Integration Testing cont …

Prof. Durga Prasad Mohapatra
Professor
Dept.of CSE, NIT Rourkela

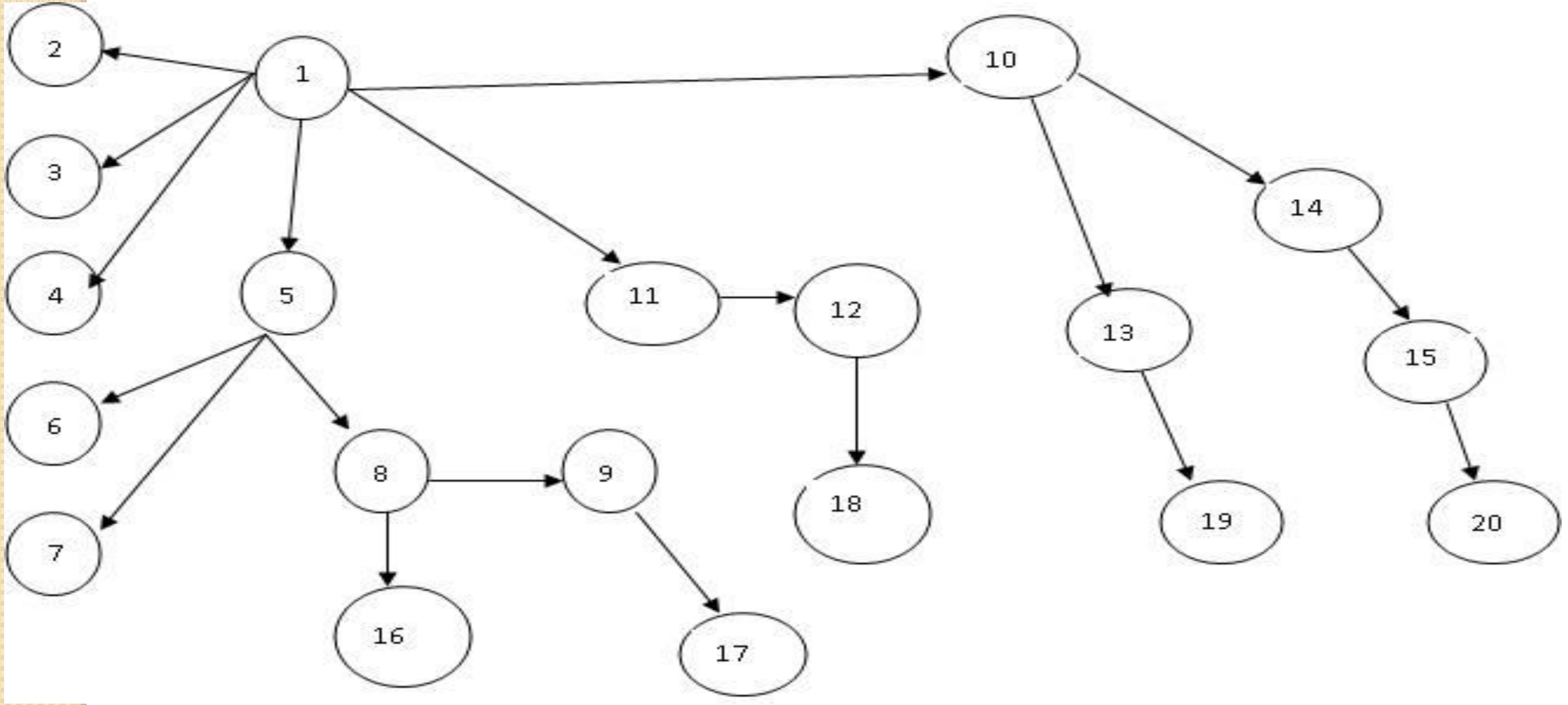
Call Graph-Based Integration

- It is assumed that integration testing detects bugs which are structural.
- However, it is also important to detect some behavioral bugs.
- If we can refine the functional decomposition tree into a form of module calling graph, then we are moving towards behavioral testing at the integration level.
- This can be done with the help of a *call graph* as given by Jorgensen.

Call Graph-Based Integration

- A call graph is a directed graph, wherein the nodes are either modules or units, and a directed edge from one node to another means one module has called another module.
- The call graph can be captured in a matrix form which is known as the adjacency matrix.

Example Call Graph



Adjacency Matrix

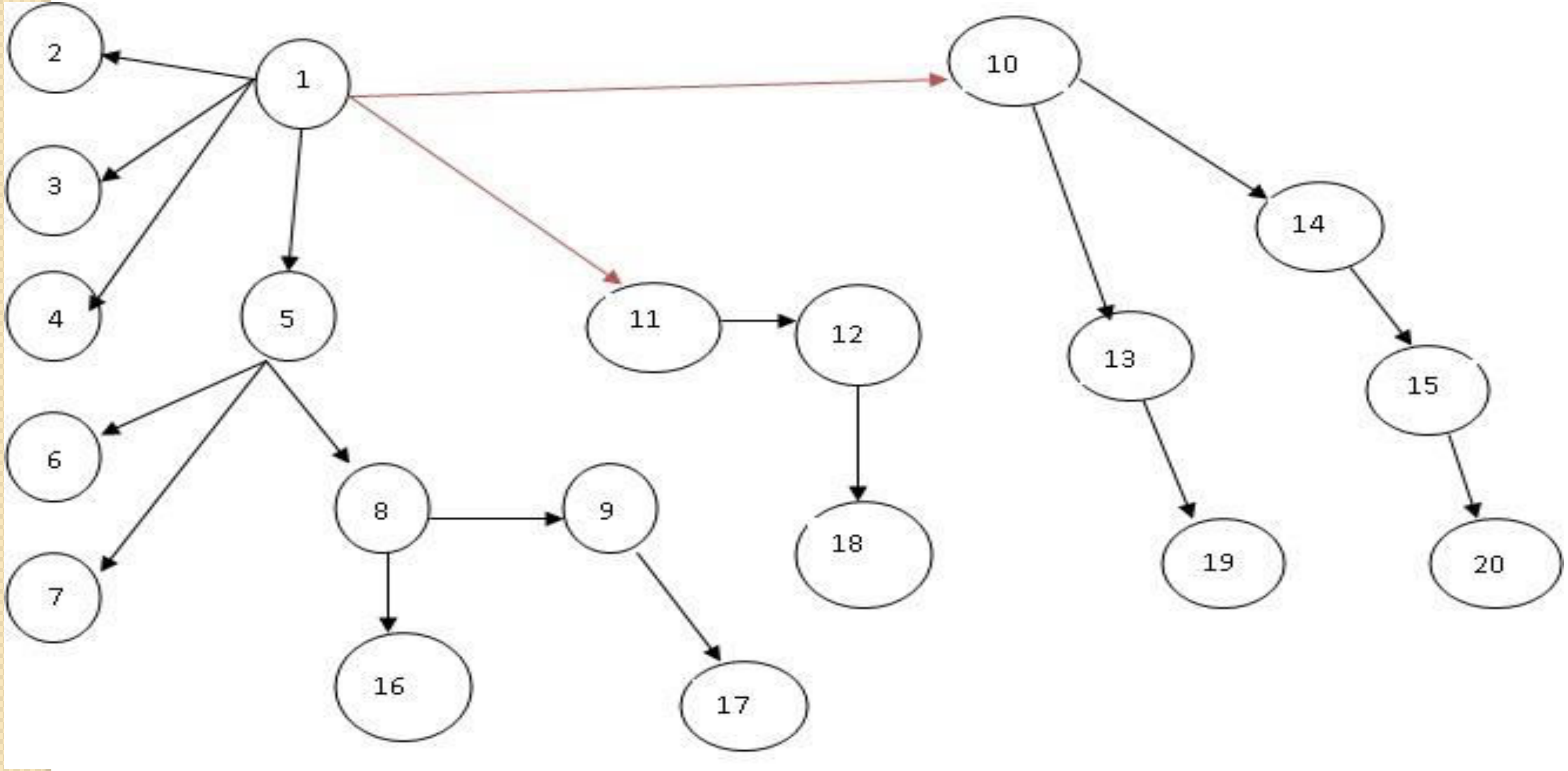
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	x	x	x	x					x	x									
2																			
3																			
4																			
5					x	x	x												
6																			
7																			
8								x							x				
9																x			
10												x	x						
11											x								
12																	x		
13																		x	
14														x					
15																			x
16																			
17																			
18																			
19																			
20																			

•The idea behind using a call graph for integration testing is to avoid the efforts made in developing the stubs and drivers.

•If we know the calling sequence, and if we wait for the called or calling function, if not ready, then call graph based integration can be used.

- There are two types of integration testing based on call graph:
  - Pair-wise Integration
    - If we consider only one pair of calling and called modules, then we can make a set of pairs for all such modules.
    - The resulting set will be the total test sessions which will be equal to the sum of all edges in the call graph.
  - Neighborhood Integration

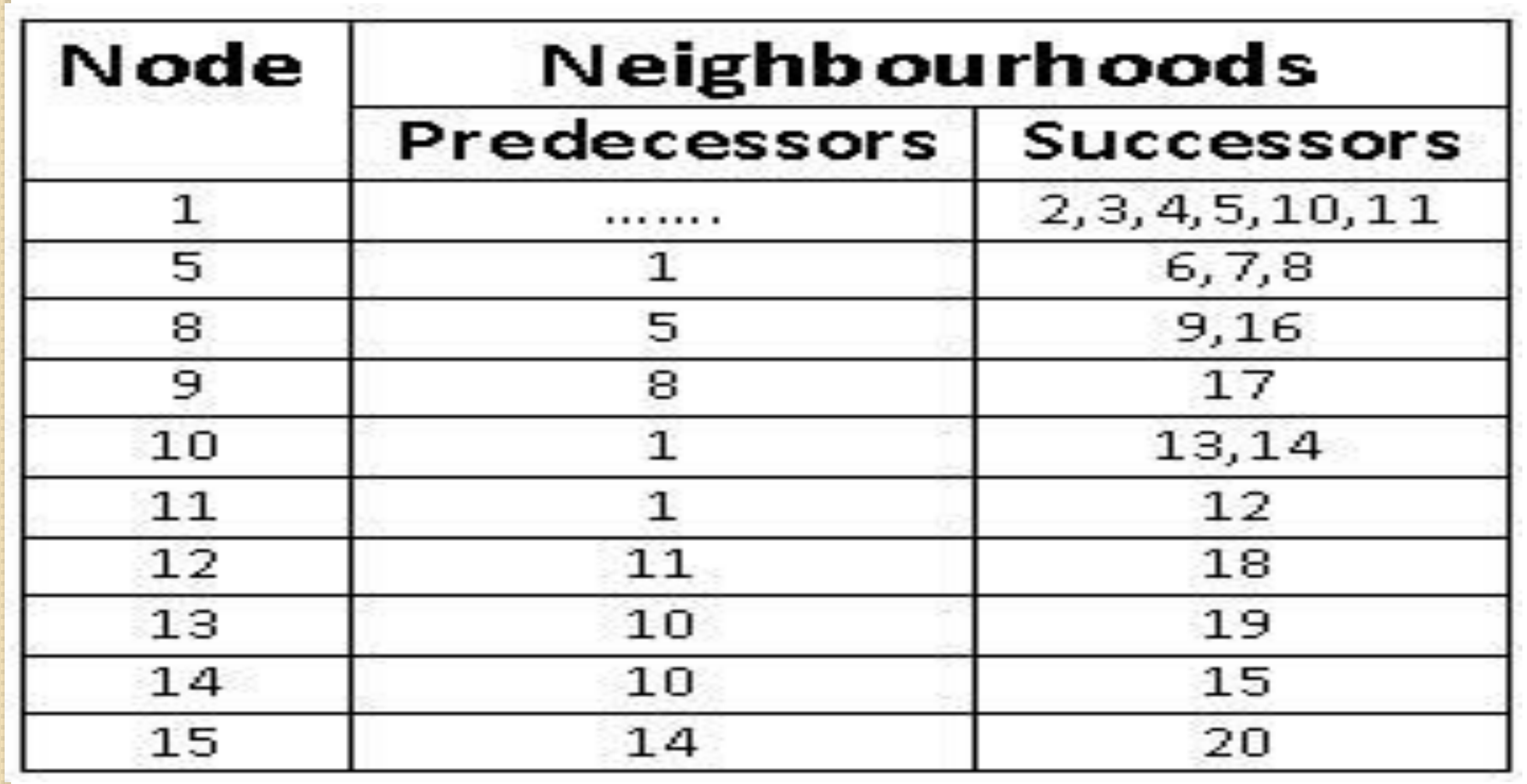
Pair wise Integration



Test sessions = No. of edges = 19

Neighborhood Integration

- There is not much reduction in the number of test sessions in pair-wise integration as compared to the decomposition-based integration.
- If we consider neighborhoods of a node in the call graph, then number of test sessions may reduce.
- Neighborhood of a node is the immediate predecessor as well as the immediate successor of the node.
- So, neighborhood of a node can be defined as the set of nodes that are one edge away from the given node



Node	Neighbourhoods	
	Predecessors	Successors
1	…….	2,3,4,5,10,11
5	1	6,7,8
8	5	9,16
9	8	17
10	1	13,14
11	1	12
12	11	18
13	10	19
14	10	15
15	14	20

The total test sessions = nodes – sink nodes= **20 – 10 = 10**

sink node is an instruction in a module at which execution terminates.

Path Based Integration

- In call graph, when a module or unit executes, some path of the source instructions is executed.

- It is possible that in the path execution, there may be a call to another unit.

- At that point, the control is passed from the calling unit to the called unit.

Path Based Integration

- This passing of control from one unit to another unit is important for integration testing.

- It can be done with path based integration.

- We need to understand the following definitions for path-based integration:

- –Source node
  - It is an instruction in the module at which the execution starts or resumes.
  - The nodes where the control is being transferred after calling the module are also source nodes.
- –Sink node
  - It is an instruction in the module at which the execution terminates.
  - The nodes from which the control is transferred are also sink nodes.
- –Module execution path (MEP)
  - It is a path consisting of set of executable statements within a module like in a flow graph.

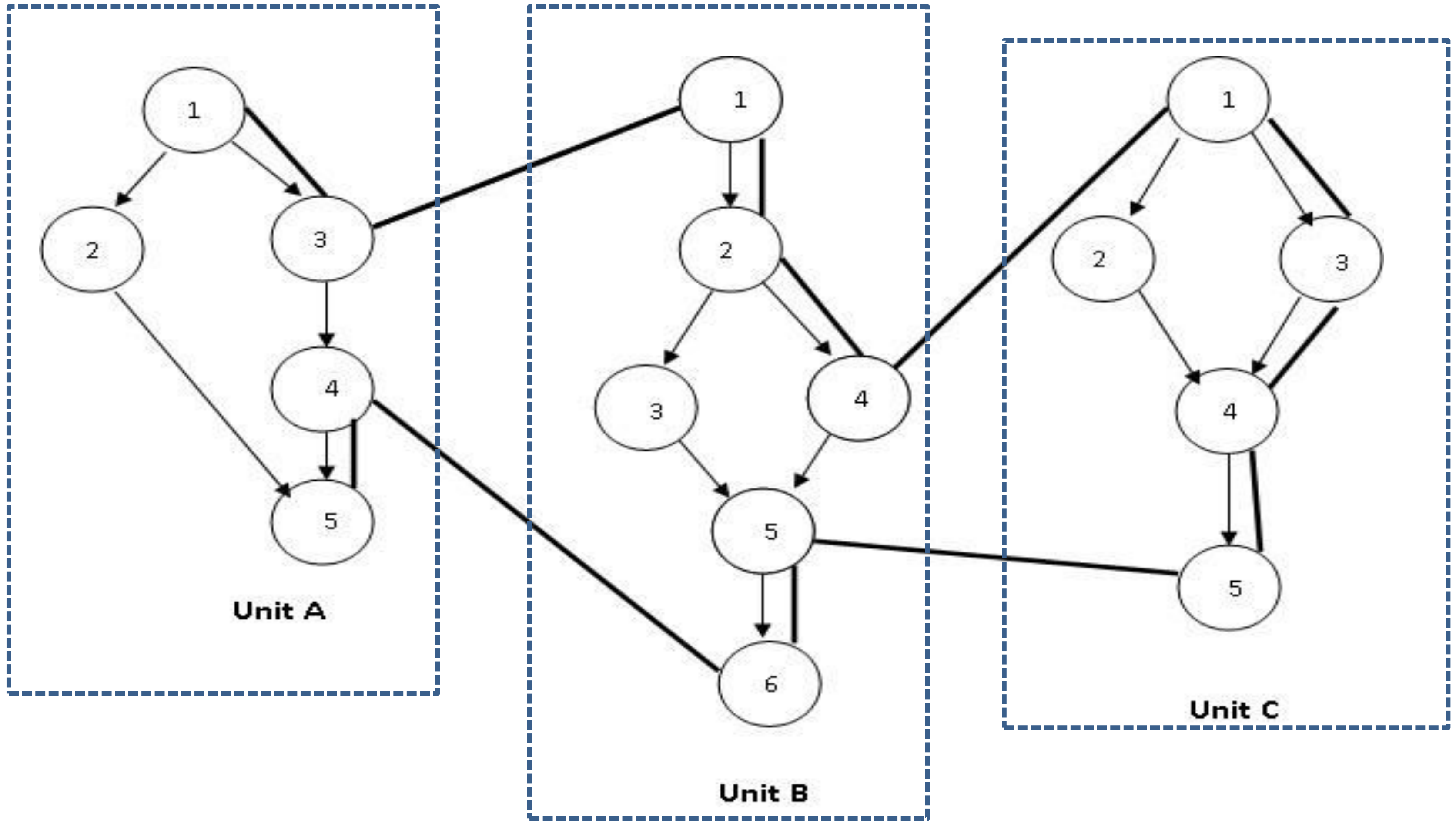
– Message
  - When the control is transferred from one unit to another, then the programming language mechanism used to do this is known as a message.
  - For example, a function call (message from one unit to another unit)

– MM-path
  - It is a path consisting of MEPs and messages.
  - The path shows the sequence of executable statements; it also crosses the boundary of a unit when a message is followed to call another unit.
  - In other words, MM-path is a set of MEPs & transfer of control among different units in the form of messages.

– MM-Path Graph

- It can be defined as an extended flow graph where nodes are MEPs and edges are messages.
- It returns from the last called unit to the first unit where the call was made.
- In this graph (shown in next slide), messages are highlighted with thick lines.

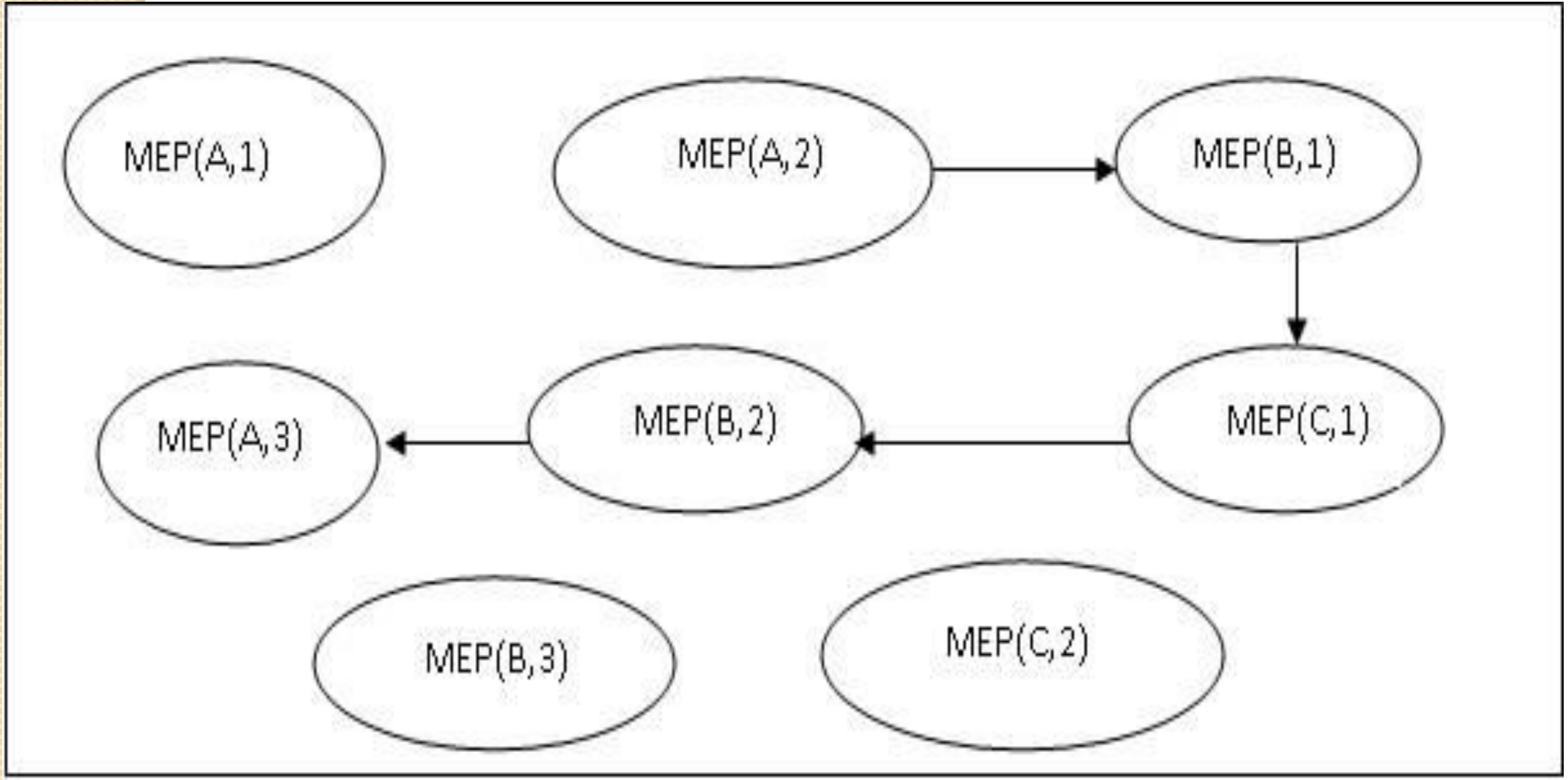
MM-Path



MM-Path Details

	Source nodes	Sink nodes	MEPs
Unit A	1,4	3,5	MEP(A,1)=<1,2,5> MEP(A,2)=<1,3> MEP(A,3)=<4,5>
Unit B	1,5	4,6	MEP(B,1)=<1,2,4> MEP(B,2)=<5,6> MEP(B,3)=<1,2,3,5,6>
Unit C	1	5	MEP(C,1)=<1,3,4,5> MEP(C,2)=<1,2,4,5>

MM-Path Graph



Function Testing

- When an integrated system is tested, all its specified functions and external interfaces are tested on the software.
- Every functionality of the system specified in the functions is tested according to its external specifications.
- An external specification is a precise description of the software behavior from the viewpoint of the outside world.

- Kit has defined function testing as the process of attempting to detect discrepancies between the functional specifications of a software and its actual behavior.
- The objective of the function test is to measure the quality of the functional components of the system.
- Tests verify that the system behaves correctly from the user/business perspectives and functions according to the requirements, models or any other design paradigm.

- The function test must determine if each component or business event:
  - Performs in accordance to the specifications,
  - Responds correctly to all conditions that may present themselves by incoming events/data,
  - Moves data correctly from one business event to the next, and
  - Is initiated in the order required to meet the business objectives of the system

- Function testing can be performed after unit and integration testing, or whenever the development team thinks that the system has sufficient functionality to execute some tests.
- The test cases are executed such that the execution of the given test case against the software will exercise external functionality of certain parts.
- To keep a record of function testing, a function coverage metric is used.

- Function coverage can be measured with a function coverage matrix.
- It keeps track of those functions that exhibited the greatest number of errors.
- An effective function test cycle must have a defined set of processes and deliverables.
- The primary processes/ deliverables for requirement based function test are as follows:

–Test Planning

- The test leader with assistance from the test team defines the scope, schedule, and deliverables for the function test cycle.
- He delivers a test plan and a test schedule- these often undergo several revisions during the testing cycle.

–Partitioning/ functional Decomposition

- It is the process of breaking down a system into its functional components or functional areas.
- Another group in the organization takes responsibility for the functional decomposition of the system.
- If decompositions are deemed insufficient, then testing organization takes up the responsibility of decomposition.

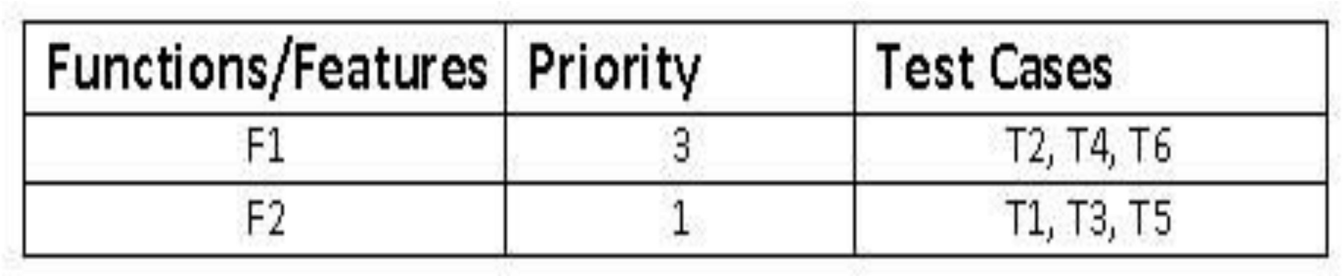
– Requirement definition
  - The testing organization needs specified requirements in the form of proper documents to proceed with the function test.
  - These requirements need to be itemized under an appropriate functional partition.

– Test case design
  - A tester designs and implements a test case to validate that the product performs in accordance with the requirements.
  - These requirements need to be itemized under an appropriate functional partition and mapped to the requirements being tested.

– Traceability matrix formation

- Test cases need to be mapped back to the appropriate requirement.
- A function coverage matrix is prepared.
- This matrix is a table, listing specific functions to be tested, the priority for testing each function, and test cases required to test each function.
- Once all the aspects of the function have been tested by one or more test cases, then the test design activity for that function can be considered complete.

Function coverage matrix



Functions/Features	Priority	Test Cases
F1	3	T2, T4, T6
F2	1	T1, T3, T5

–Test case execution

- In all phases of testing, an appropriate set of test cases need to be executed and the result of those test cases recorded.
- So, the test case to be executed should be defined in the test plan.
- If the current application does not support the testing, then it should be deferred.

Summary

- Discussed call graph-based integration testing in detail.
  - Pair-wise Integration
  - Neighborhood Integration
- Explained path-based integration testing.
- Explained Function Testing in detail.
- Described the primary processes/ deliverables for requirement based function test.

References

1. Rajib Mall, Fundamentals of Software Engineering, (Chapter – 10), Fifth Edition, PHI Learning Pvt. Ltd., 2018.
2. Naresh Chauhan, Software Testing: Principles and Practices, (Chapter – 7), Second Edition, Oxford University Press, 2016.

Thank You