STATE CHART DIAGRAM

A **state diagram** is used to represent the condition of the system or part of the system at finite instances of time. It's a **behavioral** diagram and it represents the behavior using finite state transitions. State diagrams are also referred to as **State machines** and **State-chart Diagrams**. These terms are often used interchangeably. So simply, a state diagram is used to model the dynamic behavior of a class in response to time and changing external stimuli. We can say that each and every class has a state but we don't model every class using State diagrams. We prefer to model the states with three or more states.

Uses of state chart diagram –

- We use it to state the events responsible for change in state (we do not show whatprocesses cause those events).
- We use it to model the dynamic behavior of the system .
- To understand the reaction of objects/classes to internal or external stimuli. Firstly let us understand what are **Behavior diagrams**?

There are two types of diagrams in UML :

1. **Structure Diagrams –** Used to model the static structure of a system, for example- class diagram, package diagram, object diagram, deployment diagram etc.
2. **Behavior diagram –** Used to model the dynamic change in the system over time. They are used to model and construct the functionality of a system. So, a behavior diagram simply guides us through the functionality of the system using Use case diagrams, Interaction diagrams, Activity diagrams and State diagrams.

Basic components of a statechart diagram –

1. **Initial state –** We use a black filled circle represent the initial state of a System or aclass.

Figure – initial state notation

2. **Transition –** We use a solid arrow to represent the transition or change of control from one state to another. The arrow is labelled with the event which causes the change instate.

Figure – transition

3. **State –** We use a rounded rectangle to represent a state. A state represents the conditions or circumstances of an object of a class at an instant of time.

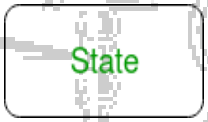


Figure – state notation

4. **Fork –** We use a rounded solid rectangular bar to represent a Fork notation with incoming arrow from the parent state and outgoing arrows towards the newly created states. We use the fork notation to represent a state splitting into two or more concurrentstates.

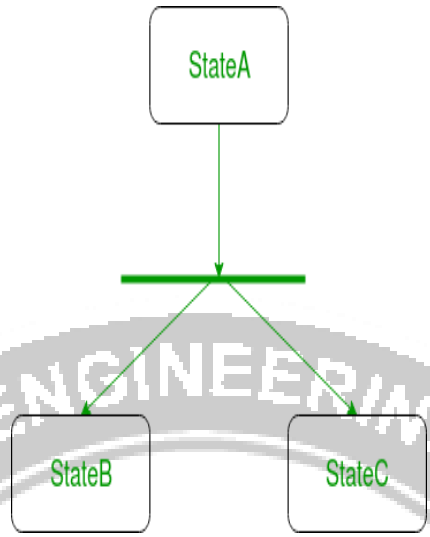


Figure – a diagram using the fork notation

5. **Join** – We use a rounded solid rectangular bar to represent a Join notation with incoming arrows from the joining states and outgoing arrow towards the common goal

state. We use the join notation when two or more states concurrently converge into one onthe occurrence of an event or events.

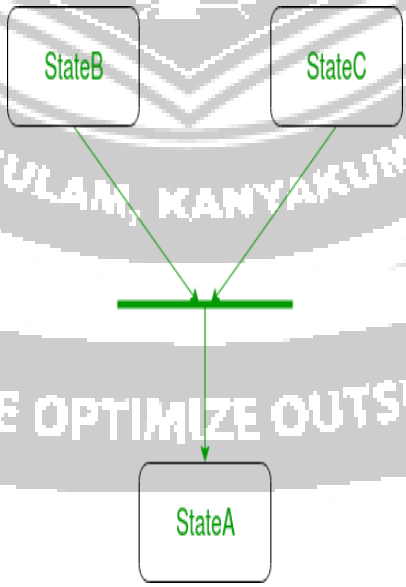


Figure – a diagram using join notation

6. **Self transition** – We use a solid arrow pointing back to the state itself to represent aself transition. There might be scenarios when the state of the

object does not change upon the occurrence of an event. We use self transitions to represent such cases.

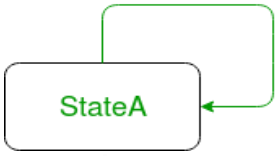


Figure – self transition notation

7. **Composite state –** We use a rounded rectangle to represent a composite state also.We represent a state with internal activities using a composite state.

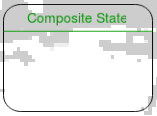


Figure – a state with internal activities

8. **Final state –** We use a filled circle within a circle notation to represent the final statein a state machine diagram.

Figure – final state notation

Steps to draw a state diagram –

1. Identify the initial state and the final terminating states.
2. Identify the possible states in which the object can exist (boundary values corresponding to different attributes guide us in identifying different states).
3. Label the events which trigger these transitions.

Example – state diagram for an online order –

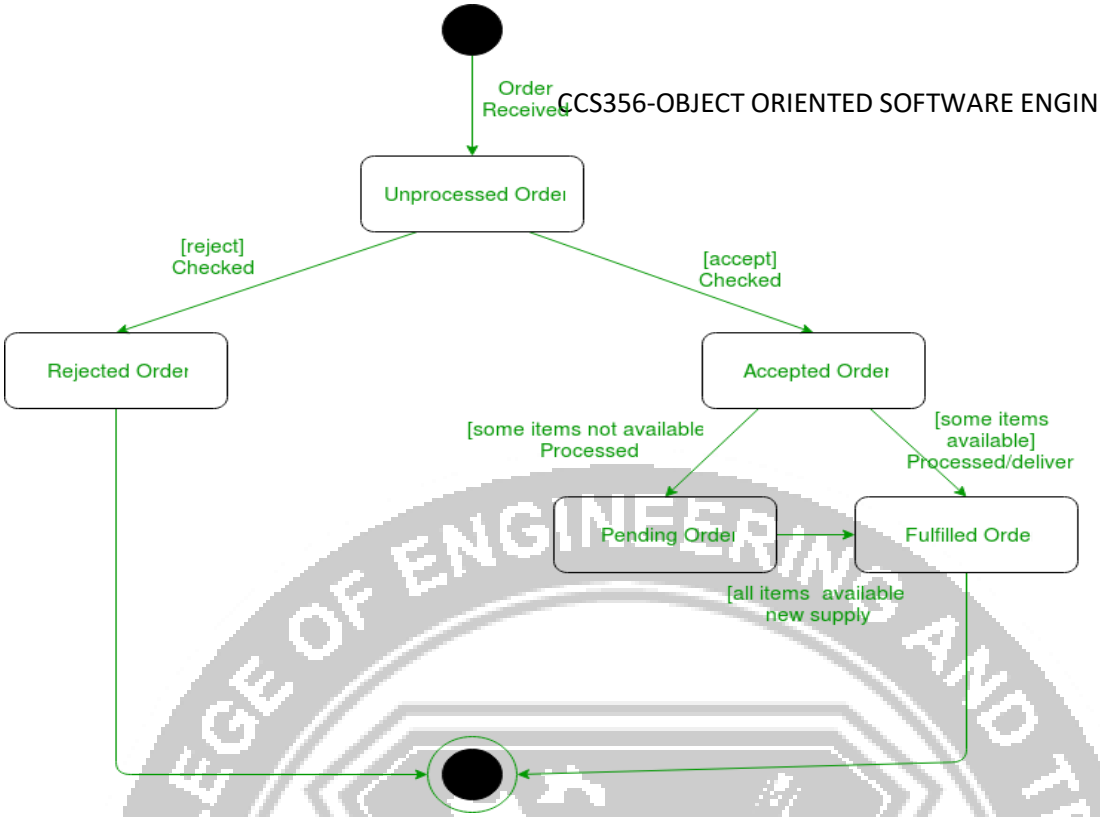


Figure – state diagram for an online order

The UMl diagrams we draw depend on the system we aim to represent. Here is just anexample of how an online ordering system might look like :

1. On the event of an order being received, we transit from our initial state toUnprocessed order state.
2. The unprocessed order is then checked.
3. If the order is rejected, we transit to the Rejected Order state.
4. If the order is accepted and we have the items available we transit to the fulfilledorder state.
5. However if the items are not available we transit to the Pending Order state.

Functional Modelling

Functional Modelling gives the process perspective of the object-oriented analysis model and an overview of what the system is supposed to do. It defines the function of the internal processes in the system with the aid of Data Flow Diagrams (DFDs). It depicts the functional derivation of the data values without indicating how they are derived when they are computed, or why they need to be computed.

DATA FLOW DIAGRAMS

Functional Modelling is represented through a hierarchy of DFDs. The DFD is a graphical representation of a system that shows the inputs to the system, the processing upon the inputs, the outputs of the system as well as the internal data stores. DFDs illustrate the series of transformations or computations performed on the objects or the system, and the external controls and objects that affect the transformation.

Rumbaugh et al. have defined DFD as, "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 on other objects."

The four main parts of a DFD are −

- Processes,
- Data Flows,
- Actors, and
- Data Stores.

The other parts of a DFD are −

- Constraints, and
- Control Flows.

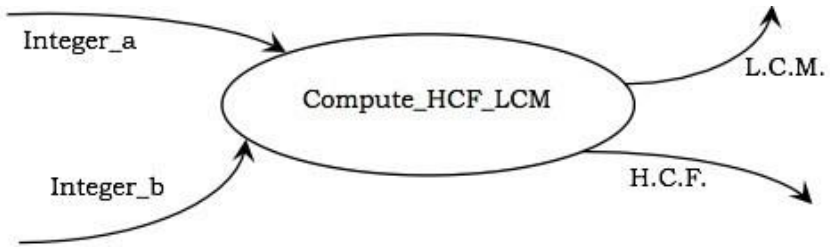
Features of a

DFDProcesses

Processes are the computational activities that transform data values. A whole system can be
visualized as a high-level process. A process may be further divided into smaller components.The lowest-level process may be a simple function.

Representation in DFD − A process is represented as an ellipse with its name written inside itand contains a fixed number of input and output data values.

Example − The following figure shows a process Compute_HCF_LCM that accepts twointegers as inputs and outputs their HCF (highest common factor) and LCM (least common multiple).



Data Flows

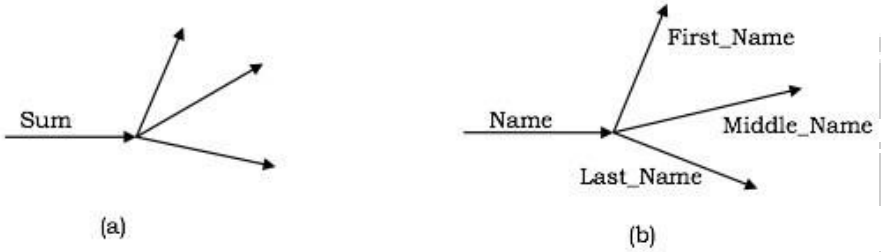
Data flow represents the flow of data between two processes. It could be between an actor and a process, or between a data store and a process. A data flow denotes the value of a data item atsome point of the computation. This value is not changed by the data flow.

Representation in DFD − A data flow is represented by a directed arc or an arrow, labelled with the name of the data item that it carries.

In the above figure, Integer_a and Integer_b represent the input data flows to the process, while
L.C.M. and H.C.F. are the output data flows.

A data flow may be forked in the following cases −

- The output value is sent to several places as shown in the following figure. Here, the output arrows are unlabelled as they denote the same value.
- The data flow contains an aggregate value, and each of the components is sent to different places as shown in the following figure. Here, each of the forked componentsis labelled.



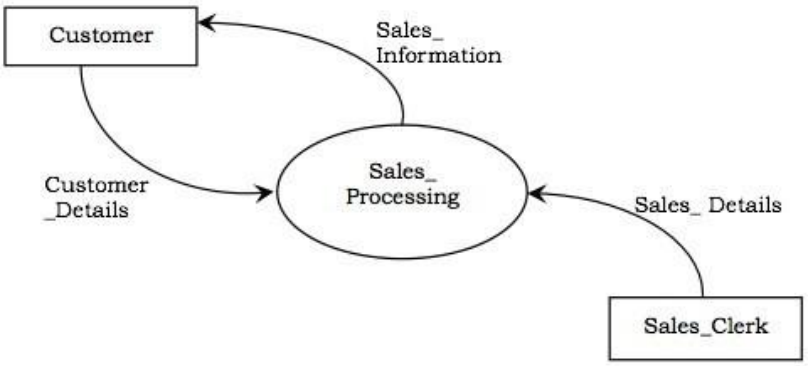
Actors

Actors are the active objects that interact with the system by either producing data and inputting them to the system, or consuming data produced by the system. In other words, actorsserve as the sources and the sinks of data.

Representation in DFD − An actor is represented by a rectangle. Actors are connected to theinputs and outputs and lie on the boundary of the DFD.

Example − The following figure shows the actors, namely, Customer and Sales_Clerk in acounter sales system.

ROHINI COLLEGE OF ENGINEERING AND TECHNOLOGY
PALKULAM, KANYAKUMARI
OBSERVE OPTIMIZE OUTSPREAD

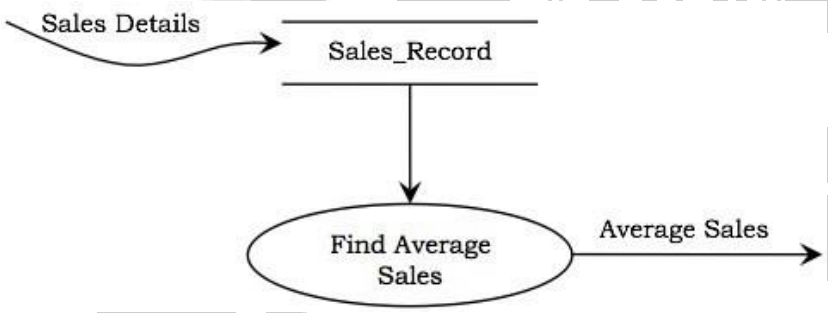


Data Stores

Data stores are the passive objects that act as a repository of data. Unlike actors, they cannot perform any operations. They are used to store data and retrieve the stored data. They representa data structure, a disk file, or a table in a database.

Representation in DFD − A data store is represented by two parallel lines containing the name of the data store. Each data store is connected to at least one process. Input arrows contain information to modify the contents of the data store, while output arrows contain information retrieved from the data store. When a part of the information is to be retrieved, the output arrowis labelled. An unlabelled arrow denotes full data retrieval. A two-way arrow implies both retrieval and update.

Example − The following figure shows a data store, Sales_Record, that stores the details of all sales. Input to the data store comprises of details of sales such as item, billing amount, date, etc. To find the average sales, the process retrieves the sales records and computes the average.



Constraints

Constraints specify the conditions or restrictions that need to be satisfied over time. They allowadding new rules or modifying existing ones. Constraints can appear in all the three

models of object-oriented analysis.

- In Object Modelling, the constraints define the relationship between objects. They may also define the relationship between the different values that an object may take at different times.
- In Dynamic Modelling, the constraints define the relationship between the states and events of different objects.
- In Functional Modelling, the constraints define the restrictions on the transformations and computations.

Advantages and Disadvantages of DFD

Advantages	Disadvantages
DFDs depict the boundaries of a system and hence are helpful in portraying the relationship between the external objects and the processes within the system.	DFDs take a long time to create, which may not be feasible for practical purposes.
They help the users to have a knowledge about the system.	DFDs do not provide any information about the time-dependent behavior, i.e., they do not specify when the transformations are done.
The graphical representation serves as a blueprint for the programmers to develop a system.	They do not throw any light on the frequency of computations or the reasons for computations.
DFDs provide detailed information about the system processes.	The preparation of DFDs is a complex process that needs considerable expertise. Also, it is difficult for a non-technical person to understand.
They are used as a part of the system documentation.	The method of preparation is subjective and leaves ample scope to be imprecise.

FUNCTIONAL MODELS

The Object Model, the Dynamic Model, and the Functional Model are complementary to eachother for a complete Object-Oriented Analysis.

- Object modelling develops the static structure of the software system in terms of objects. Thus it shows the "doers" of a system.
- Dynamic Modelling develops the temporal behavior of the objects in response to external events. It shows the sequences of operations performed on the objects.
- Functional model gives an overview of what the system should do.

Functional Model and Object Model

The four main parts of a Functional Model in terms of object model are −

- **Process** − Processes imply the methods of the objects that need to be implemented.
- **Actors** − Actors are the objects in the object model.
- **Data Stores** − These are either objects in the object model or attributes of objects.
- **Data Flows** − Data flows to or from actors represent operations on or by objects. Dataflows to or from data stores represent queries or updates.

Functional Model and Dynamic Model

The dynamic model states when the operations are performed, while the functional model states how they are performed and which arguments are needed. As actors are active objects, the dynamic model has to specify when it acts. The data stores are passive objects and they only respond to updates and queries; therefore, the dynamic model need not specify when they act.

Object Model and Dynamic Model

The dynamic model shows the status of the objects and the operations performed on the occurrences of events and the subsequent changes in states. The state of the object as a result ofthe changes is shown in the object model.

ROHINI COLLEGE OF ENGINEERING AND TECHNOLOGY
PALKULAM, KANYAKUMARI
OBSERVE OPTIMIZE OUTSPREAD