UNIT 3 FUNCTIONAL MODELING

Structure **Page Nos.**

3.0 INTRODUCTION

As discussed in the previous Unit of this Course *"the dynamic model represents control information**: the sequences of events, states, and operations** that occur within a system of objects"*. The dynamic model is a pattern that specifies possible scenarios that may occur. An event is a signal that something has happened. A state represents the interval between events, and specifies the context in which events are interpreted. An action is an instantaneous operation in response to an event.

The functional modeling is the third, and final phase of the OMT model. The functional modeling is a complex modeling. It describes how the output values in a computation are derived from input values. *The **functional model** specifies **what happens**, the dynamic model specifies when it happens, and the object model describes what it happens, and the object model describes what happens to an object*. The functional model is **consist of multiple data flow diagrams**, which show the flow of values from input to output through operations and internal data stores. It also includes constraints among values within an object model.

In this Unit you will learn functional modeling concepts and data flow diagrams. Data flow diagrams do not show control or object structure information; these belong to the dynamic and object models. In this Unit in our discussion will follow the traditional form of the data flow diagram, with which you are familiar.

3.1 OBJECTIVES

After going through this unit, you should be able to:

- describe the Function Model;
- explain the concept of DFD;
- implement dataflow in the functional model;
- describe features of the data flow diagram;
- explain limitations in the design of the Data Flow Diagram, and
- relate the Object Model, Dynamic Model, and Functional Model.

3.2 FUNCTIONAL MODELS

Let us start our discussion by answering the question "what is a functional model?"

The functional model is the third leg of the OMT methodology in addition to the **Object Model** and **Dynamic Model**. *"**The functional model specifies the results of a computation specifying how or when they are computed**"*. The functional model specifies the meaning of the **operations in the object model** and the **actions in the dynamic model**, as well as any constraints in the object model. Non-interactive programs, such as compilers, have a trivial dynamic model; the purpose of a compiler is to compute a function. The functional model is the main model for such programs, although the object model is important for any problem with nontrivial data structures. Many interactive programs also have a significant functional model. By contrast, databases often have a trivial functional model, since their purpose is to store and **organize data**, not **to transform it**.

For example, **spreadsheet is** a **kind of functional model**. In many cases, the values in the spreadsheet are trivial and cannot be structured further. The only interesting object structure in the spreadsheet is the cell. The aim of the spreadsheet is to specify values in terms of other values.

Let us take the case of a compiler. A compiler is almost a **pure computation**. The input for a compiler is the text of a program in a particular language; the output is an object file that implements the program in another language often the machine language of a particular computer. Here, the **mechanics of compilation are** not **concerned** with the **functional model**. Now, we will discuss the data flow diagram. It is very helpful in visualizing the flow of data in the system, and to show the involvement of different processes at different levels.

3.3 DATA FLOW DIAGRAMS

Here, we will discuss data flow diagram and their uses in a Functional Model. As you know, the functional model consists of **multiple data flow diagrams** which specify the meanings, of operations and constraints. A data flow diagram (DFD) shows the functional relationships of the values computed by a system, including input values, output values, and internal data stores. "***A data flow diagram is a graph which shows the flow of data values from their sources in objects through processes that transform them to their destinations in other objects***". DFDs do not show control information, such as the time at which processes are executed, or decisions among alternate data paths. This type of information belongs to the dynamic model. Also, the arrangement values into object are shown by the object model, but not by the data flow diagram.

A data flow diagram contains **processes** which **transform data, data flows** which move data, **actor objects** which produce and consume data, and **data store** objects that store data passively. *Figure 1* shows a data flow diagram for the display of an icon on a windowing system. Here in this figure, the icon name and location are inputs to the diagram from an unspecified source. The icon is expanded to vectors in the application coordinate system using existing icon definitions. The vectors are clipped to the size of the window, then offset by the location of the window on the screen, to obtain vectors in the screen coordinate system. Finally, the vectors are converted to pixel operations that are sent to the screen buffer for display. The data flow diagram represents the sequence of transformations performed, as well as the **external values** and objects that affect the computation process.

Now, let us turn to the basic feature of DFD.

3.4 FEATURES OF A DATA FLOW DIAGRAM

The DFD has many features. It shows the computation of values in different states. The building blocks and features of DFD are:

3.4.1 Processes

"*The term **process** means all the computation activities that are involved from the input phase to the output phase"*. Each process contains a fixed number of input and output data arrows. These arrows carry a value of a given type. A process transforms data values. The lowest-level processes are pure functions without side effects.

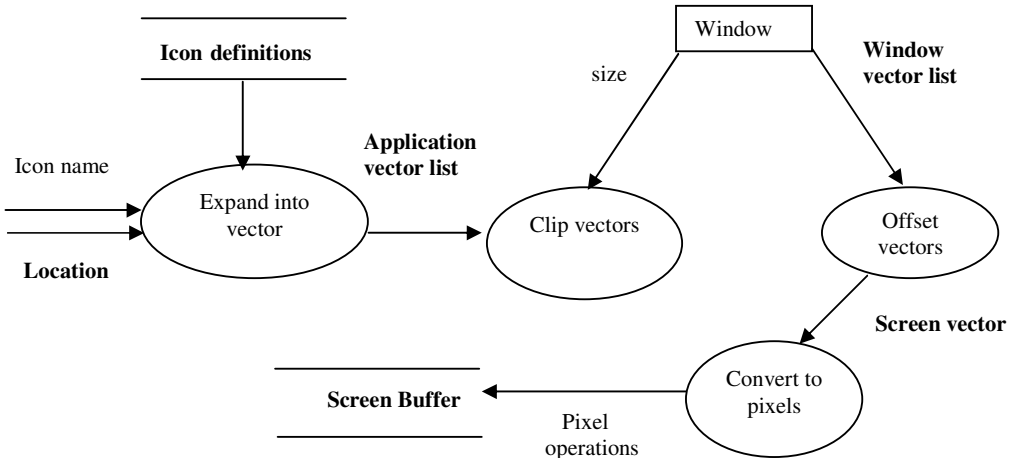


Figure 1: Data flow diagram for windowed graphics display

An entire data flow graph is a **highlevel process**. A process may have side effects if it contains non-functional components, such as **data stores** or **external objects**. The functional model does not uniquely specify the results of a process with side effects. The functional model only indicates **the possible functional paths**; it does not show which path will actually occur. The results of such a process depend on the **behavior of the system**, as specified by the dynamic model. Some of the examples of non-functional processes are **reading and writing files**, a **voice recognition algorithm** that **learns from experience**, and the display of images within a workstation windowing system.

A process is represented with the ellipse symbol and the name of process is written in it. Each process has a fixed number of input and output data arrows, each of which carries a value of a given type. The inputs and outputs can be labeled to show their role in the computation. In *Figure 2,* two processes are shown. Here, you should note that a process can have more than one output. The display icon process represents the entire data flow diagram of *Figure1* at a higher level of abstraction.

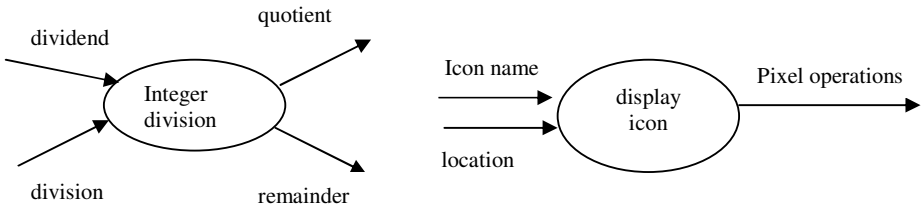


Figure 2: Processes

The diagram of processes shows only the pattern of inputs and outputs. The computation of output values from input values must also be specified. A high-level process can be expanded into an entire data flow diagram such as a subroutine which can be expanded into lower-level subroutine. Recursion processes **must be stopped** in a data flow diagram. The atomic processes must be described directly, in natural

language, mathematical equations, or by some other means. For example, *integer division* could be defined mathematically and "display icon" would be defined in terms of *Figure 1*. The atomic processes are trivial and simply access a value from an object.

3.4.2 Data Flows

The term data flow literally means flow of data. A data flow connects the output of an object or process to the input of another object or process. It represents an intermediate data value within a computation. Here, you should note that the value is not changed by the data flow.

A data flow is represented with the symbol arrow, and is used to connect the producer and the consumer of the data value. The arrow is labeled with a description of the data, usually, its name or type. The same value can be sent to several places and this is indicated by a fork with several arrows emerging from it. The output arrows are unlabeled because they represent the same value as the input. Some data flows are shown in *Figure 3* below.

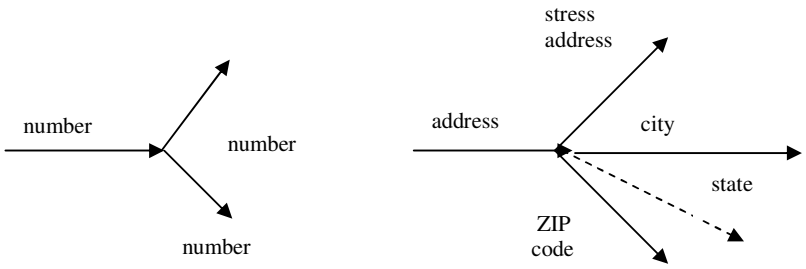


Figure 3: Data flows to copy a value and split an aggregate value

Sometimes, an aggregate data value is split into its components, each of which goes to a different process. This is shown by **a fork** in the path in which each outgoing arrow is labeled with the name of its component. The combination of several components into an aggregate value is just the opposite of it.

Each data flow represents a value at some point in the computation. The data flows internal to the diagram represent intermediate values within a computation and do not necessarily have any significance in the real world.

Flows on the boundary of a data flow diagram are its inputs and outputs. These flows may be unconnected, or they may be connected to objects. The inputs in *Figure 3* are the number and address of the location; their sources must be specified in the larger context in which the diagram is used.

3.4.3 Actors

The term actor means an object which can perform actions. It is drawn as a rectangle to show that it is an object. An actor is an active object that drives the data flow graph by producing or consuming values. Actors are attached to the inputs and outputs of a data flow graph. In a sense, the actors lie on the boundary of the data flow graph, but terminate the flow of data as sources and sinks of data, and so are sometimes called **terminators**. Examples of actors are the user of a program, a thermostat, and a motor under computer control. The actions of the actors are **outside the scope** of the data flow diagram, but should be part of the dynamic model.

3.4.4 Data Stores

The term data store literally means the place where data is stored. It is a passive object within a data flow diagram that stores data for later access. Unlike an actor, a data store does not generate any operations on its own, but merely responds to requests to store and to access data. A data store allows values to be accessed in a different order than they are generated. Aggregate data stores, such as lists and tables,

provide accesses to data by insertion order or by index keys. Some of the examples of data stores are the database of airline seat reservations, a bank account, and a list of temperature readings over the past day.

A data store is represented by a pair of parallel lines containing the name of the store. Input arrows indicate information or operations that modify the stored data. Some of the operations which we can perform are adding elements, modifying values, or deleting elements. Output arrows indicate information retrieved from the store. This includes retrieving at the values, or some part of it.

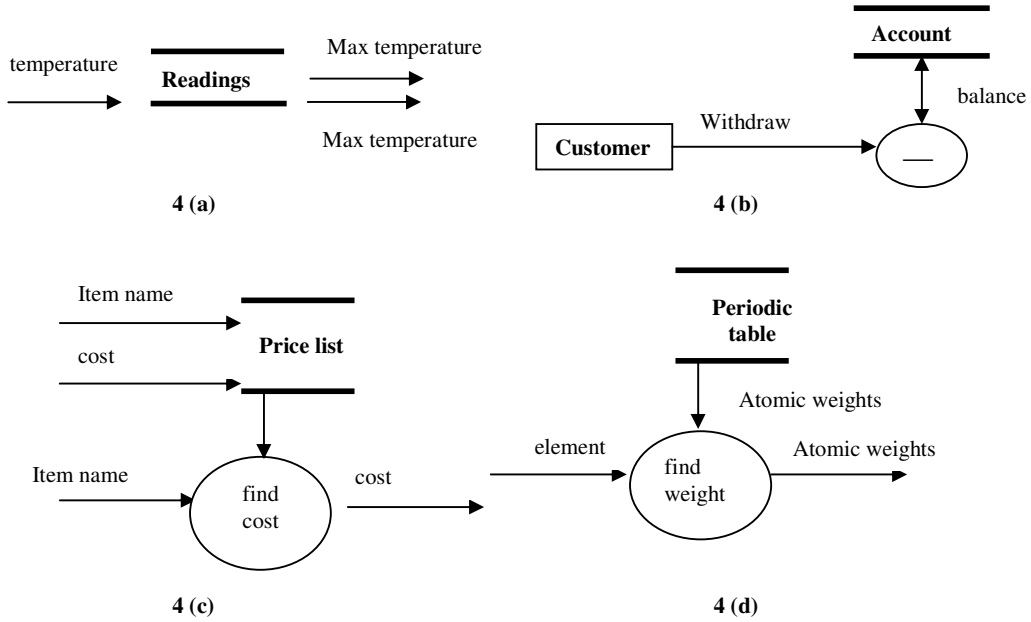


Figure 4: Data stores

Figure 4 shows a data stores for temperature readings. Every hour a new temperature reading enters the store. At the end of the day, the maximum and minimum readings are retrieved from the store. The data store permits many pieces of data to be accumulated so that the data can be used later on.

In *Figure 4b* data store for a bank account is given. The double-headed arrow indicates that a balance is both an input and an output of the subtraction operation. This can be represented with two separate arrows. Accessing and updating of the value in a data store is a common operation.

In *Figure 4c,* a price list for items is shown. Input to the store consists of pairs of item name and cost values. Later, an item is given, and the corresponding cost is found. The unlabeled arrow from the data store to the process indicates that the entire price list is an input to the selection operation.

To find the atomic weight of an element from a periodic table we can use a data flow diagram. This data flow diagram is represented in *Figure 4d*. Obviously, the properties of chemical elements are constant and not a variable of the program. It is convenient to represent the operation as a simple access of a constant data store object. Such a data store has no inputs.

Take the case that both actors and data stores can be represented as objects. We distinguish them because their behavior and usage is generally different, although in an object-oriented language they might both be implemented as objects. On the other hand, a data store might be implemented **as a file** and an actor as an **external device**. Some data flows are also objects, although in many cases they are pure values, such as integers, which lack individual identity.

An object as **a single value** and **as a data store containing many values** have different views. In *Figure 5*, the customer name selects an account from the bank. The result of this operation is the account object itself, which is then used as a data store in the update operation. A data flow that generates an object used as the target of another operation is represented by a hollow triangle at the end of the data flow. In contrast, the update operation modifies the balance in the account object, as shown by the small arrowhead. The hollow triangle represents a data flow value that subsequently is treated as an object.

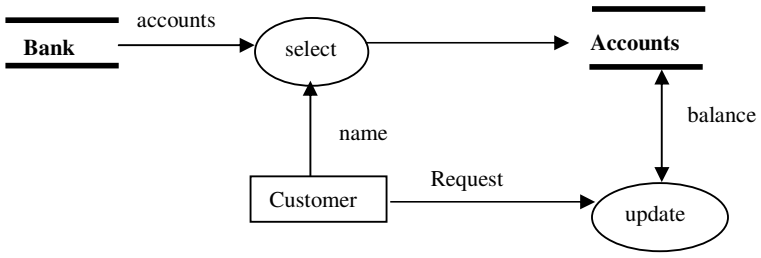


Figure 5: Selection with an object as result

The data flow diagram given in *Figure 6* represents the creation of a new account in a bank. The result of the create account process is a new account, which is stored in the bank. The customer's name and deposit are stored in the account. The account number from the new account is given to the customer. In this example, the account object is viewed both as a **data value** and as a **data store**.

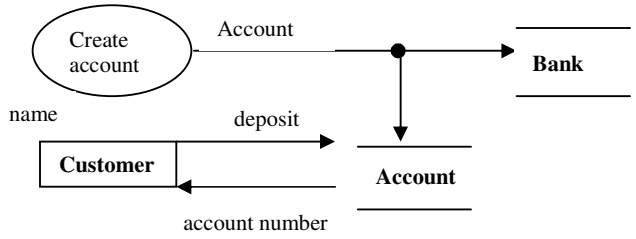


Figure 6: Creation of a new object

3.4.5 Constraints

A constraint shows the relationship between two objects at the same time, or between different values of the some object at different times. For example, in coordinate geometry, the location of a point can be obtained by finding the x and y coordinates, and the scale of these two abscissas can be the same or different.

Constraints can appear in each kind of model. **Object constraints** describe where some objects depend entirely or partially on other objects. **Dynamic constraints** represent relationships among the states or events of different objects. Similarly, the **functional constraints** shows the restrictions on operations, such as the scaling transformation.

A constraint between values of an object over time is often called an invariant. For example, conservation laws in physics are invariants: the total energy, or charge, or angular momentum of a system remains constant. Invariants are useful in **specifying the behavior of operations**.

3.4.6 Control Flows

A control flow is a Boolean value that affects whether a process is evaluated. The control flow is not an input value to the process itself. It is shown by a dotted line from a process producing a Boolean value to the process being controlled. The use of control flow is shown in the *Figure 7*. This figure shows the withdrawal of money from a bank account. The customer enters a password and an account. The withdrawal

was made after successfully verifying the password. The update process also can be explained by using a similar control flow to guard against overdrafts.

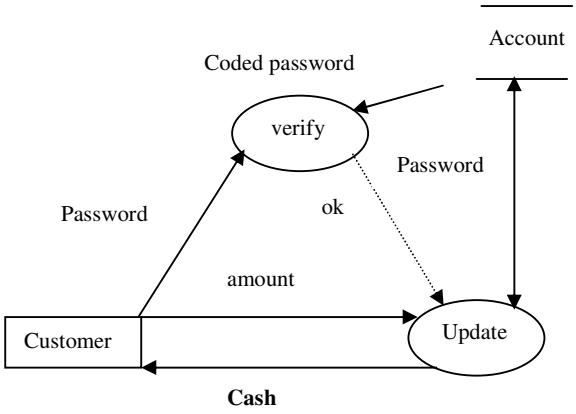


Figure 7: Control flow

☞ Check Your Progress 1

1) Explain Functional Mode with the help of an example.

……..

……..

……..

2) i) What is a Data Flow Diagram?
ii) What is the State Diagram?

…….

…….

…….

…….

3) Explain the following.

i) Process
ii) Data Flows
iii) Actor
iv) Data Stores

……..

…….

……

……..

Now we will discuss some limit actions of DFD.

Although the data flow diagram is very helpful in describing the functional model of an object. But similar to other models such as **object model** and **dynamic model** it also have some limitations.

3.5 DESIGN FLAWS IN DFD

Some of the common limitations of the data flow diagram are as follow:

- The Functional Model does not specify when values are computed.

- The Functional Model does not specify how often values of an object are computed.
- The Functional Model does not specify why the values of an object are changes.
- A Data Flow Diagram contain different symbols to represent different objects and actions, and is very difficult to prepare.
- The Data Flow Diagram for some lengthy problems becomes very complex to prepare as well as to remember.

After a detailed discussion of the various components of functions model, now, you are able to understand the complete sample functional model. In the next section, we describe the functional model for a flight simulator.

3.6 A SAMPLE FUNCTIONAL MODEL

The flight simulator is responsible for handling the pilot input controls, computing the motion of the airplane, computing and displaying the outside view from the cockpit window, and displaying the cockpit gauges. The simulator is intended to be an accurate, but simplified model of flying an airplane, ignoring some of the smaller effects and making some simplifying assumptions. For example, we ignore the rudder, under the assumption that it is held so as to keep the plane pointing in the direction of motion. In *Figure 6,* the **top-level data flow diagram** for the flight simulator is shown. There are two input actors; the **Pilot**, who operates the airplane controls, and the **weather**, which varies according to some specified pattern. There is one output actor the **Screen**, which displays the pilot's view.

There are two read-only data stores: The **Terrain database**, which specifies the geometry of the surrounding terrain as a set of colored polygonal surfaces, and the **Cockpit database**, which specifies the shape and location of the cockpit view port and the locations of the various gauges. There are three internal data stores which are Spatial parameters, which holds the 3-D position, velocity, orientation, and rotation of the plane; **Fuel**, which holds the amount of fuel remaining; and **Weight**, which holds the total weight of the plane. The initialization of the internal data stores is necessary but is not shown on the data flow diagram.

We can divide the processes given in DFD into three kinds, which are **handling controls, motion computation**, and **display generation**. The control handling processes are adjust controls, which transforms the position of the pilot's controls (such as joysticks) into positions of the airplane control surfaces and engine speed; consume fuel, which computes fuel consumption as a function of engine speed; and compute weight, which computes the weight of the airplane as the sum of the base weight and the weight of the remaining fuel. Process adjust controls is expanded in *Figure 7* where it can be seen as comprising three distinct controls; the elevator, the ailerons, and the throttle. There is no need to expand these processes further, as they can be described by input-output functions easily.

The motion computation processes are compute forces, which computes the various forces and torques on the plane and sums them to determine the net acceleration and the rotational torques, and integrate motion, which integrates the differential equations of motion. Process compute forces incorporates both geometrical and aeronautical computations. It is expanded in *Figure 8*. Net force is computed as the vector sum of drag, lift, thrust, and weight.

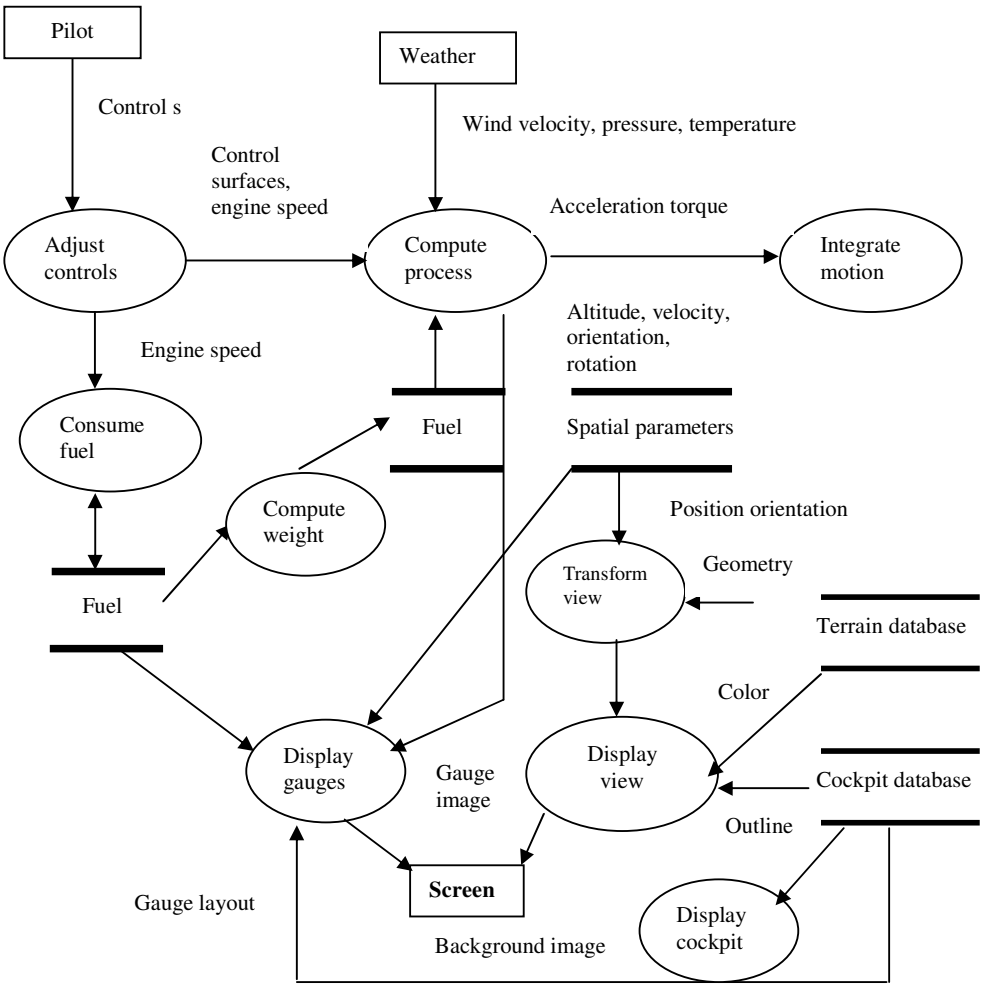


Figure 8: Functional model of flight simulator

These forces in turn depend on intermediate parameters, such as airspeed, angle of attack, and air density. The aerodynamic calculations must be made relative to the air mass, so the wind velocity is subtracted from the plane's velocity to give the airspeed relative to the air mass.

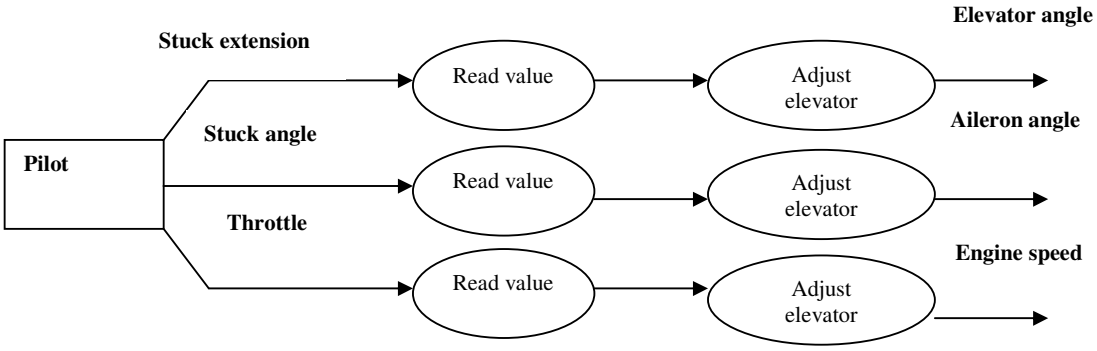


Figure 9: Expansion of adjust controls process

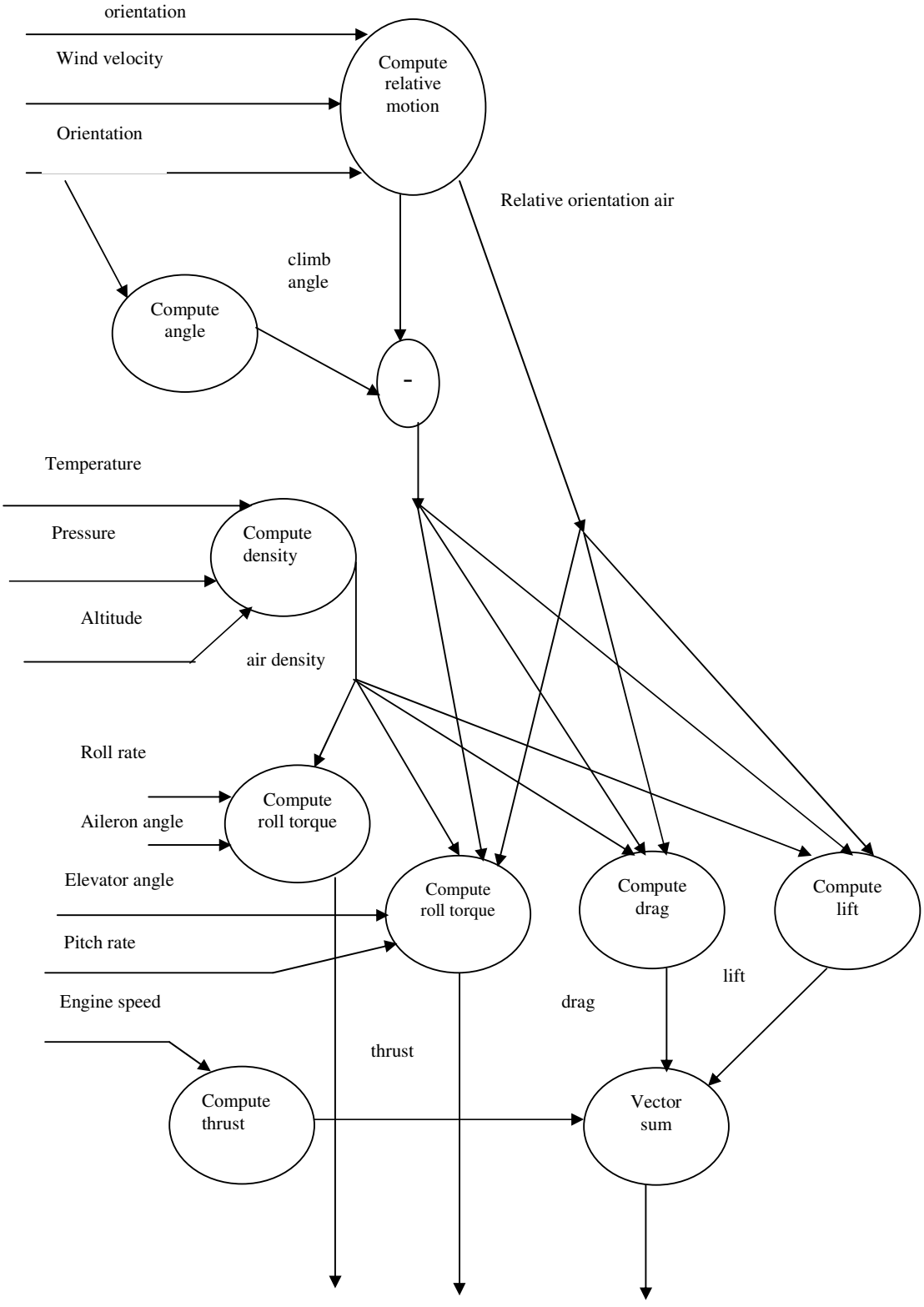
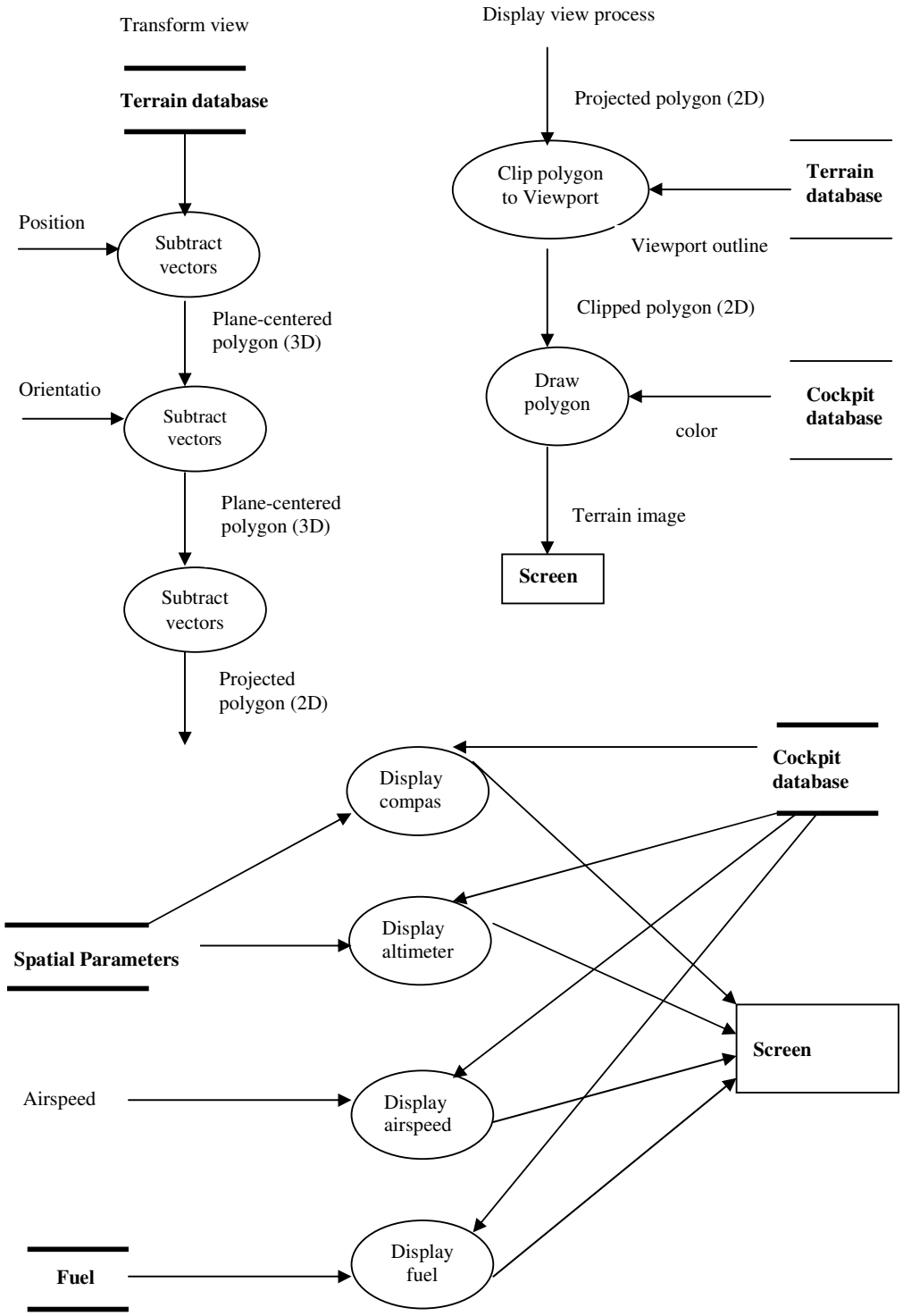


Figure 10: Expansion of compute forces processes

Air density is also computed and used in subsequent processes. The intermediate parameters are computed in terms of data store parameters, such as airplane velocity, orientation, rotation rates, roll rate, pitch rate, and altitude, obtained from spatial parameters; **wind velocity**, **temperature**, and **pressure**, obtained from **Weather**; weight, obtained from **Weight**; and in terms of output data flows from other processes, such as elevator angle, aileron angle, and engine speed, obtained



Display gauges process

Figure 11: Expansion of display processes

from process adjust controls. The internal processes, such as compute drag, compute lift and compute density, would be specified by aeronautical formulas and look-up tables for the specific airplane. For example, computer lift is specified by the equation **L=C (a) SpV^2/2**, where L is lift, a is the angle of attack, S is the wing area, P is the air density, V is the airspeed, and C is the coefficient of lift as a function of angle of attack, specified by a table for the particular kind of wing. Process integrate motion is the solution to the differential equations of motion.

The display processes are transform view, display view, display gauges, and display cockpit. These processes convert the airplane parameters and terrain into a simulated view on the screen. They are expanded on *Figure 9*. Process transform view transforms the coordinates of a set of polygons in the Terrain database into the pilot's coordinate system, by first offsetting them by the plane's position, rotating the polygons by the plane's orientation, and then transforming the polygons' perspective onto the viewing plane to produce a 2-D image in the pilot's eye view. The position and orientation of the plane are input parameters. Process display view clips the image of the transformed polygons to remain within the output of the cockpit view port, whose shape is specified in a Cockpit database. The clipped 2-D polygons are drawn on the screen using the colors specified in the Terrain database. Process display gauges displays various airplane parameters as gauges, using locations specified in the cockpit database. Process display cockpit, displays a fixed image of the stationary parts of the cockpit, and need not be expanded. You must have observed in this example, that the functional model does not specify when, why, and how often values are computed.

☞ Check Your Progress 2

1) Explain the use of constraints in functional model with suitable example.
…….

…….

2) Take any object from your surrounding environment and describe, it by making a complete functional model of it, using data flow diagrams.
…….

…….

3) Prepare a data flow diagram for computing the volume and surface area of a cylinder. Inputs are the height and radius of the cylinder. Outputs are the volume and surface area of the cylinder. Discuss several ways of implementing the data flow diagram.
…….

…….

3.7 RELATION OF FUNCTIONAL TO OBJECT AND DYNAMIC MODEL

Let us now discuss the relationship between the **Object Model**, **Dynamic Model,** and **Functional Model**. The functional model shows what **has to be done** by a system. The leaf processes are the operations on objects. The object model shows the "**doers**" of the objects. Each process is implemented by performing a method on some object. The dynamic model shows **the sequences** in which the operations are performed. Each sequence is implemented as a sequence, loop, or **alternation** of statements within some method. The three models come together in the implementation of methods. The functional model is a guide to the methods.

The processes in the functional model **correspond** to operations in the object model. Often, there is a direct correspondence between each level. A top level process corresponds to an operation on a complex object, and lower level processes correspond to operations on more basic objects that are part of the complex object or that implement it. Sometimes, one process corresponds to several operations, and one operation corresponds to several processes.

Processes in the functional model show objects that are related by function. Often, one of the inputs to a process can be identified as the target object, with the rest being

parameters to the operations. The target object is a client of the other objects because it uses them in performing the operations. The target knows about the clients, but the clients do not necessarily know about the target. The target object class is dependent on the argument classes for its operations. The client-supplier relationship establishes implementation dependencies among classes; the clients are implemented in terms of suppliers class, and are therefore dependent on the supplier classes.

A process is usually implemented as a method. If the same class of object is an input and an output, then the object is usually the target, and the other inputs are arguments. If the output of the process is a data store, the data store is the target. If an input of the process is a data store, the data store is the target. Frequently, a process with an input **from** or output **to** a data store corresponds to two methods, one of them being an **implicit selection** or **update** of the data store. If an input or output is **an actor**, then it is the target. If an input is an object and an output is **a part** of the object or a neighbor of the object in the object model, then the object is the target. If an output object is created out of input parts, then the process represents a class method. If none of these rules apply, then the target is often implicit and is not one of the inputs or outputs. Often the target of a process is the target of the entire subdiagram. For example, in *Figure10* the target of compute forces is actually the airplane itself. Data stores weight and spatial parameters are simply components of the airplane that are accessed during the process.

Actors are explicit objects in the object model. Data flows to or from actors represent operations on or by the objects. The data flow values are the arguments or results of the operations. Because actors are self-motivated objects, the functional model is not sufficient to indicate when they act. The dynamic model for an actor object specifies when it acts.

Data stores are also objects in the object model, or at least fragments of objects, such as attributes. Each flow into a data store is an update operation. Each flow out of a data store is a query operation, with no side effects on the data store object. Data stores are passive objects that respond to queries and updates, so the dynamic model of the data store **is irrelevant to its behavior**. A dynamic model of the actors in a diagram is necessary to determine the order of operations.

Data flows are values in the object model. Many data flows are simply pure values, such as numbers, strings, or lists of pure values. Pure values can be **modeled as classes** and implemented as objects in most languages. A pure value is not a container whose value can change, but just the value itself. A pure value, therefore, has no state and no dynamic model. Operations on pure values yield other pure values and have no side effects. Arithmetic operations are examples of such operations.

Relative to the Functional Model: The **object model** shows the structure of the actors, data stores, and flows in the functional model. The **dynamic model** shows the sequence in which processes are performed.

Relative to the Object Model: The **functional model** shows the operations on the classes, and the arguments of each operation as well. The **dynamic model shows** the status of each object and the operations that are performed as it receives events and changes state.

Relative to the Dynamic Model: The **functional model** shows the definitions of the leaf actions and activities that are undefined with the **dynamic mode**. The object model shows changes of state during the operation.

Operations can be specified by a variety of means, including mathematical equations, tables, and constraints between the inputs and outputs. An operation can be specified by pseudopodia, but a specification does not imply a particular implementation; it may be implemented by a different algorithm that yields equivalent results. Operations have signatures that specify their external interface and transformations that specify their effects. Queries are operations without side effects; they can be implemented as pure functions. Actions are operations with side effects and duration; they must be

implemented as tasks. Operations can be attached to classes within the object model and implemented as methods. Constraints specify additional relationships that must be maintained between values in the object model.

☞ Check Your Progress 3

1) Describe the meaning of the data flow diagram in *Figure 12*.

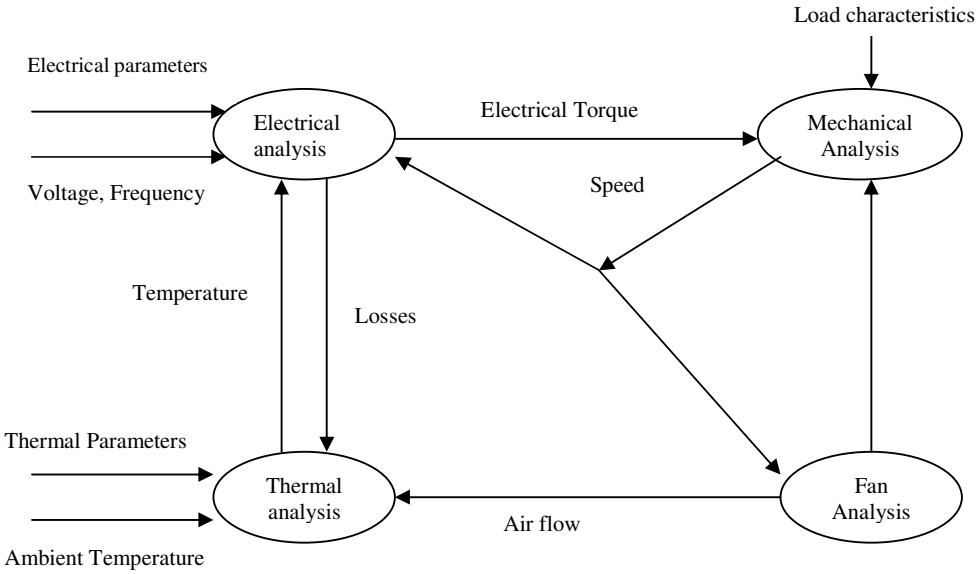


Figure 12: Data flow diagram of motor analysis

……..

……..

2) Prepare a data flow diagram for computing the mean of a sequence of input values. A separate control input is provided to reset the computation. Each time a new value is input, the mean of all values input since the last reset command should be output. Since you have no way of knowing how many values will be processed between resets, the amount of data storage that you use should not depend on the number of input values. Detail your diagram down to the level of multiplications, divisions, and additions.

……..

……..……

3) Using the quadratic formula as a starting point, prepare a data flow diagram for computing the roots of the quadratic equation $ax^2 + bx + c = 0$. Real numbers a, b and c are inputs. Outputs are values of X= R1 and X = R2, which satisfy the equation. Remember. R1 and R2 may be real or complex, depending on the values of a, b, and c. The quadratic formula for R1 and R2 is $(-b \pm \text{SQRT}(b^2 \pm 4ac)) / (2a)$.

……..……..…..

…….

3.8 SUMMARY

The functional model shows a computation and the functional derivation of the data values in it without indicating how, when, or why the values are computed. The dynamic model controls which operations are performed and the order in which they are applied. The object model defines the structure of values that the operations

operate on. For batch-like computations, such as compilers or numerical computations, the functional model is the primary model, but in large systems all three models are important.

Data flow diagrams show the relationship between values in a computation. A data flow diagram is a graph of process, data flows, data stores, and actors. Processes transform data values. Low-level processes are simple operations on single objects, but higher-level processes can contain internal data stores subject to side effects. A data flow diagram is a process. Data flows relate values on processes, data stores, and actors. Actors are independent objects that produce and consume values. Data stores are passive objects that break the flow of control by introducing delays between the creation and the use of data.

The object model, dynamic model, and functional model all involve the same concepts, namely, data, sequencing, and operations, but each model focuses on a particular aspect and leaves the other aspects uninterrupted. All three models are necessary for a full understanding of a problem, although the balance of importance among the models varies according to the kind of application.

3.9 SOLUTIONS/ ANSWERS

Check Your Progress 1

1) **Functional Model**

The functional model shows a computation and the functional derivation of the data values in it without indicating how, when, or why the values are compounded. For example a spreadsheet is a type of functional model. The values in a spreadsheet can be calculated by using some formula, but it can not be structured further.

2 i) **Data Flow Diagram**: A data flow diagram is a graph which shows the flow of data values from their sources in objects through processes that transform them to their destinations in other objects. It does not show how the values are controlled during computation. The Data Flow Diagram shows the functional relationship of the values computed by a system. DFD contains *processes* that transform data, *data flows* that move data, *actor* objects that produce and consume data, and *data store* objects that store data passively.

ii) **State Diagram:** An object can receive a sequence of input instructions. The state of an object can vary depending upon the sequence of input instructions. If we draw a diagram which will represent all the processes (input) and their output (states) then that diagram is known as state diagram. Processes are represented by the arrow symbol and states by an oval symbol. For example, the screen of an ATM machine has many states like main screen state, request password state, process transaction state.

Short Note

3i) **Process:** A process transforms data values. It is represented as an ellipse containing a description of the transformation. Each process has a fixed number of input and output data arrows, each of which carries a value of a given type.

ii) **Data Flows:** A data flow connects the output of an object or process to the input of another object or process. It represents an intermediate data value within a computation. Data flow specifies direction of flow of data from source objects to the destination object.

iii) **Actors:** An actor is an active object that drives the data flow graph by producing or consuming values. Actors are attached to the inputs and outputs of a data flow graph.

iv) **Data Stores:** A data store is a passive object within a data flow diagram that stores data for later access. A data store allows values to be accessed in a different order than they are generated.

Check Your Progress 2

1) A constraint shows the relationship between two objects at the same time, or between different values of the same object at different times. A constraint may be expressed as a total function or as a partial function. For example, a coordinate transformation might specify that the scale factor for the x-coordinate and the y-coordinate will be equal; this constraint totally defines one value in terms of the other.

2) Give a functional model for your example system with the help of section 9.6 of this Unit.

3) The data flow diagram for computing the volume and surface area of a cylinder is given in *Figure13* below.

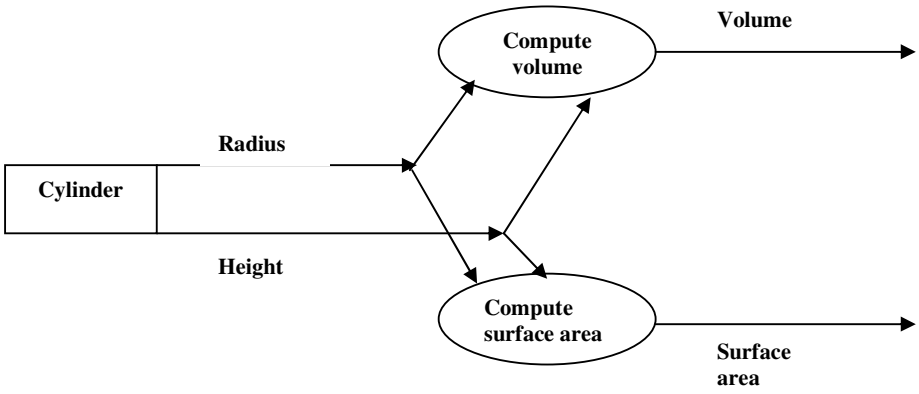


Figure 13: Data flow diagram for computing volume and surface area of a cylinder.

In this figure, the object cylinder is represented by a rectangle. The processes compute volume and surface are by ellipse, and the data flow by the arrow. The formula for volume and surface area of represented cylinder can be taken respectively.

Check Your Progress 3

1) The data flow diagram shows the relationship between values in a computation. The flow of electrical parameters is shown by the arrow symbol. In this DFD, there are **four processes**. These are **electrical analysis**, **mechanical analysis**, **fan analysis**, and **thermal analysis**. These four processes compute the various electrical parameters that are required for the safe running of an electrical motor.

The input electrical parameters are checked by the electrical analysis process. After proper verification, the mechanical movement (electrical torque) is measured by the mechanical analysis process. The characteristics of fan are computed by fan analysis process. The temperature of the motor is computed by the thermal analysis process. The flow of data from source to destination for this DFD is given by

Electrical analysis – mechanical analysis – fan analysis–thermal analysis, and *vice versa*.

2) *Figure 14* (a) or (b) below shows the data flow diagram for computing the mean for a sequence of input values.

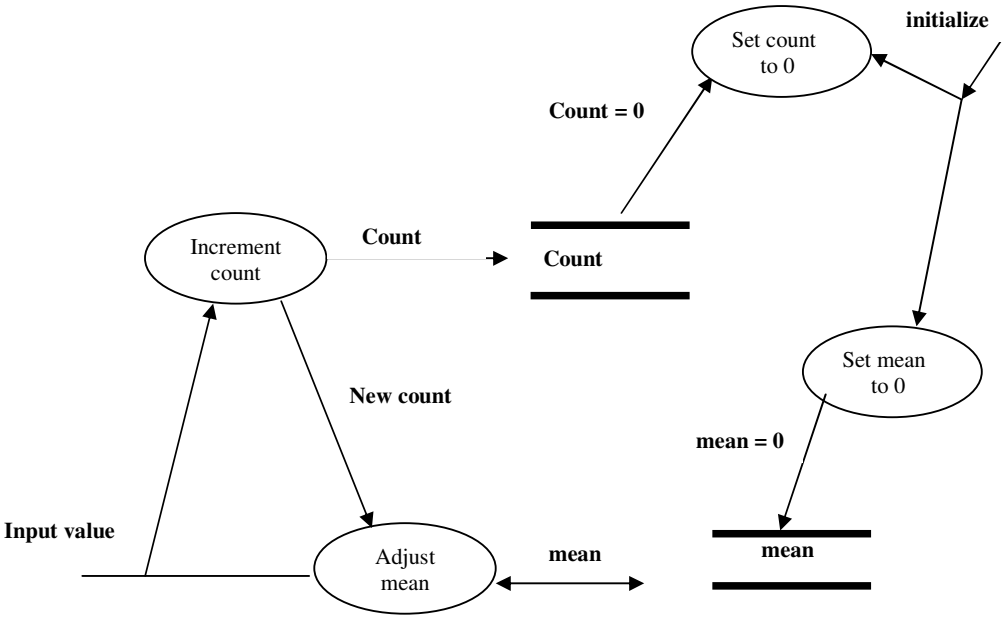
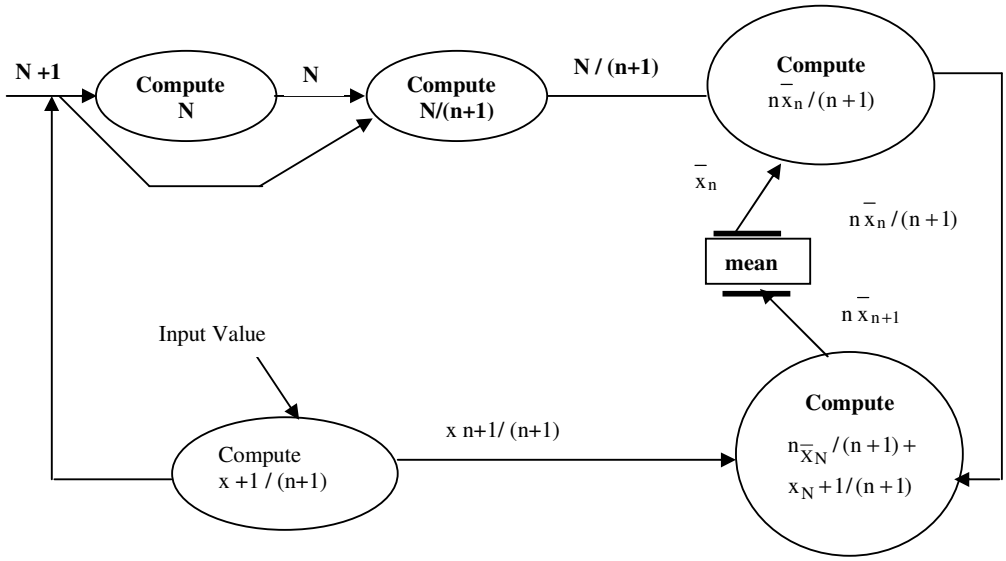


Figure 14 (a): DFD for computing mean



(Note: n+1 = new count, x_n = n th value, $\bar{x}_n$ = average values)

Figure 14 (b): Data flow diagram for computing mean of a sequence of values

The above data flow diagram computes the mean of a sequence of input values. This DFD is the required solution of the problem.

3) The Data Flow Diagram for computing the roots of the quadratic equation is given by the following diagram.

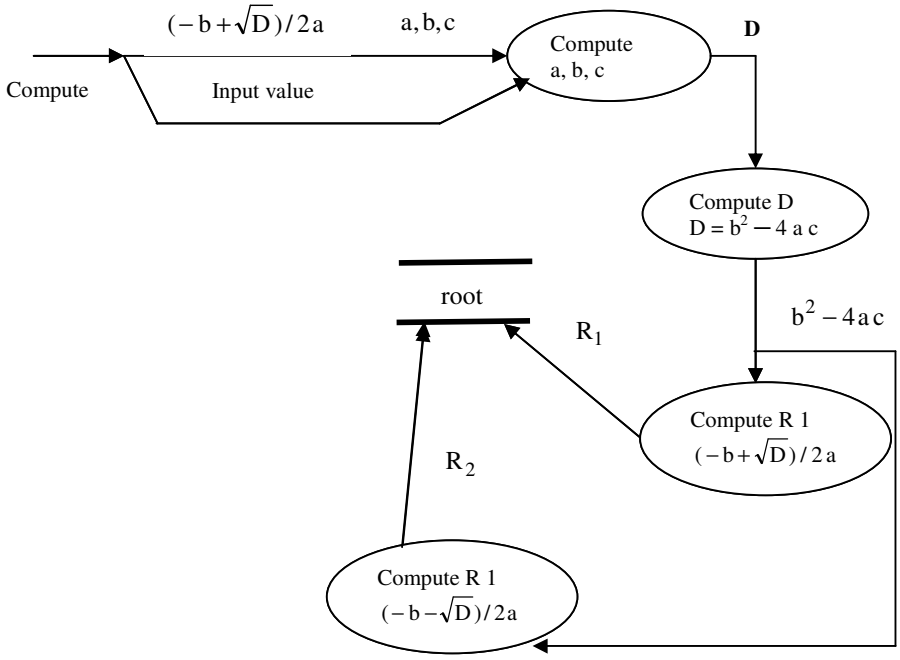


Figure 15: Data Flow Diagram for computing the roots of quadratic equation