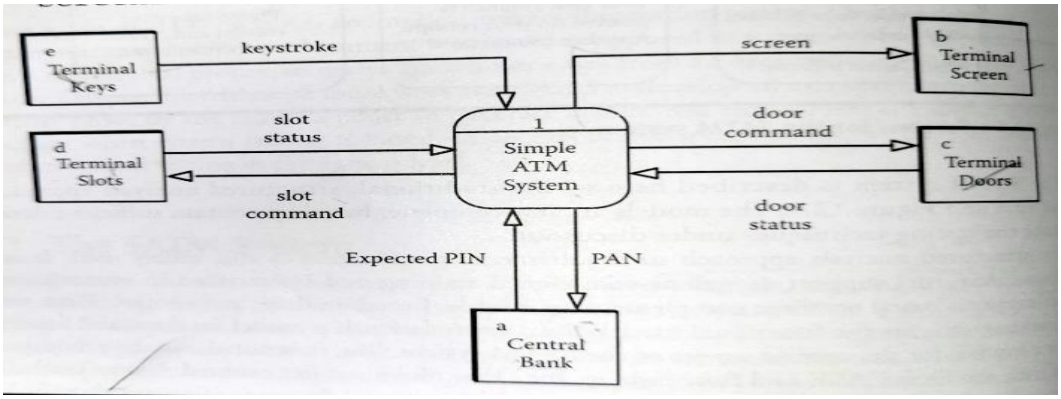
USN										

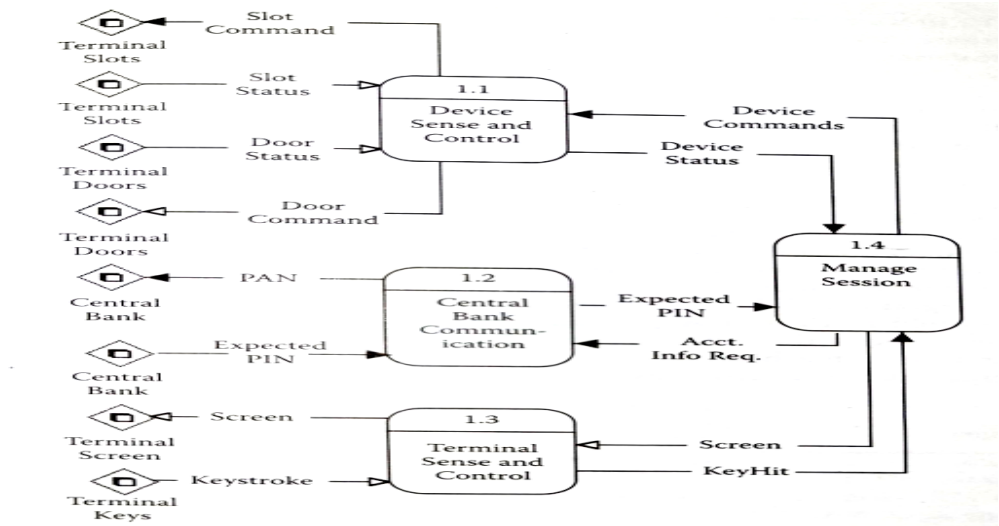
Internal Assessment Test 3 – JUL 2024

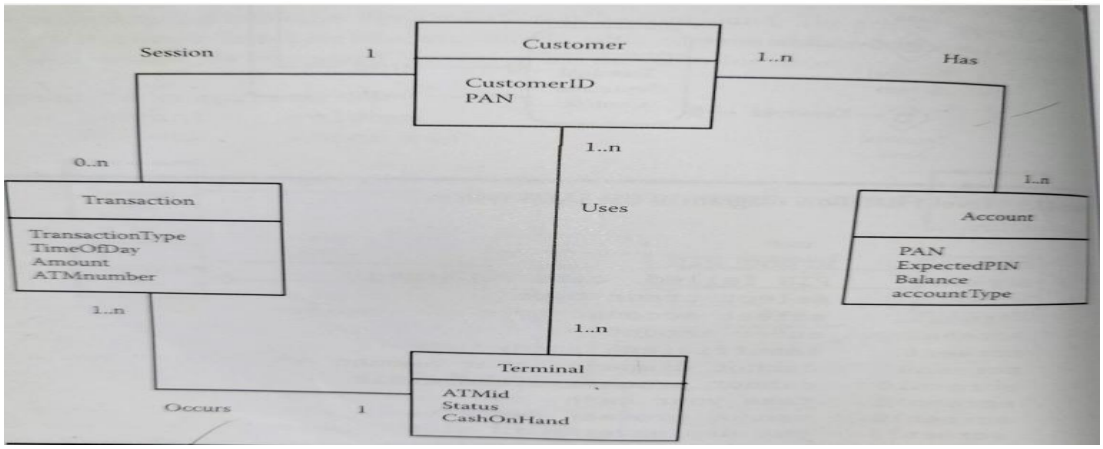
Sub:	**SOFTWARE TESTING**					Sub Code:	**21IS63**	Branch:	**ISE**
Date:	**29/07/2024**	Duration:	**90 min**	Max Marks:	**50**	Sem/Sec:	**VI/ A, B & C**		**OBE**

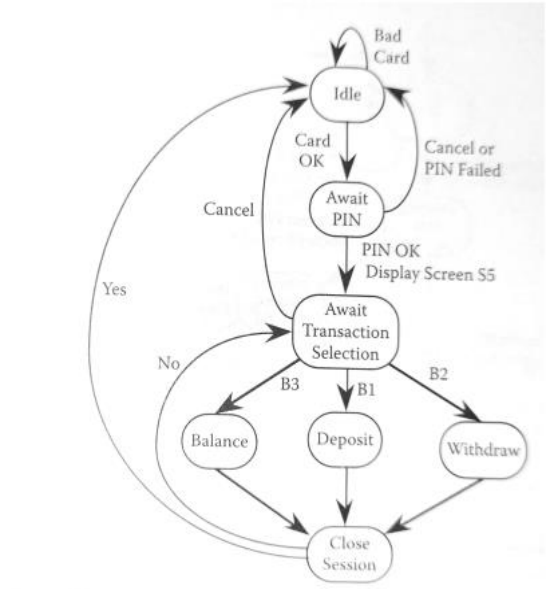
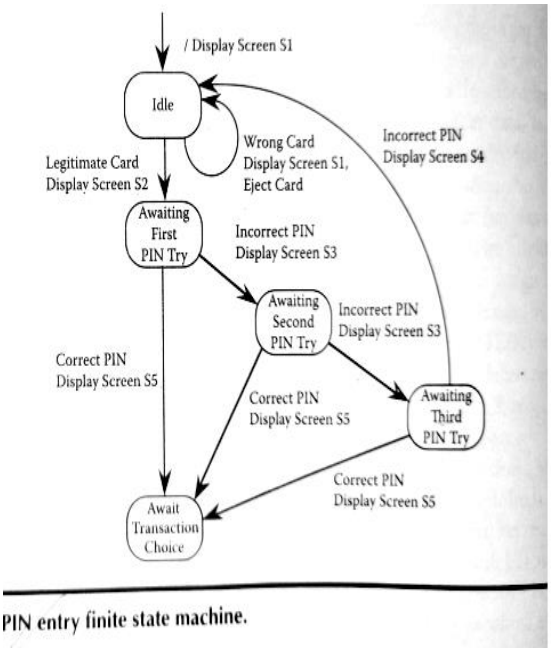
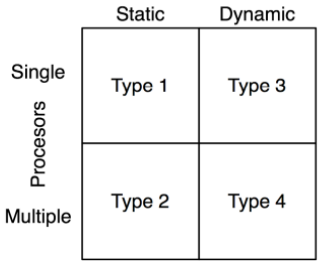
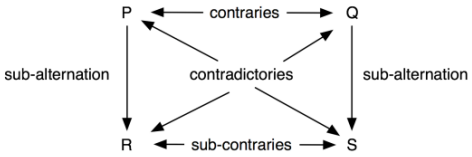
Answer any FIVE FULL Questions

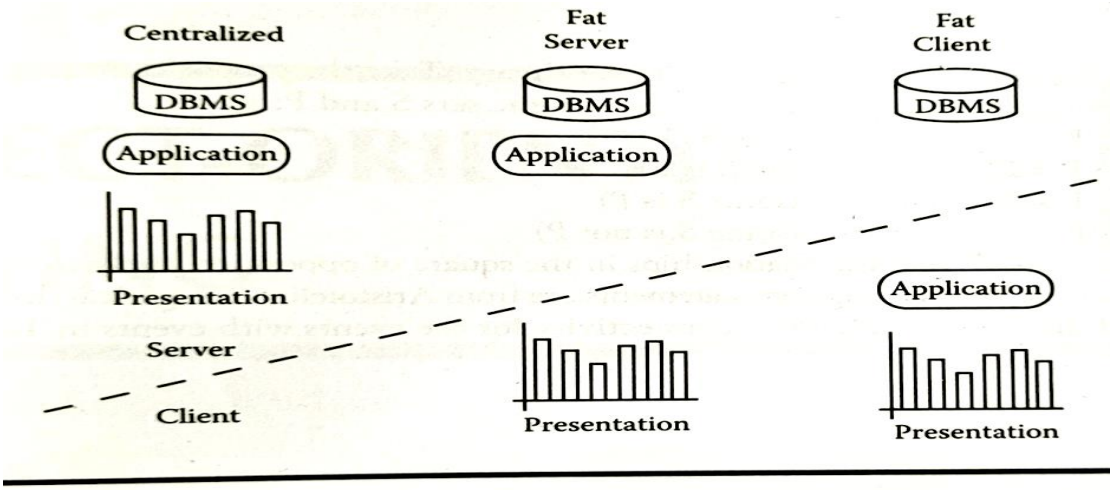
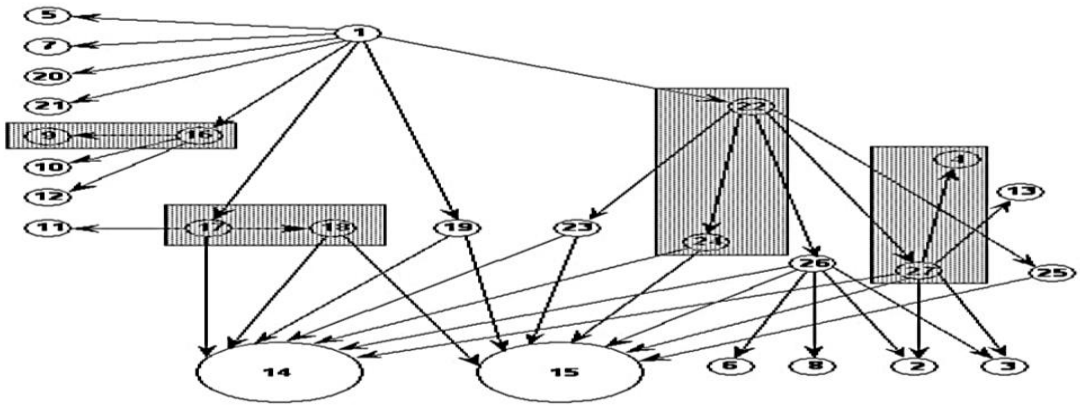
		MARKS	CO	RBT
1.	a. Explain context diagram, Level-1 dataflow diagram, Entity/relationship model of the SATM System in Testing. **Scheme: Explanation + Diagram – 2+2+2 marks** **Solution:**	[6]	4	L2

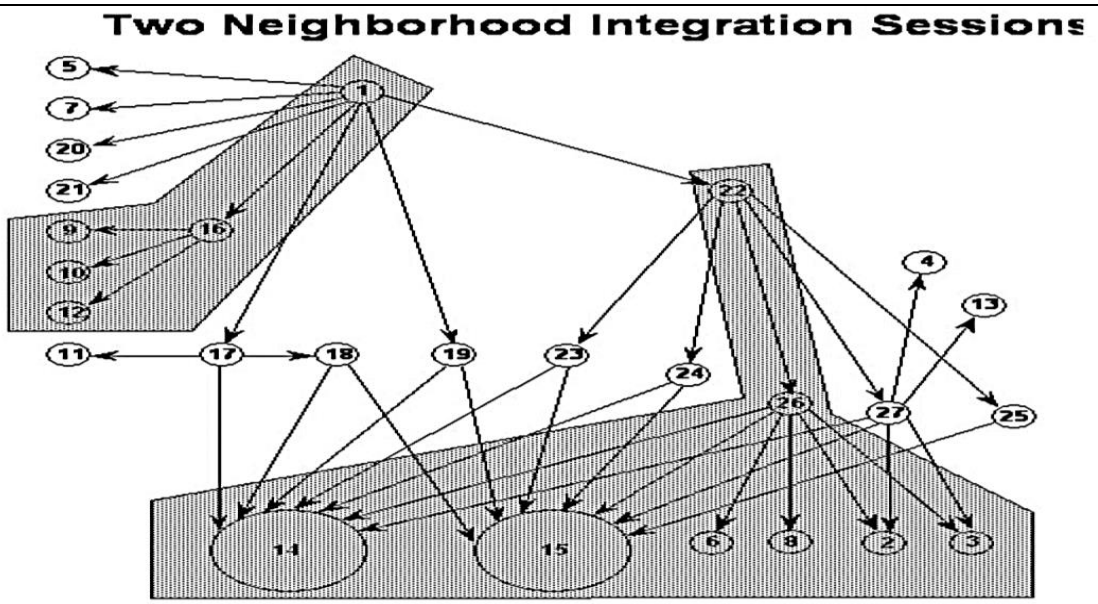






	b. Explain the following for SATM System: i. Upper-level FSM ii. Pin Entry FSM **Scheme: Explanation + Diagram – 2+2 marks** **Solution:**  Upper-level SATM finite state machine.  PIN entry finite state machine.	[4]		
2.	a. Explain any two types of interaction taxonomies in testing. **Scheme: Explanation + Diagram – 3+3 marks** **Solution:** ▪ Static interactions in a single processor system ▪ Static interactions in multiprocessor system ▪ Dynamic interactions in a single processor system ▪ Dynamic interactions in multiprocessor system ▪ Given two propositions P and Q ▪ **They are contraries if both cannot be true** ▪ **Sub-contraries if both cannot be false** ▪ **Contradictories if exactly one is true** ▪ **R is a subaltern of P if the truth of P guarantees the truth of R – i.e. P → R**   **Static interactions in a single processor** ▪ Analogous to combinatorial circuits ▪ **Model with decision tables and unmarked event-driven Petri nets** ▪ **Telephone system example** ▪ **Call display and unlisted numbers are contraries** ▪ Both cannot be satisfied ▪ Both could be waived	[6]	5	L2

	b. Explain Fat Clients and Servers in interaction testing. **Scheme: Explanation + Diagram – 2+2 marks** **Solution:**  **Fat clients and servers.**	[4]		
3.	Explain the call graph-based integration testing with the help of: a. Pair-wise Integration b. Neighborhood Integration **Scheme: Explanation + Diagram – 5+5 marks** **Solution:** **Pair-Wise Integration** - The idea behind Pair-Wise integration testing is to eliminate the need for developing stubs/drivers - The objective is to use actual code instead of stubs/drivers - In order not to deteriorate the process to a big-bang strategy, we restrict a testing session to just a pair of units in the call graph - The result is that we have one integration test session for each edge in the call graph **Some Pair-wise Integration Sessions**  - We define the neighbourhood of a node in a graph to be the set of nodes that are one edge away from the given node - In a directed graph means all the immediate predecessor nodes and all the immediate successor nodes of a given node - Neighborhood Integration Testing reduces the number of test sessions	[10]	4	L2



4. Apply ASF Sequences for NextDate Function in system testing. [10] 5 L3

Scheme: Diagram + 4 different sequences – 2+2+2+2+2 marks

Solution:

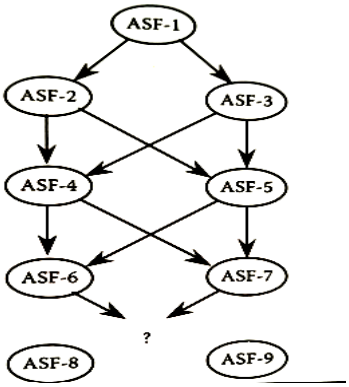


Table 14.13 NextDate Input Events

Event	Input Event Description	Statement Numbers
e0	Start program event	1
e1	Enter a valid month	67
e2	Enter an invalid month	67
e3	Enter a valid day	69
e4	Enter an invalid day	69
e5	Enter a valid year	71
e6	Enter an invalid year	71

Table 14.14 NextDate Output Events

Event	Output Event Description	Statement Numbers
e7	Welcome message	2
e8	Print today's date	4
e9	Print tomorrow's date	6
e10	"month OK"	39
e11	"month out of range"	41
e12	"day OK"	47
e13	"day out of range"	49
e14	"year OK"	54
e15	"year out of range"	56
e16	"date OK"	60
e17	"please enter a valid date"	62
e18	"enter a month"	66
e19	"enter a day"	68
e20	"enter a year"	70
e21	"Day is month, day, year"	89

Table 14.15 First Attempt at ASFs for NextDate

Atomic System Function	Inputs	Outputs
ASF-1 start program	e0	e7
ASF-2 enter a valid month	e1	e10
ASF-3 enter an invalid month	e2	e11
ASF-4 enter a valid day	e3	e12
ASF-5 enter an invalid day	e4	e13
ASF-6 enter a valid year	e5	e14
ASF-7 enter an invalid year	e6	e15
ASF-8 print for valid input		
ASF-9 print for invalid input		

Table 14.16 Second Attempt at ASFs for NextDate

Atomic System Function	Inputs	Outputs
ASF-1 start program	e0	e7
ASF-2 enter a date with an invalid month, rest OK	e2, e3, e5	e11, e12, e14, e17
ASF-3 enter a date with an invalid day, rest OK	e1, e4, e5	e10, e13, e14, e17
ASF-4 enter a date with an invalid year, rest OK	e1, e3, e6	e10, e12, e15, e17
ASF-5 enter a date with valid month, day, and year	e1, e3, e5	e10, e12, e14, e16, e21
ASF-6 enter a date with valid month, rest invalid		
ASF-7 enter a date with valid day, rest invalid		
ASF-8 enter a date with valid year, rest invalid		
ASF-9 enter a date with invalid month, day, year		

5.	Explain Functional Strategies for thread testing. **Scheme: Explanation + Diagram – 5+5 marks** **Solution:**	[10]	5	L2

Coverage metrics for functional models can be based on

- Events
- Ports
- Data

There are five port input thread coverage metrics

- PI1: each port input event occurs
  - Minimum requirement
- PI2: each common sequences of port input events occurs
  - Normal usage; difficult to quantify
- PI3: each port input event occurs in every "relevant data context'
  - Physical events with different logical meaning (e.g. B1 with different meaning because of the context"
- PI4: for a given context, all "inappropriate" input events occurs
  - Negative scenarios/unexpected input events (e.g., B1 for digit) /hardware failures
- PI5: for a given context, all possible input events occurs
  - Testers have difficulty to anticipate things that should not happened)

There are two port output thread coverage metrics

- PO1: each port output event occurs
  - Set of output msgs for each error conditions
- PO2: each port output event occurs for each cause

Port-based Thread Testing

- works for systems having port devices coming from external supplies
- Identify for each port the corresponding event (s)
- Find threads that exercise input/output ports w.r.t. to the event lists

Data-centric thread based testing

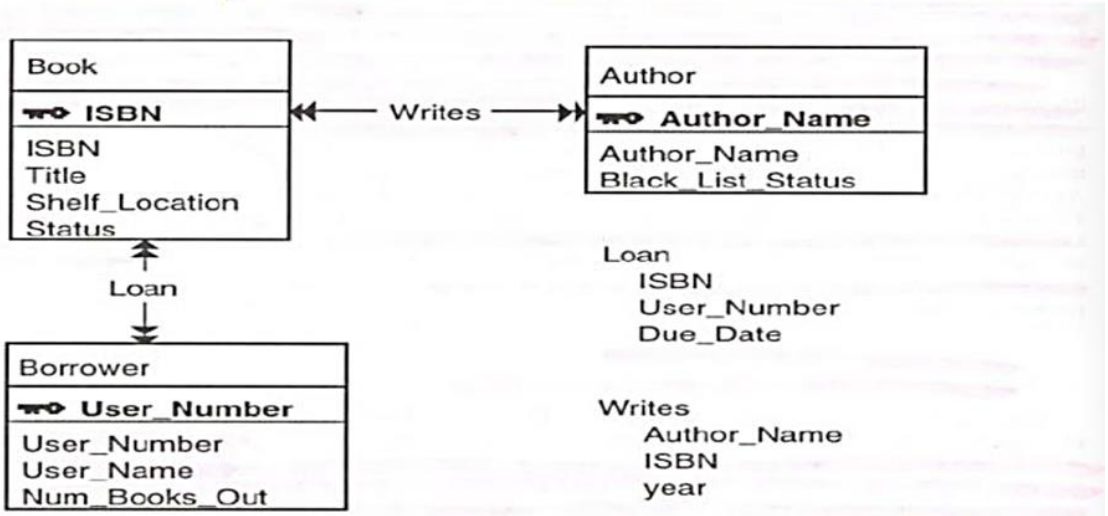
- works for data driven system
- Not appropriate for event-driven or reactive systems

- Uses ERD (or similar models as starting points)
- The coverage includes
  - DM1: exercise the cardinality of every relationship in ERD
  - DM2: exercise the participation of every relationship in ERD
  - DM3:Exercise the functional dependencies among relationships in ERD

Example: Transactions for Library system

- Some typical transactions in the library system
  - Add a book to the library
  - Delete a book from the library
  - Add a borrower to the library
  - Delete a borrower from the library
  - Loan a book to a borrower
  - Process the return of a book from a borrower

Example of Data-based thread

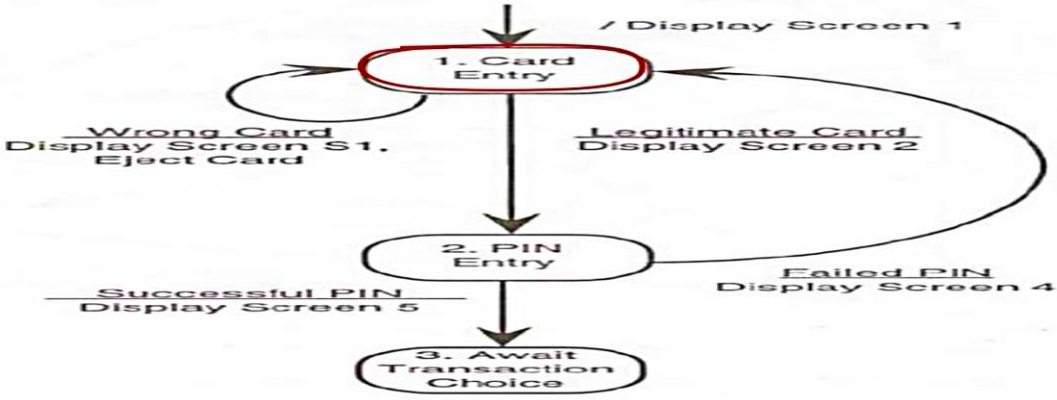


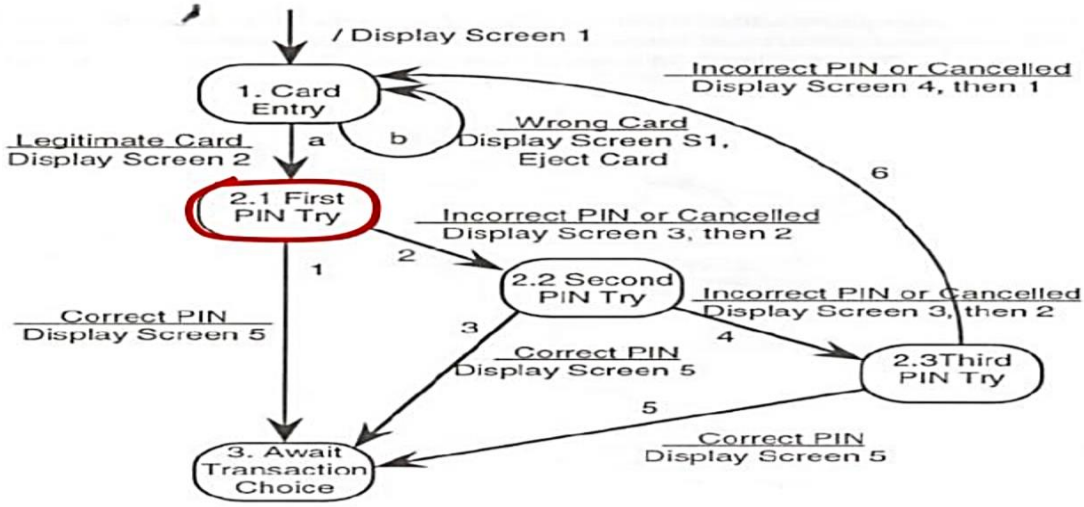
6.	Describe the concept of finding threads in system testing. **Scheme: Explanation + Diagram – 5+5 marks** **Solution:**	[10]	5	L2

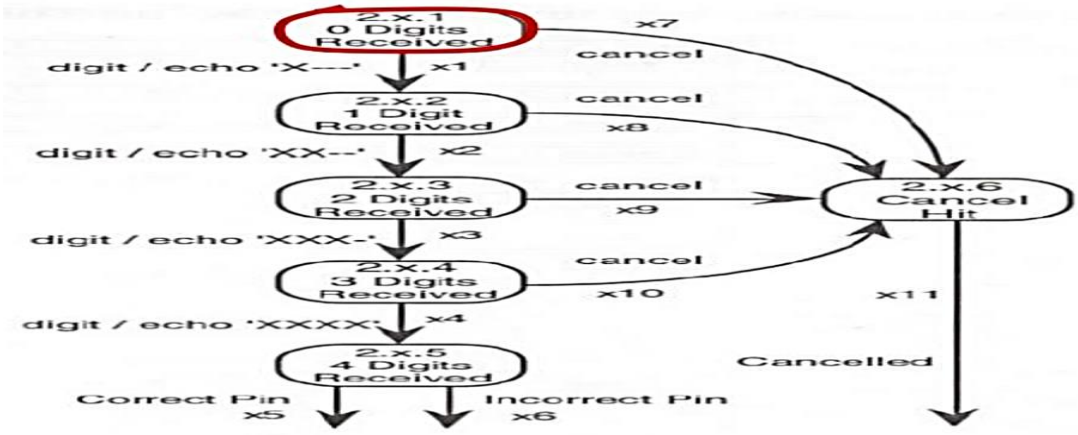
A hierarchy of state machines;

- the **upper level** - states correspond to stages of processing, and transitions are caused by logical (rather than port) events.

- The **Card Entry "state"** for example, would be decomposed into **lower levels** that deal with details like **jammed cards, cards that are upside-down, stuck card rollers, and checking the card** against the list of cards for which service is offered.







Events in the PIN Entry Finite State Machine

Port Input Events	Port Output Events
Legitimate Card	Display screen 1
Wrong Card	Display screen 2
Correct PIN	Display screen 3
Incorrect PIN	Display screen 4
Canceled	Display screen 5

Faculty Signature CCI Signature HOD Signature