**Module-2**

**Chapter-5**

**Dynamic Modeling**

* Temporal relationships are difficult to understand
* A system can be well understood by first examining its static behavior
* Next we examine the changes to the objects and relationship over time
* Aspects of system that are concerned with time and changes are called dynamic model

**5.1Events and states**

**5.1.1 Events**

- Indiviual stimulus from one object to another is an event

- Occurrence at any point in time

- Eg: power turned on, alarm set

- Has no duration

- There may or may not be any relation in between two events

- Two events that are unrelated are said to be concurrent. They have no effect on each other

- Eg: flight 123 may depart before or after flight 456 departs ROME

- Concurrent events can occur in any order

- Event is one way transmission of information from one object to another

- Every event is a unique occurrence but the events are grouped into event classes and give event class a name to indicate common structure and behavior

- Eg: Fight 123 departs from Chicago and flight 456 departs from Rome. Both are instances of class airplane flight departs

- Most event classes has attributes

- Eg: airplane flight departs has attributes flight number, city

Airplane flight departs (flight number, city)

Mouse button pushed (button, location)

Digit dialed (digit)

**Fig 5.1 Event classes and attributes**

* Some classes of events may be simply signals while other classes convey data values
* Data values conveyed by an event is called attributes
* Attributes are shown in parentheses after event class name

**5.1.2 Scenario and Event traces**

- Scenario is a sequence of events that occur during one particular execution of system

Caller lifts receiver

Dial tone begins

Caller dials digit

Called phone being ringing

Ringing tone appears in calling phone

Called party answers

Phone connected

Called party hangs up

Phones disconnected

Caller hangs up

**Fig 5.2 Scenario**

* The scope of scenario can vary
* It can include all events in the system or it may include only those events generated by certain objects in the system
* Scenario can be historical record of executing a system
* Sequence of events and objects exchanging events can be shown in an augmented scenario called event trace
* Object is shown as vertical line and each event as horizontal arrow from sender object to receiver object

caller lifts receiver

Caller Phone line Callee

dial tone begins

dials (5)

dial tone ends

dials (5)

dials (1)

ringing tone phone rings

tone stops ringing stops

answers phone

phones connected phones connected

callee hangs up

connection broken connection broken

caller hangs up

**Fig 5.3 Event trace for phone call**

**5.1.3 States**

- State is the present condition

- State is an abstraction of attribute values and links of an object

- The set of values are grouped together into a state according to the properties that affect the behavior of object

- Eg: State of bank is solvent or insolvent depending on whether its assets exceed its liabilities

- State specifies the response of object to input events

- The response of an object to an event may include an action or change of state by the object

- Eg: After the caller lifts the receiver and before entering the first digit the phone is in the state dial tone. After the digits are dialed the phone is in the state dialing

- State has duration

- A state corresponds to the interval between two events received by an object

- Events represent points in time and state represents interval of time

- State is associated with continuous activity

**5.1.4 State Diagrams**

- State diagram relates events and states

- Change of state caused by an event is called transition

- State diagram is a graph whose node represents state and arc represents transition

- State is drawn as rounded box containing an optional name

- Transition is drawn as an arrow from receiving state to target state

- Label on arrow is the name of event causing transition

on-hook

on-hook

Idle

Dial tone

Dialing

Ringing

Disconnected

Connecting

Connected

Time out

Recorded msg

Busy tone

off-hook

timeout

digit

no. busy

valid no

invalid no

routed

msg done

called phone answers

called phone hangs up

**Fig 5.4 State diagram of telephone**

* State diagram describes the behavior of single class of object
* State diagram can represent one shot life cycle or continuous loop
* One shot life cycle has initial and final state
* Initial state is represented using solid circle
* Final state is represented using bulls eye

Black wins

checkmate

White wins

Start

stalemate

Draw

stalemate

white moves

black moves

White wins

Black wins

checkmate

**Fig 5.5 One shot state diagram for chess game**

**5.1.5 Conditions**

- Boolean function of object values

- Eg: Temperature is below freezing from Nov to March

- Here the condition is temperature below freezing

- Has duration

- Conditions can be used as guards on transitions

- A guarded transition fires when its event occur only if condition is true



**Fig 5.6Representation of Condition**

**5.2 Operations**

How to trigger operations?

**5.2.1 Controlling operations**

- State diagrams could be little use if they just described the pattern of events

- A behavior description of an object must specify what the object does in response to events

- Operation attached to states or transitions are performed in response to corresponding states or events

- An activity is an operation that takes time to complete

- An activity is associated with a state

- Activities include continuous as well as sequential events

- do:activity within the state box indicates that activity starts an entry to state and stops on

exit

* If an event causes a transition from the state before the activity is complete, it terminates prematurely
* An action is an instantaneous operation and is associated with an event
* Notation for action is ‘ / ‘

**5.2.2 Summary of notation for state diagram with operation**

event1(attr)[condition]/action1

State2

…………

State 1

do:activity 1

**Fig 5.7 Notation for state diagram with operation**

* State name is written in boldface within rounded box
* Event name is written on transition arrow and may optionally followed by one or more attributes within parentheses
* Condition may be listed within square brackets
* An activity is indicated within a state box by the keyword “do” followed by name
* An action is indicated using ‘ / ‘

**5.3 Nested state diagram**

- State diagrams can be structured to permit concise description of complex systems

- The way of structuring state machines are similar to the way of structuring object

- Generalization allows states and events to be arranged into generalization hierarchies with inheritance of common structure and behavior

- Aggregation allows a state to be broken into orthogonal components

**5.3.1 Problems with flat state diagrams**

- State diagrams are criticized because they lack expressive power and are impractical for large problems

- Consider an object with n independent Boolean attributes that affect control

- Representing such object with flat state diagram require 2n states

- By partitioning state into n independent state machines only 2n states are required

**Fig 5.8 Combinatorial explosion of transitions in flat state diagram**

**5.3.2 Nesting state diagrams**

- An activity in a state can be expanded as lower level state diagram, each state

representing one step of activity

- Nested activities are one shot state diagrams with input and output transitions

- The set of nested state diagrams form a lattice

- The lack of event label indicates that transition fires automatically when activity in state is complete

**5.3.3 State generalization**

- Nested state diagram is a form of generalization on states

- Generalization is or-relationship

- An object in a state in high level diagram must be exactly one state in nested diagram

- It must be in first state or second state or in one of other state

- The states in nested diagram are all refinements of state in high level diagram

- States may have substates that inherit the transitions of superstates

- Fig 5.9 shows state diagram for car transmission

- The transmission can be reverse, neutral or forward

- If it is in forward; it can be in first, second or third gear

- States first, second and third are substates of state Forward

- The superstate is drawn as a large rounded box enclosing all its substates

- Substates can enclose further substates

- The transitions of superstate are inherited by each of its substate

**Transmission**

push R

- Selecting N in any forward gear causes transition to neutral

push N

Reverse

Neutral

- Selecting F in neutral causes transition to forward

push F

push N

**Forward**

upshift

upshift

stop

5.3.3 State generalization

downshift

downshift

downshift

**Fig 5.9 State diagram of car transmission with generalization**

**5.3.4 Event generalization**

- Events can be organized into generalization hierarchy with inheritance of event attributes

- Fig 5.10 shows event hierarchy for keyboard events

- Events mouse button down and keyboard character are two kinds of user input

- Both events inherit attribute time and event event and attribute device from event user

input

- Mouse button down and mouse button up inherit location from mouse button

- Keyboard characters can be divided into control characters and graphic characters

- Providing an event hierarchy permits different levels of abstraction to be used at

different places in model

event

time

user input

device

mouse button

keyboard character

location

character

mouse button down

mouse button up

control

graphic

space

alphanumeric

punctuation

**Fig 5.10 Partial event hierarchy for keyboard events**

5**5.4 Concurrency**

**5.4.1 Aggregation concurrency**

- A dynamic model describes a set of concurrent objects each with its own state and state diagram

- Objects in the system are inherently concurrent and can change state independently

- State of entire system cannot be represented by a single state in a single object

- State diagram for an assembly is a collection of state diagrams, one for each component

- Aggregation implies concurrency

- Aggregation state corresponds to combined state of all the component diagram

- Aggregation is the and relationship

Car

Ignition

Transmisssion

Brake

Accelerator

On

release key

**Ignition**

Off

turn key to start

Starting

turn key off

Neutral

Reverse

push R

push N

push N

push F

stop

upshift

upshift

**Forward**

**Transmission**

Off

On

depress accelerator

release accelerator

**Accelerator**

Off

On

depress brake

release brake

Brake

**Fig 5.11 Aggregation and its concurrent state diagrams**

**5.4.2 Concurrency within an object**

- Concurrency within the state of single object arises when object can be partitioned into subset of attributes, each of which has its own subdiagram

- Subdiagram need not be independent, the same event can cause transition in more than one subdiagram

- The name of overall composite state can be written in a separate region of box, represented by solid line from the concurrent subdiagram

Playing rubber

N-S game

N-S wins rubber

**N-S vulnerability**

N-S game

Not vulnerable

Vulnerable

E-W game

E-W wins rubber

**E-W vulnerability**

E-W game

Vulnerable

Not vulnerable

**Fig 5.12 Bridge game with concurrent states**

**5.5 Advanced dynamic modeling concepts**

**5.5.1 Entry and Exit action**

- Actions can be associated with entering or exiting a state

**-** Fig 5.13 Shows the control of a garage door opener

- The user generates depress events with a push button to open and close the door

- The control are motor up and motor down

- The events are door open and door close

- Entry and exit action are useful in nested state diagram

Opening

Closed

Open

Closing

door open/motor off

depress/motor up

depress/motor down

depress/motor up

door closed/motor off

**Fig 5.13 Actions on transitions**

**Opening**

entry/motor up

depress

door open

Closed

entry/motor off

**Open**

entry/motor off

depress

**Closing**

entry/motor down

door closed

depress

**Fig 5.14 Actions on entry to states**

**5.5.2 Internal actions**

- An event can cause an action to be performed without causing a state change

- The event name is written inside the state box followed by a **/** and name of action

**State 1**

**do:activity1**

**entry/action2**

**exit/action3**

**event/action4**

**State 2**

**…**

**Object class**

event1 (attribs1) [condition 1]/action1

event2 (attribs2)

**Fig 5.15 Summary of extended notation for state diagrams**

**5.5.3 Automatic transitions**

- When an activity completes, transition to another state fires

- An arrow without an event name indicates automatic transition

- Unlabeled transitions are called lambda transitions

- If a state has one or more automatic transitions but none of the guard conditions are satisfied then the state remains active until one of the conditions is satisfied or until an event causes another transition to fire

**5.5.4 Sending events**

- An object can perform the action of sending an event to another object

- The action “send E(attributes)” sends event E with given attributes to objects

- Eg: The phone line sends a connect(ph no) event to switcher when complete phone number has been dialed

- If a state can accept events from more than one object the order in which concurrent events are received may affect the final state. This is called race condition

**5.5.5 Synchronization of concurrent activities**

- One object must perform two or more activities concurrently

- The activities must be completed before the object progress to next state

- Eg: Consider an ATM that dispenses cash and return the users card at the end of transaction. The machine must not reset itself until the user takes both cash and card

**Setting up**

**Emitting**

**Ready to reset**

do:dispense cash

do:eject card

ready

cash taken

card taken

**Fig 5.16 Synchronization of control**

**5.6 Sample dynamic model**

- Refer text pg no:119-124

**5.7 Relation of object and dynamic model**

- Dynamic model specifies allowable sequence of changes to objects from object model

- State diagram describes all or part of behavior of one object of a given class

- States are equivalence classes of attributes and link values for object

- Events can be represented as operations on object model

- Substate refines attributes and link values

- Each substate restricts the values that the object can have

- Single object can have different states over time

- Object preserves identity but it cannot have different classes

- Composite state is the aggregation of more than one concurrent substate

- There are three sources of concurrency within object model:

* Aggregation of objects: Each component of aggregation has its own independent state
* Aggregation within an object: Attributes and links of object are its part
* Concurrent behavior of object
* Dynamic model of a class is inherited by its subclass
* Subclass inherit states of ancestors and transitions
* Subclass can have its own state diagram
* Subclass has a composite state composed of concurrent state diagram

**Chapter- 6**

**Functional Modeling**

**6.1 Functional modeling**

- Functional model describes computation within a system

- Specifies what happens

- Consists of multiple data flow which show the flow of values from external inputs through

operations and internal data stores to external output

* Specifies the result of computation without specifying how or when they are computed
* Specifies the meaning of operation in object model and actions in dynamic model
* Spreadsheet is a kind of functional model

**6.2 Dataflow diagram**

- Data flow diagram shows functional relationship of values computed by a system including

input values, output values and internal data stores

- Data flow diagram is a graph whose node represent process and arc represents data flow

Data flow

Process

**Fig 6.1 Data Flow Diagram**

* Data flow diagram doesnot contain control information
* 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
* Fig 6.2 shows a data flow diagram for the display of an icon on a windowing system
* The icon name and location are inputs
* The icon is expanded to vectors using existing icon definition
* The vectors are clipped to the size of window then offset by location of window on screen to obtain vectors
* Finally the vectors are converted to pixel operations that are sent to screen buffer for display

location

size

application vector list

Icon definitions

**Window**

Icon name

location

screen vector list

pixel operations

Screen buffer

**Fig 6.2 Dataflow diagram for windowed graphics display**

**6.2.1 Processes**

- Process transforms data values

- Process is drawn as ellipse containing description of transformation usually its name

- Process has fixed number of inputs and output data arrows

- The inputs and outputs can be labeled

pixel operations

icon name

location

**Fig 6.3 Processes**

**6.2.2 Data Flows**

- Data flows connects the output of an object or process to input of another object

- Data flows is drawn as an arrow between producer and consumer of data value

- Arrow is labeled with a description of data

**Fig 6.4 Data flow**

**6.2.3 Actors**

- Actors are active object that drives the data flow graph by producing or consuming values

- Actors are attached to inputs and outputs of data flow graph

- Actors lie on boundary of data flow graph but terminate the flow of data and are called terminators

- Actors are drawn as rectangle

- Eg: user of program

**6.2.4 Data stores**

- Data stores are passive object within a data flow diagram that stores data for later access

- Data store doesnot generate any operations

- Data store is drawn as a pair of parallel lines containing the name of store

- Input arrows indicate information or operations that modify stored data; this includes adding elements, modifying value or deleting elements

- Data store might be implemented as a file

Max temp

Min temp

Temperature

Readings

**Fig 6.5 Data Store**

* Fig 6.5 shows data store for temperature reading. Every hour a new temperature reading enters the store. At the end of the day the maximum and minimum reading are retrieved from the store

**6.2.5 Nested data flow diagram**

- DFD is useful for showing high level functionality of a system

- A process can be expanded into another DFD

- Each input and output of process is an input or output of new diagram

- The new diagram may have data stores

- Nesting of data flow diagram permits each level to be coherent and understandable

- Nesting of diagrams is also called leveling since the diagrams are organized into different levels

**6.2.6 Control flows**

- Boolean value that affects whether a program is evaluated

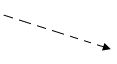
- Control flows is shown by a dotted line from a person producing a Boolean value to the process being controlled

password

balance

Account

Password ok



**Fig 6.6 DFD for withdrawal from bank account**

amount

cash

Customer

* Fig 6.6 shows DFD for a withdrawal from a bank account. The customer supplies a password and an amount. The withdrawal occurs only if the password is successfully verified. The update process could be expanded with a similar control flow to guard overdraft

**6.3 Specifying Operations**

- Process in data flow diagram must be implemented as operation on object

- Both high level and low level process are considered as operations

- Each operation may be specified in various ways:

* Mathematical functions such as trigonometric functions
* Pre and post conditions
* Decision tables
* Pseudocode
* Natural language
* Specification of an operation includes signature and transformation
* Signature defines the interface to the operation arguments required and value it returns
* Operation is listed in object model
* Transformation defines the effect of operation
* External specification of an operation describes changes visible outside operation
* During implementation of operation internal values may be created
* Internal details do not appear in external specification
* Access operation are operations that read/write attributes or links of an object
* Non trivial operations can be divided into three categories
* Queries
* Actions
* Activities
* Query is an operation that has no effect on externally visible state of an object
* Action is a transformation that has side effects on target object
* Action has no duration
* Activity is an operation which has duration

**6.4 Constraints**

- Constraints show relationship between two object at same time or between different values of

same object at different time

* Constraints may be expressed as total function or partial function
* Constraint can appear in each model
* Object constraints specify that same object depend entirely or partially on another
* Dynamic constraint specifies relation among states or events of different object
* Functional constraint specify restrictions on operation
* Constraint between values of an object over time is called invariant

**6.5 Relation of Functional to Object and Dynamic Model**

- The functional model shows what has to be done by a system

- Object model shows doers-the object

- Dynamic model shows the sequence in which operations are performed

- The processes in functional model correspond to operations in object model

- Top level process corresponds to an operation on complex object and lower level process

correspond to operations on more basic objects

- Process in functional model show objects that are related by function

- One of the input to a process can be identified as a target object

- Target object is a client of other objects(called suppliers)

- Process is implemented as method

- If the same class of object is an input and output, then object is the target and other inputs are

arguments

- If the output of process is a data store, data store is the target

- Actors are explicit objects in object model

- Data flows to or from actors represent operations on or by the objects

- Data stores are passive objects that respond to queries and updates

- Data flows are values in object model such as numbers, strings or lists of pure values

- Pure values do not have identity

- Pure value is not a container whose value can change

- Pure value has no state and no dynamic model

- Relative to functional model: The object model shows the structure of actors, data stores and

flows in functional model. Dynamic model shows the sequence in which processes are

performed

- Relative to object model: The functional model shows the operation on classes and arguments

of each operation. It shows supplier-client relationship among classes. Dynamic model shows

state of each object and operations that are performed as it receives events and change state

- Relative to dynamic model: The functional model shows definition of actions and activities that

are undefined with dynamic model. Object model shows what changes state and undergoes

operation

**6.6 Sample functional model**

- Refer text pg no: 147-151