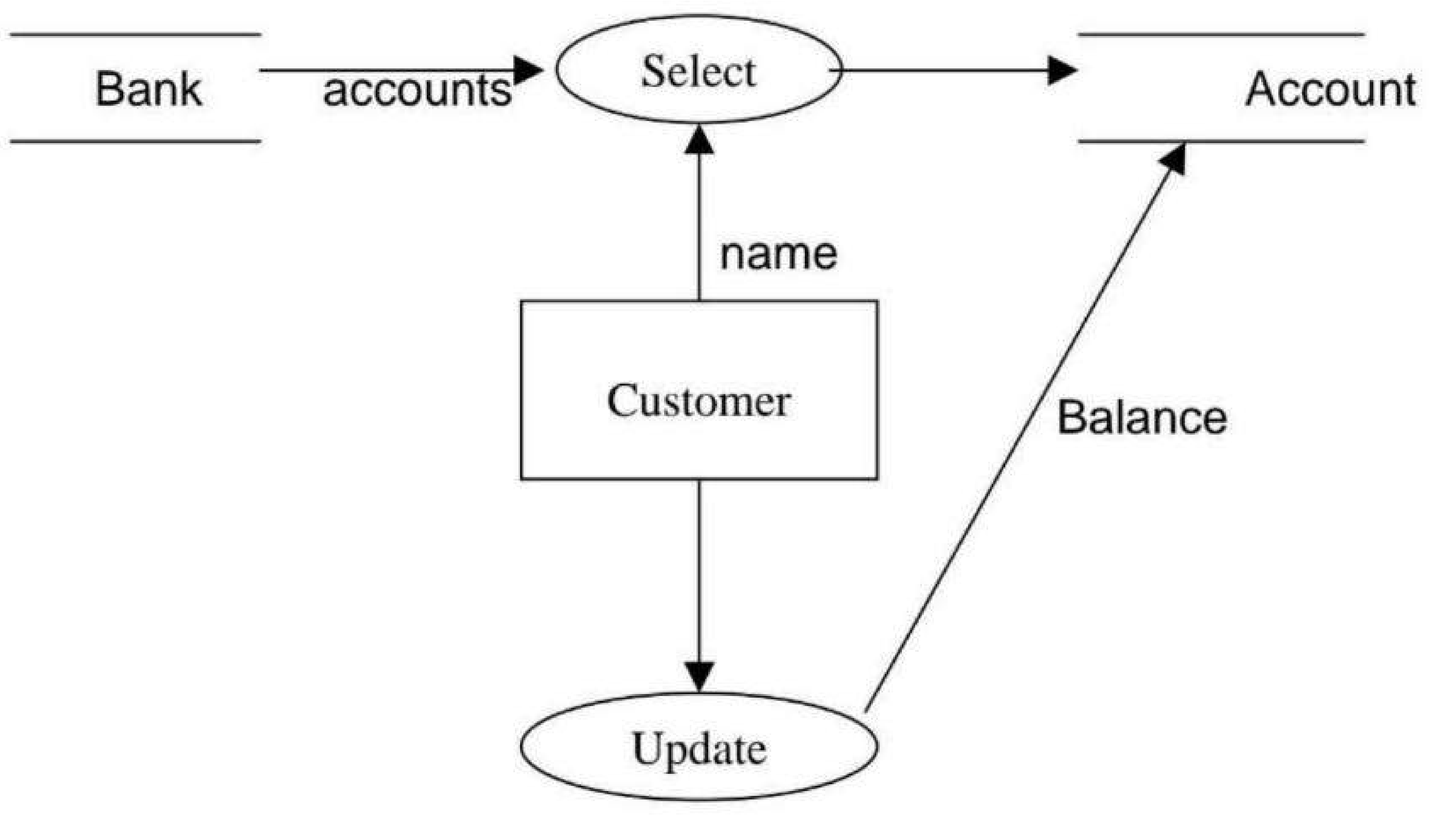
Scanned by TapScanner
Scanned by TapScanner

From the level one it shows the publisher's present ordering system. Let's expand process order to elaborate on the logical functions of the system. First, incoming orders are checked for correct book titles, author's names, and other information and then batched into other book orders from the same bookstore to determine how may copies can be shipped through the ware house. Also, the credit status of each book stores is checked before shipment is authorized. Each shipment has a shipping notice detailing the kind and numbers of booked shipped. This is compared to the original order received (by mail or phone) to ascertain its accuracy. The details of the order are normally available in a special file or data store, called "Bookstore Orders". It is shown in the second level DFD diagram.

Following the order verification and credit check, a clerk batches the order by assembling all the book titles ordered by the bookstore. The batched order is sent to the warehouse with authorization to pack and ship the books to the customer. It is shown in the third level DFD diagram.

Further expansion of the DFD focuses on the steps in billing the bookstore shown in the fourth level DFD, additional functions related to accounts receivable.

Example 3: DFD below shows updation of a bank account in a banking system.



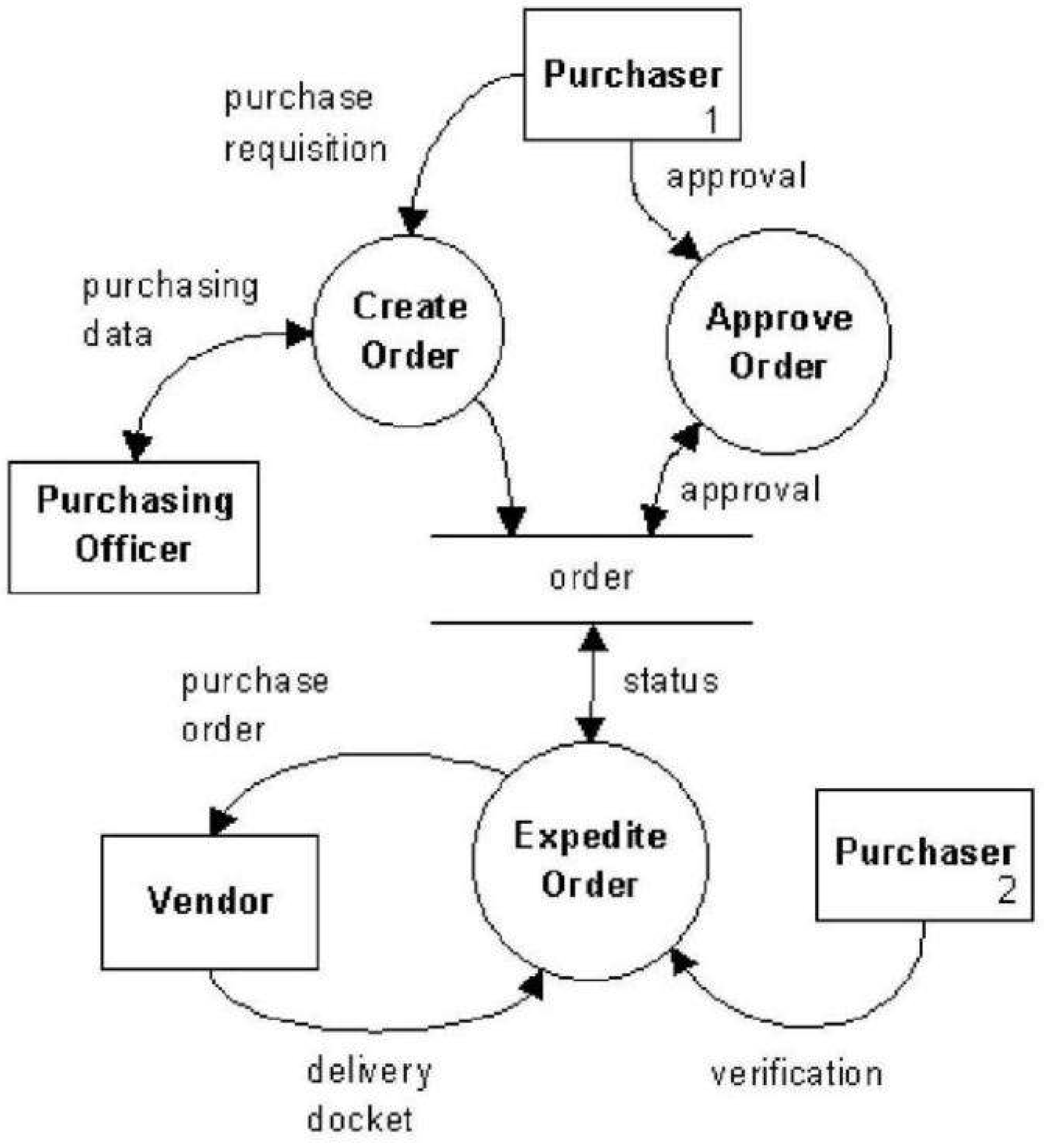
Scanned by TapScanner
Scanned by TapScanner

Example 4: DFD below shows the purchase order processing system. There are four actors involved in this system – Purchasing Officer, Vendor, Purchaser 1 and Purchaser 2. Purchasing Officer creates purchase orders on receiving the purchase acquisition requests from the perspective purchasers. Purchasers approve orders and they can verify the status of the orders. All order details are kept in a database.

This functional model represents:

Functions. For example, create order and approve order.
Data flows. For example, purchasing data flows from the purchasing officer to the create order function.
External entities. For example, a vendor is an entity external to the system.
Data stores. For example, orders are kept in an order data store

Scanned by TapScanner
Scanned by TapScanner



4.2.2 Data Dictionary and Meta Data

In the data flow diagrams, we have given 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, the interest is to build some structures place to keep details of the contents of data flows, processes and data stores. Here comes the concept of data dictionary.

A data dictionary is a structured repository of data about data. It is a set of rigorous definitions of all DFD data elements and data structures. To define the data structure, different notations are used. These are similar to the notations for regular expression. Essentially, besides sequence or composition (represented by +) selection and iteration are included. Selection (represented by vertical bar "|") means one or the other, and repetition (represented by "*") means one or more occurrences.

Scanned by TapScanner
Scanned by TapScanner

The data dictionary for the DFD of system that pays to workers given above is created as shown below:

Weekly timesheet = Employee_Name + Employee_ID + {Regular_hours + overtime_hours}

Pay_rate = {Horly | Daily | Weekly} + Rupees_amount

Employee_Name = Last + First + Middle_Initial

Employee_ID = digit + digit + digit + digit

Most of the data flows in the DFD are specified here. Some of the most obvious ones are not shown here. The data dictionary entry for weekly timesheet specifies that this data flow is composed of three basic data entities - the employee name, employee ID and many occurrences of the two - tuple consisting of regular hours and overtime hours. The last entity represents the daily working hours of the worker. The data dictionary also contains entries for specifying the different elements of a data flow.

Once we have constructed a DFD and its associated data dictionary, we have to somehow verify that they are "correct". There can be no formal verification of a DFD, because what the DFD is modeling is not formally specify anywhere against which verification can be done. Human processes and rule of thumb must be used for verification. In addition to the walkthrough with the client, the analyst should look for common errors. Some common errors are

- Unlabeled data flows.
- Missing data flows: Information required by a process is not available.
- Extraneous data flows: Some information is not being used in the process
- Consistency not maintained during refinement
- Missing processes
- Contains some control information

The DFDs should be carefully scrutinized to make sure that all the processes in the physical environment are shown in the DFD. It should also be ensured that none of the data flows is actually carrying control information.

Scanned by TapScanner
Scanned by TapScanner

Meta Data: It is loosely defined as data about data. Metadata is a concept that applies mainly to electronically archived or presented data and is used to describe the: a) definition, b) structure and c) administration of data files with all contents in context to ease the use of the captured and archived data for further use. For example, a web page may include metadata specifying what language it's written in, what tools were used to create it, where to go for more on the subject and so on.

Metadata is defined as data providing information about one or more other pieces of data, such as:

- means of creation
- purpose of the data
- time and date of creation
- creator or author of data
- placement on a network (electronic form) where the data was created,
- What standards used etc.

For example: A digital image may include metadata that describes how large the picture is, the color depth, the image resolution, when the image was created, and other data. A text document's metadata may contain information about how long the document is, who the author is, when the document was written, and a short summary of the document.

Metadata is data. As such, metadata can be stored and managed in a database, often called a registry or repository. However, it is impossible to identify metadata just by looking at it. We don't know when data is metadata or just data.

Metadata is structured data which describes the characteristics of a resource. It shares many similar characteristics to the cataloguing that takes place in libraries, museums and archives. The term "meta" derives from the Greek word denoting a nature of a higher order or more fundamental kind. A metadata record consists of a number of pre-defined elements representing specific attributes of a resource, and each element can have one or more values. Below is an example of a simple metadata record:

Scanned by TapScanner
Scanned by TapScanner

Element name	**Value**
Title	Web catalogue
Creator	Rajender Nath
Publisher	G.J.U. Hisar
Identifier	www.gju.ac.in.metadata.html
Format	Text/html
Relation	Distance Education Web site

Each metadata schema will usually have the following characteristics:

- a limited number of elements
- the name of each element
- the meaning of each element

Typically, the semantics is descriptive of the contents, location, physical attributes, type (e.g. text or image, map or model) and form (e.g. print copy, electronic file). Key metadata elements supporting access to published documents include the originator of a work, its title, when and where it was published and the subject areas it covers. Where the information is issued in analog form, such as print material, additional metadata is provided to assist in the location of the information, e.g. call numbers used in libraries. The resource community may also define some logical grouping of the elements or leave it to the encoding scheme. For example, Dublin Core may provide the core to which extensions may be added.

In a nutshell, metadata is structured information that describes, explains, locates, or otherwise makes it easier to retrieve, use, or manage an information resource. Metadata is often called data about data or information about information.

There are three main types of metadata:

Scanned by TapScanner
Scanned by TapScanner

Descriptive metadata: It describes a resource for purposes such as discovery and identification. It can include elements such as title, abstract, author, and keywords.
Structural metadata: It indicates how compound objects are put together, for example, how pages are ordered to form chapters.
Administrative metadata: It provides information to help manage a resource, such as when and how it was created, file type and other technical information, and who can access it.

4.2.3 Steps to Produce a DFD

- Identify and list external entities providing inputs/receiving outputs from system
- Identify and list inputs from/outputs to external entities
- Draw a context DFD

Defines the scope and boundary for the system and project

- Think of the system as a container (black box)
- Ignore the inner workings of the container
- Ask end-users for the events the system must respond to
- For each event, ask end-users what responses must be produced by the system
- Identify any external data stores
- Draw the context diagram

    i. Use only one process

    ii. Only show those data flows that represent the main objective or most common inputs/outputs

- identify the business functions included within the system boundary
- identify the data connections between business functions
- confirm through personal contact sent data is received and vice-versa
- trace and record what happens to each of the data flows entering the system (data movement, data storage, data transformation/processing)
- Draw an overview DFD

Scanned by TapScanner
Scanned by TapScanner

- Shows the major subsystems and how they interact with one another
- Exploding processes should add detail while retaining the essence of the details from the more general diagram

- Consolidate all data stores into a composite data store

- Draw middle-level DFDs

  - Explode the composite processes

- Draw primitive-level DFDs

  - Detail the primitive processes

  - Must show all appropriate primitive data stores and data flows

- verify all data flows have a source and destination;
- verify data coming out of a data store goes in;
- review with "informed";
- Explode and repeat above steps as needed.

Balancing DFDs

- Balancing: child diagrams must maintain a balance in data content with their parent processes
- Can be achieved by either:
- exactly the same data flows of the parent process enter and leave the child diagram, or
- the same net contents from the parent process serve as the initial inputs and final outputs for the child diagram or
- the data in the parent diagram is split in the child diagram

Rules for Drawing DFDs

- A process must have at least one input and one output data flow

Scanned by TapScanner
Scanned by TapScanner

- A process begins to perform its tasks as soon as it receives the necessary input data flows
- A primitive process performs a single well-defined function
- Never label a process with an IF-THEN statement
- Never show time dependency directly on a DFD
- Be sure that data stores, data flows, data processes have descriptive titles. Processes should use imperative verbs to project action.
- All processes receive and generate at least one data flow.
- Begin/end data flows with a bubble.

Rules for Data Flows

- A data store must always be connected to a process
- Data flows must be named
- Data flows are named using nouns Customer ID, Student information
- Data that travel together should be one data flow
- Data should be sent only to the processes that need the data

Use the following additional guidelines when drawing DFDs

- Identify the key processing steps in a system. A processing step is an activity that transforms one piece of data into another form.
- Process bubbles should be arranged from top left to bottom right of page.
- Number each process (1.0, 2.0, etc). Also name the process with a verb that describes the information processing activity.
- Name each data flow with a noun that describes the information going into and out of a process. What goes in should be different from what comes out.
- Data stores, sources and destinations are also named with nouns.
- Realize that the highest level DFD is the context diagram. It summarizes the entire system as one bubble and shows the inputs and outputs to a system
- Each lower level DFD must balance with its higher level DFD. This means that no inputs and outputs are changed.

Scanned by TapScanner
Scanned by TapScanner

- Think of data flow not control flow. Data flows are pathways for data. Think about what data is needed to perform a process or update a data store. A data flow diagram is not a flowchart and should not have loops or transfer of control. Think about the data flows, data processes, and data storage that are needed to move a data structure through a system.
- Do not try to put everything you know on the data flow diagram. The diagram should serve as index and outline. The index/outline will be "fleshed out" in the data dictionary, data structure diagrams, and procedure specification techniques.

4.2.4 Different Types of Keys

This concept is related to relational data base management systems. A relation is two dimensional table that has rows called tuples and columns called domains. There are different types of keys, namely Primary keys, alternate keys, etc. , which are described below.

Primary key: Within a given relation, there can be one attribute with values that are unique within the relation that can be used to identify the tuples of that relation. That attribute is said to be primary key for that relation. For example, in a Student relation roll no is a primary key.

Composite primary key: Not every relation will have a single-attribute primary key. There can be a possibility that some combination of attributes when taken together have the unique identification property. These attributes as a group is called composite primary key. A combination consisting of a single attribute is a special case.

Existence of such a combination is guaranteed by the fact that a relation is a set. Since sets don't contain duplicate elements, each tuple of a relation is unique with respect to that relation. Hence, at least the combination of all attributes has the unique identification property.

Scanned by TapScanner
Scanned by TapScanner

- In practice it is not usually necessary to involve all the attributes-some lesser combination is normally sufficient. Thus, every relation does have a primary (possibly composite) key.
- Tuples represent entities in the real world. Primary key serves as a unique identifier for those entities.

Candidate key: In a relation, there can be more than one attribute combination possessing the unique identification property. These combinations, which can act as primary key, are called candidate keys.

EmpNo	SocSecurityNo	Name	Age
1011	2364236	Harry	21
1012	1002365	Sympson	19
1013	1056300	Larry	24

Table: having "EmpNo" and "SocSecurityNo" as candidate keys

Alternate key: A candidate key that is not a primary key is called an alternate key. In fig. 8.6 if EmpNo is primary key then SocSecurityNo is the alternate key.

4.3 Summary

- The functional model describes computations and specifies those aspects of the system concerned with transformations of values - functions, mappings, constraints, and functional dependencies.
- The functional model is represented graphically with multiple data flow diagrams, which show the flow of values from external inputs, through operations and internal data stores, to external outputs.
- DFDs play a major role in designing of the software and also provide the basis for other design-related issues. Some of these issues are addressed in this chapter. All the basic elements of DFD are further addressed in the designing phase of the development procedure

Scanned by TapScanner
Scanned by TapScanner

- Metadata is structured information that describes, explains, locates, or otherwise makes it easier to retrieve, use, or manage an information resource. Metadata is often called data about data or information about information.

- Within a given relation, there can be one attribute with values that are unique within the relation that can be used to identify the tuples of that relation. That attribute is said to be primary key for that relation.

4.4 Self-Assessment Questions

1. What is functional model of OMT? How is it related to other models of OMT? Explain.
2. What is DFD? What are elements of DFD? Explain the purpose of DFD with a suitable example.
3. Distinguish between control flow and data flow.
4. Discuss the steps to draw a DFD systematically.
5. What is metadata? Explain its usefulness.
6. What is data dictionary? How is it created?
7. What is a key? Explain different types of keys with suitable examples.

Scanned by TapScanner
Scanned by TapScanner