→ UNIT-II Transaction flow Testing

Topics:
Transaction flows, transaction flow testing techniques Dataflow testing, basics of dataflow testing, strategies in dataflow testing, application of dataflow testing.

1) Introduction:-
A transaction is a unit of work seen from a system user point of view.
It consists of sequence of operations, some of which are performed by a system, person or devices, that are outside of the system.
Transactions are created as a result of some external act.
- At the end of transactions processing, the transactions is no longer existing in the system.

example: A transaction for an online information retrieval system might consist of following steps
- Accepting input
- validating input
- Transmit Acknowledgement to the requester.
- Do input processing
- search file.
- Request direction from user
- Again Accepting input
- validating input
- Processing request
- update file
- Transmit output
- Record transaction in a log and clean up

→ Transaction flow Graph
Transaction flows are introduced as representation of systems processing.
Transaction flow Graph is to create behavioural model of the program that leads to functional testing.

26/06/25 :

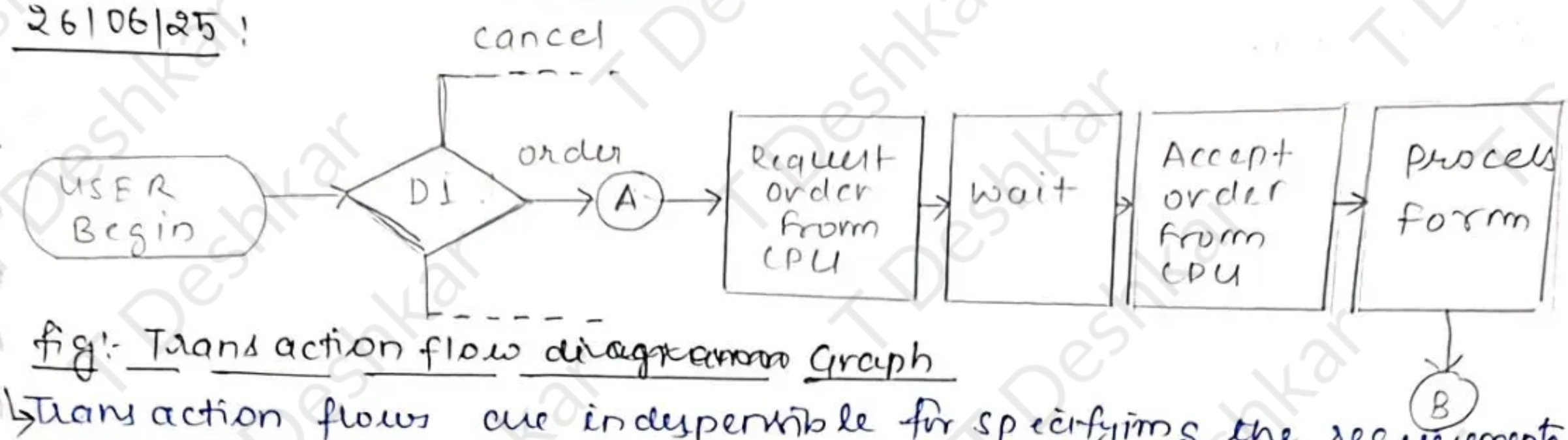


fig:- Transaction flow diagram Graph

↳ Transaction flows are indespensible for specifying the requirements of complicated systems.

example: A big system such as airtraffic control or airline reservation system has thousands of different transaction flows.

Another example is for ATM system. An ATM system allows the user to try (atleast three times) and it'll take the card away for the 4th time. for any transaction.

~~To try for any particular tra~~

→ Complications :

→ In simple cases the transaction have a unique identity from the time they are created to the time they are completed.

(2) In many system transaction can give birth to another transaction and can also merge two different transactions.

→ Birth

- there are three possible interpretations of the decision symbols. It can be

1) Decision , 2) Biosis , 3) Mitosis

→ Decision :

Here the transaction will take one alternative or the other alternative.

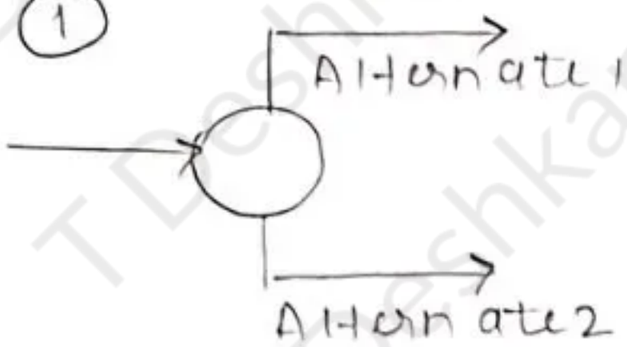
→ Biosis

Here the incoming transaction gives birth to a new transaction and both the transaction continue on their seperate paths.

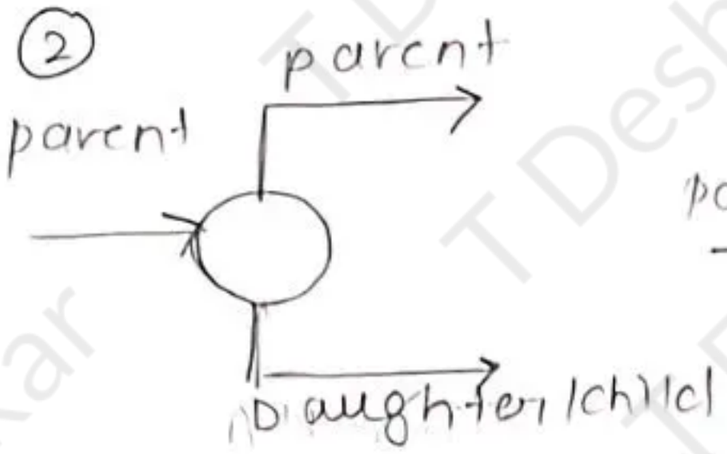
→ Mitosis

Here the parent transaction is destroyed and two new transaction

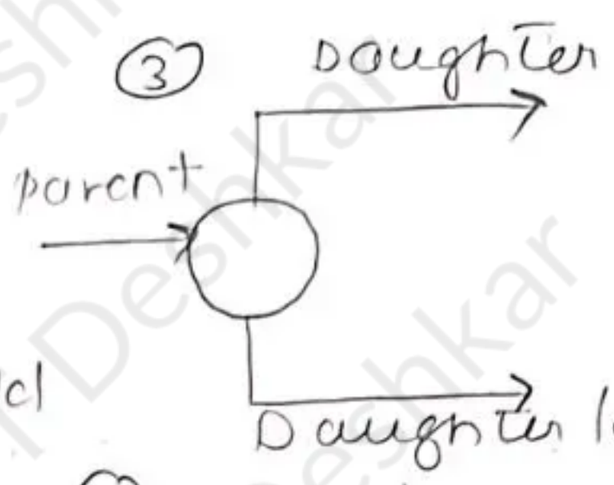
are created.



a) Decision



(b) Biasis



(c) Mitosis

27/06/25 **Transaction flow Testing strategies**

1) Inspection reviews and walkthrough

→ Inspections :- Any formal review is conducted while it is executing a task then it is called an inspection. or operational

- Inspections are done to identify errors in the program

1) checking the syntax errors
2) Refering the program
3) ~~Validating~~ Voleting ~~Rules~~ validating Rules
4) comparison of code
5) comparing the flowchart

→ Review: Examining on cross checking a project related work log and a process related work is called a review.

→ Reviews are done to identify errors in the semantic document.

→ It includes (1) Review planning
(2) Review document information
(3) Preparing Review meeting
(4) objective of Review meeting
(5) Review scheduling
(6)

→ walkthroughs: Training sessions or knowledge transfers session (KT) about the process / technologies are walkthroughs

2) In conducting the walk through, you should

1) Discuss enough transaction types to account for 98-99% of the transactions of the system is expected to process

2) Discuss path through flows in functionality rather than technical terms.

(3) Ask the Designers to relate every-flow to the specifications and to show how their transactions directly or indirectly follow the requirements.

↳ Advantages and Disadvantages ↓

1) ↳ Transaction-flow uses structural path testing for selecting a path	1) path selected is the longest path for transaction testing
2) Bugs don't occur in the path selection	2) The code correction is expensive
(3) path selection is effective	3) The bugs are cost due to implementation errors.

→ 30 June 26 Data flow testing

1) Basics of Data flow testing:

Data flow testing is the name given to a family of test strategies based on selecting the path through program control in order to explore the sequence of events related to the states of data objects

2) for example- pick enough paths to ensure that every data object has been initialized prior to use. or that all the defined data objects have already been used.

→ Data flow machines

There are two types of data flow machines with different architecture.

1) Von Neumann Machine architecture
2) Multi instruction, multi data machine (MIMD)

1) Most computers today are based on Von Neumann machine this architecture features are inter changable storage of instructions and data in the same memory unit

2) Von Neumann machine architecture executes one instructions at a time in the following micro instruction sequence.
   1) fetch Instruction from memory
   2) Interprete instruction
   3) fetch operands
   4) process or execute

5) Store result
6) Increment program counter
7) Go to one

→ Multi data instruction, Multi data machines (MIMD)

1) These machines can fetch several instructions & data objects in parallel
2) they can also do arithmetic and logical operations simultaneously on different data objects
3) The decisions of sequence of events depends on compiler

→ Data flow Graph

1) The Data flow graph is a graph consisting of nodes and directed links.
2) we will use a control flow graph to show what happens to a data object at particular movement.

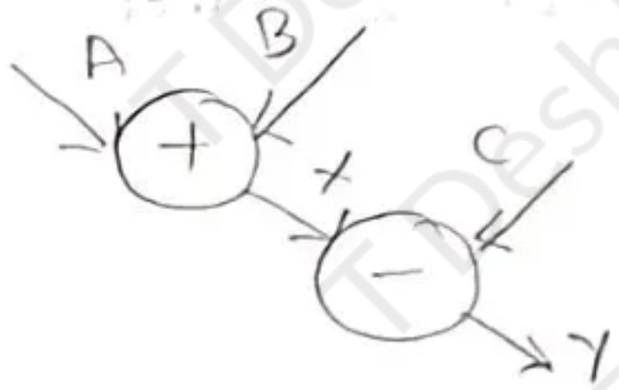
02/07/25 Data flow Graph Model

- Data driven model where the program executions is driven by the data
- It Translates the requirements in the data-flow.
- There are two types of DFG 1) Non cyclic 2) cyclic
- Cyclic: No multiple values, No feedback, No events
- Non-cyclic :- The system is with feedback and events

→ DFG (Data flow Graphs)

- Visual Model where operations on the data is represented by using circle.
- Data flow is represented by arrow
→ Inward arrows to the ~~circle~~ circle: represents input data.
→ Outward arrows to the circle: represents o/p data.

(Ex) $X = A + B$ & $Y = X - C$



→ components of the model

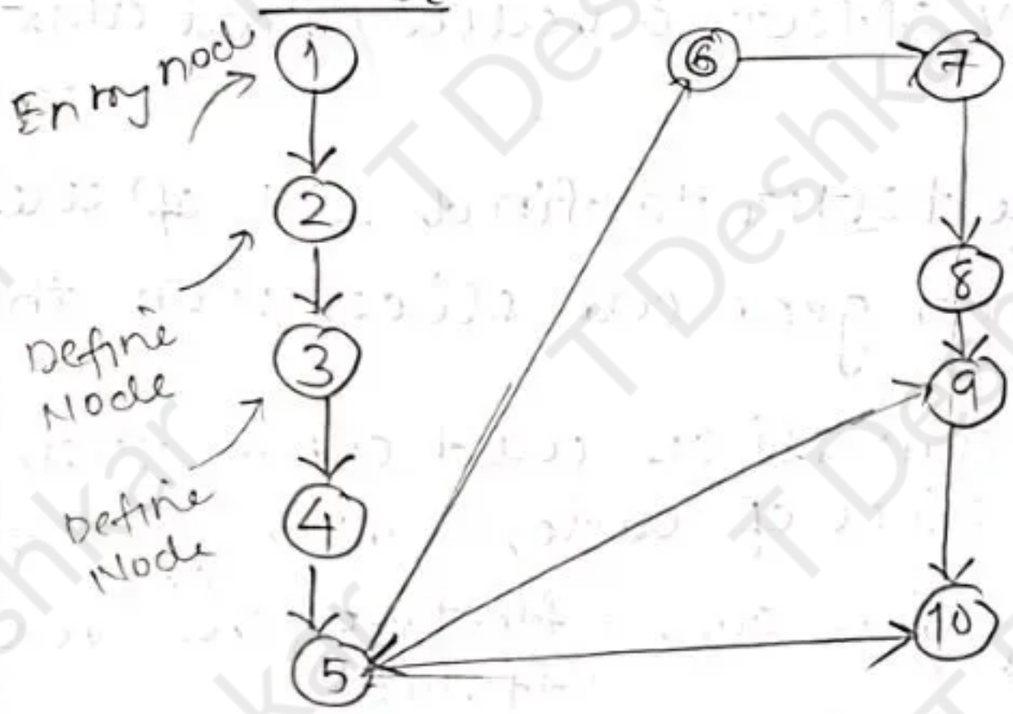
- To every statement there is are node with unique name every predicate node has atleast one outlink and atleast one out inlink. except for start exit and entry node.
- Exit nodes are the dummy nodes place at outgoing arrow of exit statements. similarly the entry nodes are the dummy nodes placed at entry statement.
- out link of simple statement are weighted by proper sequence. of data flow action

Predicates Nodes (if, then, else, do, while, case are weighted with the path

03/07/25 Data flow testing Example (strategy & Techniques).

1)	int i, n, sum
2)	i = 0
3)	sum = 0
4)	input(n)
5)	while(n!=0)
6)	sum = sum+n
7)	i = i+n
8)	O/p(sum) I/p(n)
9)	End while
10)	print("sum")

→ DFG



→ In data flow testing they are two testing techniques & strategy.

1) DU testing technique - Define Use Testing Technique
2) program slicing Technique

- Data flow testing strategies are structural strategies (DFT)
- DFT strategies are based on selecting test path segment (subpath)

04/07/25

Use Node: where we will be using variables to perform opr.

The other examples of Nodes used in DU testing are.

i Node: Here i is for performing the iterations ex i = i+1

L Node: Here L Node is for tracing location. ex accessing address of array

or pointer.

③ O Node : It is for output use.

Process : process for DU testing includes selecting a set of paths and then testing for any bugs or anamolies (confirming outputs)
- Here a matrix is created that represents all the path.
② all the edges ③ all the nodes.

→ program slicing : Dividing a large program
A program slice wrt to a variable at a certain point in the program, is the set of program statements from which the value of the variable at that point of the program is calculated.

- slicing takes a slice or a group of program statements in their program for testing particular test connection condition or cases that may effect a value/variable. at a particular point of Interest.
- slicing allows the tester to find list of ures nodes from the graph and then generate slices with them.
- Notation: program slices used a notation S(V,N)
S : particular slice of code, V: variable, N: no. of lines.
- How many no. of lines are ~~affecting~~ affecting the set value of variable NL statement No. (i.e. with the node number)ka wrt to program graph)

→ slicing Guidelines / Rules

① All the statements where variables are defined and redefined should be considered

(2) All the statements where variables are receiving values externally should be considered.

(3) All statements where output of a variable is printed should be considered

(4) All the statements where relevant output is printed should be considered

(5) The status of all the variables may be considered at last statement of the program.

05/07/25

(ex)
1) a=3
2) b=6
3) c=b²
4) d=a²+b²
(5) c=a+b

Program slice
wrt c variable ~~SC(1,7)~~ = {
SC(1,3) = {~~2,5~~} {2,3}
SC(1,5) = {~~1,2,3,5~~} {1,2,3,5}

ex-(1) 1) a=3
2) b=6
3) c=b²-14 · d=10
~~4) d=a+c~~
5) c=a+b
6) d=c+b

program slice
wrt d variable
S(d,4) = {3,4}
S(d,5) = {1,4,5}
S(d,7) = {2,3,4,5,7,6}

ex-(11)

9/7/25 Good

10-July-25

⇒ Applications of Dataflow Testing

1) Identifying the uninitialized variable
ex. int a; if(a>b) # error.
int b=5;

2) Detecting unused variable. → int a,b=5, c=20;
if(a>b)
#statements

3) Tracking Data Dependency → int a,b=5, c=20
c = a*b; if(c>b)

4) Testing data updating & manipulating.

→ Identifying the uninitialized variable: DFT helps to find instances where variables are used before they are assigned a value, which can lead to unexpected program behaviour / errors

→ Detecting unused variable
By identifying variables that are declared but never actually used, DFT helps to streamline to code and reduced unnecessary memory allocation.

(ex) int a, b, sum=0;
print(a+b);

→ Tracking data dependency:
DFT allows for tracking data flow from its defination to all its use, ensuring that data is being manipulated and passed correctly between different parts of the program.

→ Testing data updating and manipulations
DFT verifies that data is being updated and modified as intended.

throughout the program execution, preventing the logical errors and inconsistencies.

(4) Uncovering dataflow between functions :- DFT is particularly useful for understanding how data moves between different functions or modules, ensuring that data is being shared and used as expected.

(5) Identifying data inconsistencies :- DFT can detect situations where a variables value is unexpectedly overwritten/modified which could lead to bugs or incorrect result

(7) Checking for Data leaks :-
DFT can help identifying the situation where sensitive data is not properly handled or cleared from the memory after use. potentially leading to security vulnerabilities.