

CSCE 747 Software Testing and Quality Assurance

Lecture 10 – Integration Testing

9/30/2013

1

Last Time

- Integration & System Testing Part III
- Levels of Testing Ch 12, pp 181-199

Today

- Integration & System Testing Part III
- Integration Testing Ch 13, pp 201-227

Table 13.1 SATM Units

Table 13.1 SATM Units and Abbreviated Names

<i>Unit Number</i>	<i>Level Number</i>	<i>Unit Name</i>			
1	1	SATM System	14	1.3.1	Screen Driver
A	1.1	Device Sense & Control	15	1.3.2	Key Sensor
D	1.1.1	Door Sense & Control	C	1.4	Manage Session
2	1.1.1.1	Get Door Status	16	1.4.1	Validate Card
3	1.1.1.2	Control Door	17	1.4.2	Validate PIN
4	1.1.1.3	Dispense Cash	18	1.4.2.1	GetPIN
E	1.1.2	Slot Sense & Control	F	1.4.3	Close Session
5	1.1.2.1	WatchCardSlot	19	1.4.3.1	New Transaction Request
6	1.1.2.2	Get Deposit Slot Status	20	1.4.3.2	Print Receipt
7	1.1.2.3	Control Card Roller	21	1.4.3.3	Post Transaction Local
8	1.1.2.4	Control Envelope Roller	22	1.4.4	Manage Transaction
9	1.1.2.5	Read Card Strip	23	1.4.4.1	Get Transaction Type
10	1.2	Central Bank Comm.	24	1.4.4.2	Get Account Type
11	1.2.1	Get PIN for PAN	25	1.4.4.3	Report Balance
12	1.2.2	Get Account Status	26	1.4.4.4	Process Deposit
13	1.2.3	Post Daily Transactions	27	1.4.4.5	Process Withdrawal
B	1.3	Terminal Sense & Control			

SATM Functional Decomposition

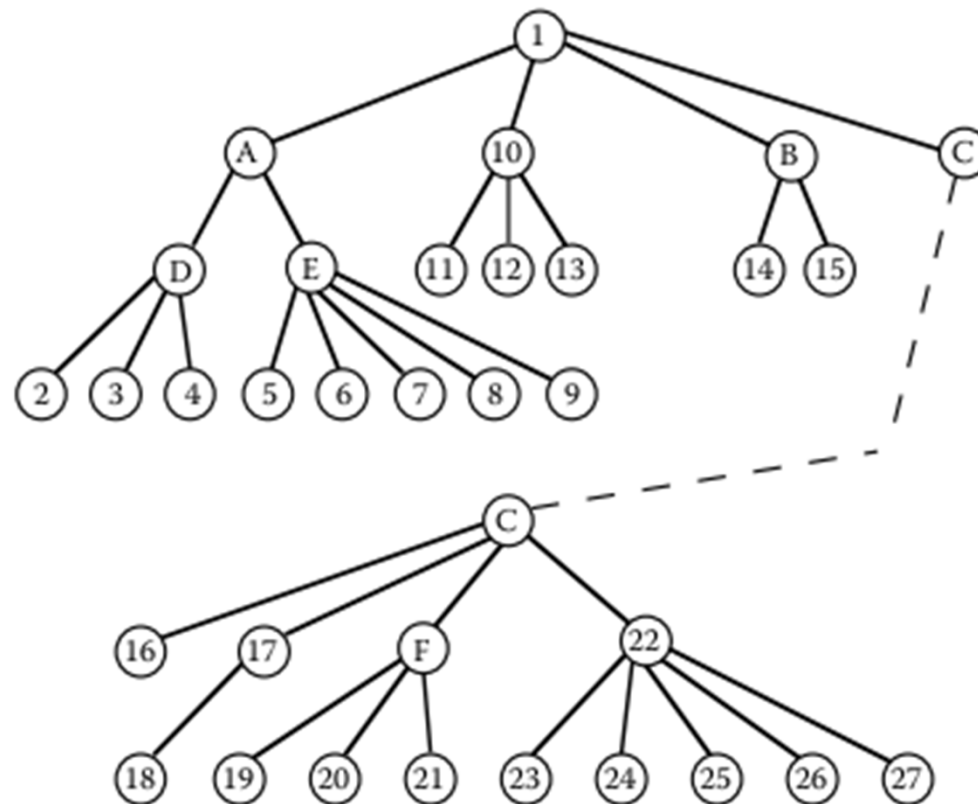
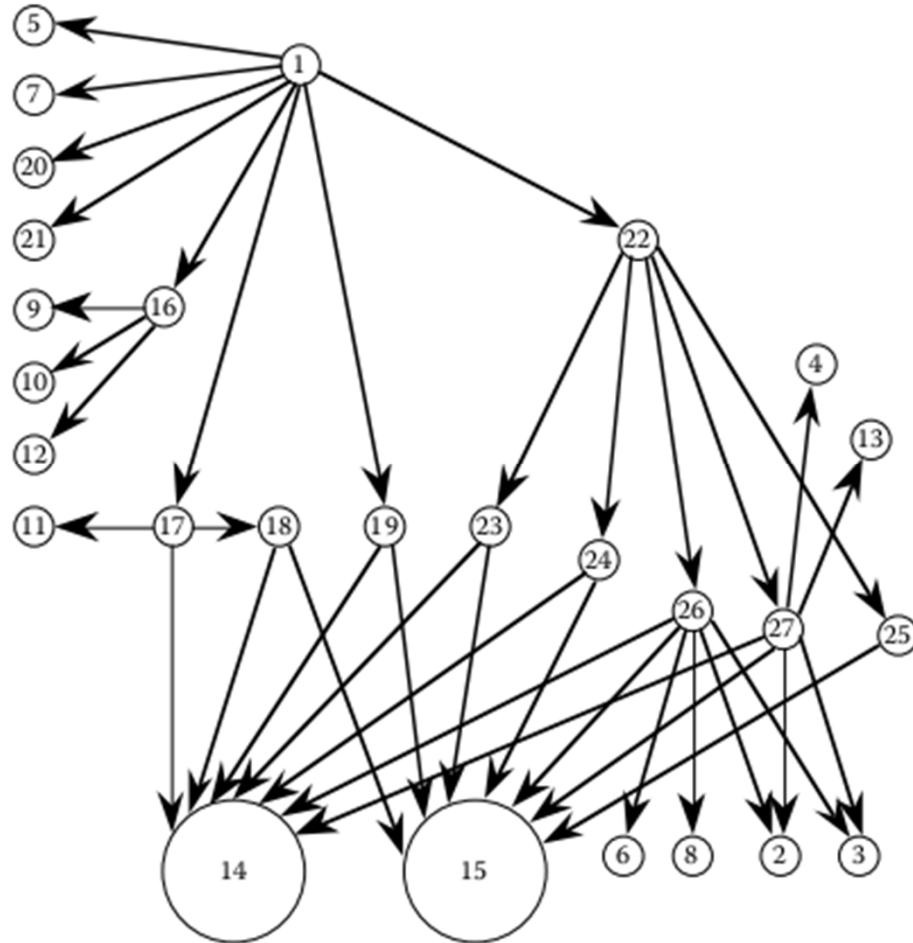


Figure 13.1 SATM functional decomposition tree.

Table 13.2 Adjacency Matrix for the SATM Call Graph

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
1				X		X									X	X		X	X	X	X					
2																										
3																										
4																										
5																										
6																										
7																										
8																										
9																										
10																										
11																										
12																										
13																										
14																										
15																										
16									X	X		X														
17										X			X				X									
18													X	X												
19													X	X												
20																										
21																										
22																						X	X	X	X	X
23													X	X												
24													X	X												
25														X												
26	X	X			X		X						X	X												
27	X	X	X		X		X					X	X	X												

SATM Call Graph



Stubs for Top Down Integration

```
Procedure GetPINforPAN(PAN, ExpectedPIN) STUB  
If PAN = '1123' Then ExpectedPIN = '8876'  
If PAN = '1234' Then ExpectedPIN = '8765'  
If PAN = '8746' Then ExpectedPIN = '1253'  
End
```

```
Procedure KeySensor(KeyHit) STUB  
data: KeyStrokes STACK OF '8', '8', '7', 'cancel'  
KeyHit = POP (KeyStrokes)  
End
```

Figure 13.3 Top-down Integration

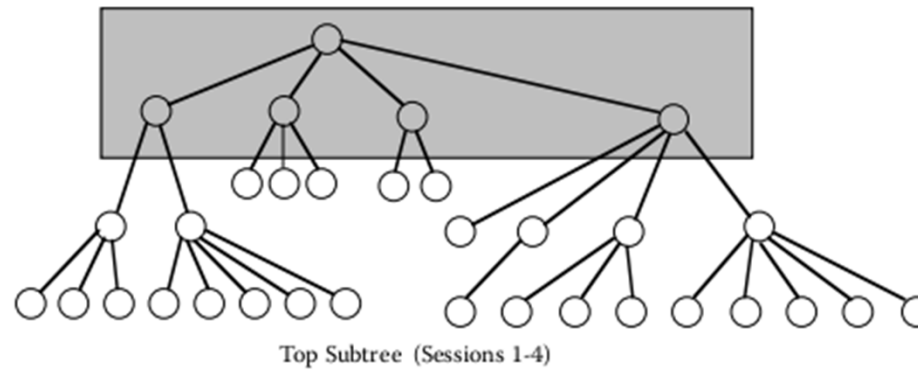


Figure 13.3 Top-down Integration

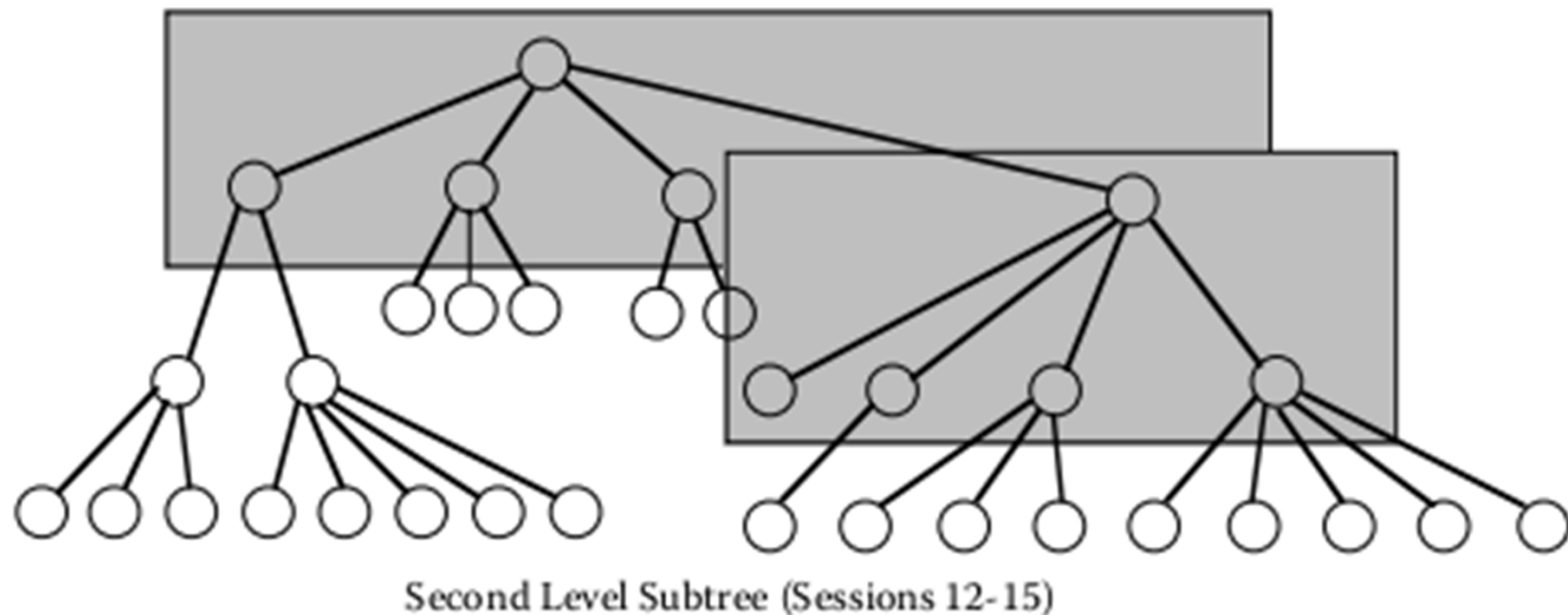
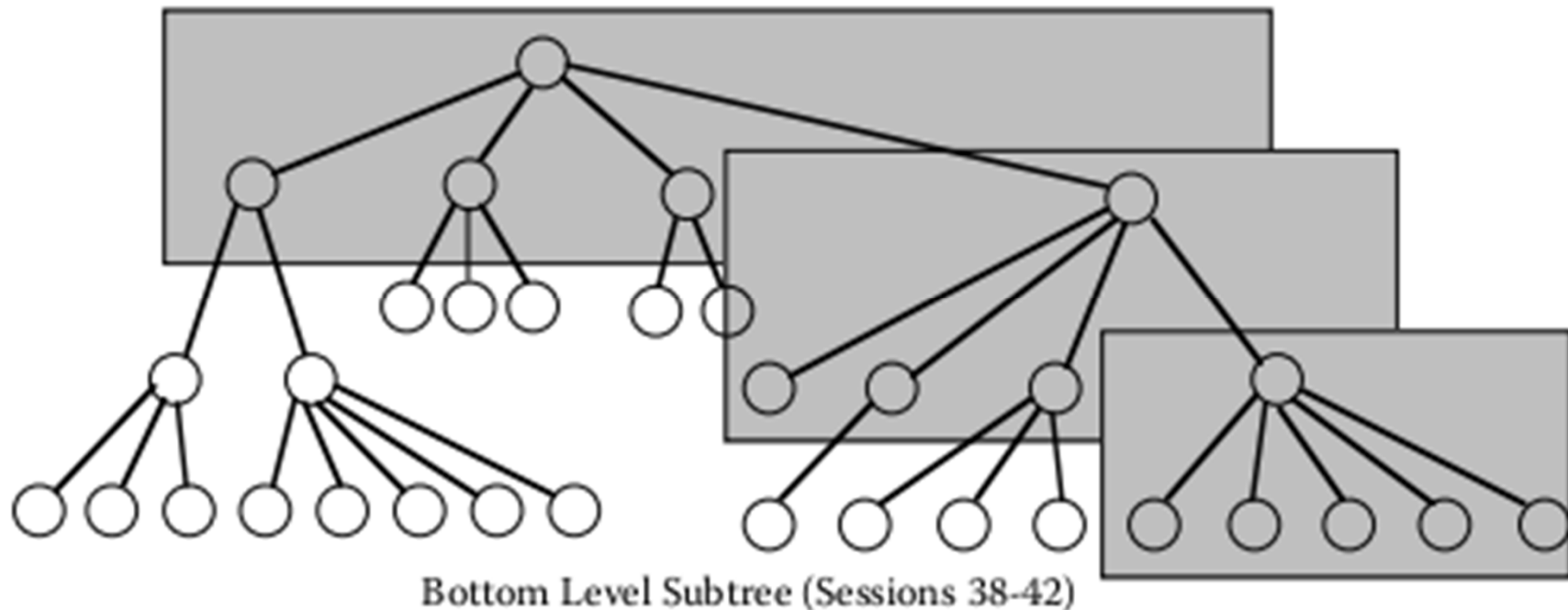
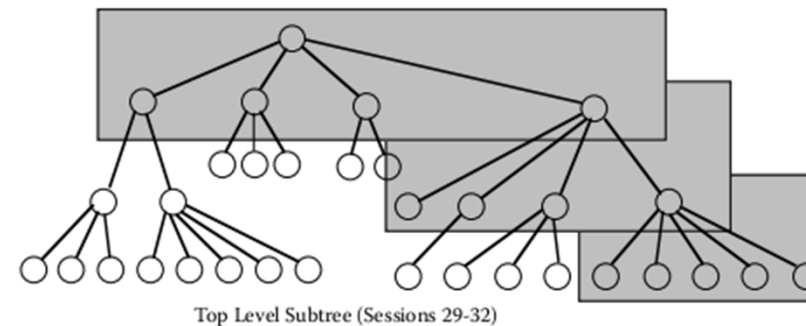
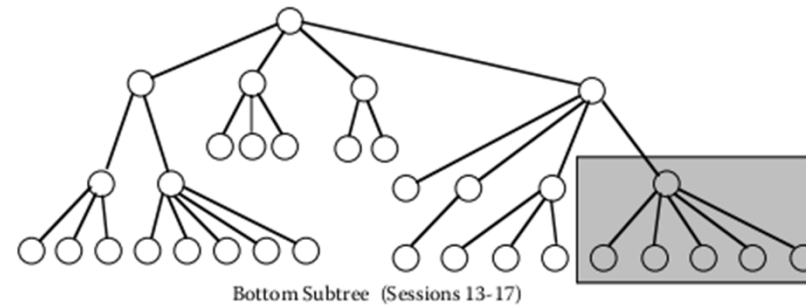


Figure 13.3 Top-down Integration

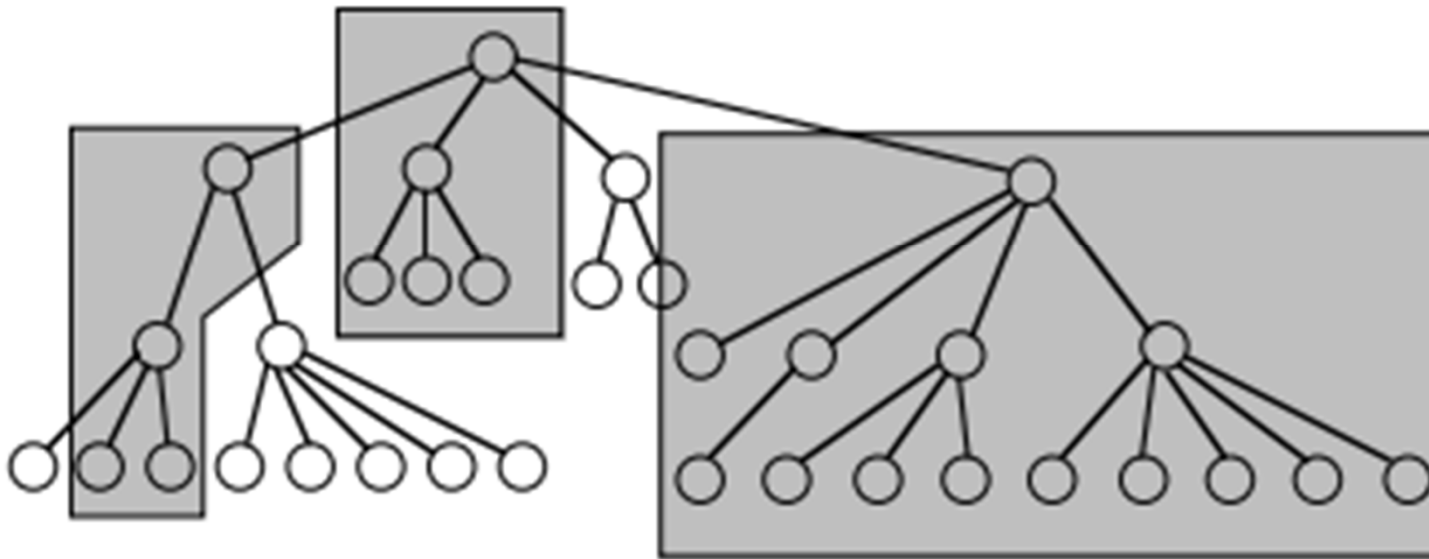


Bottom-Up Integration



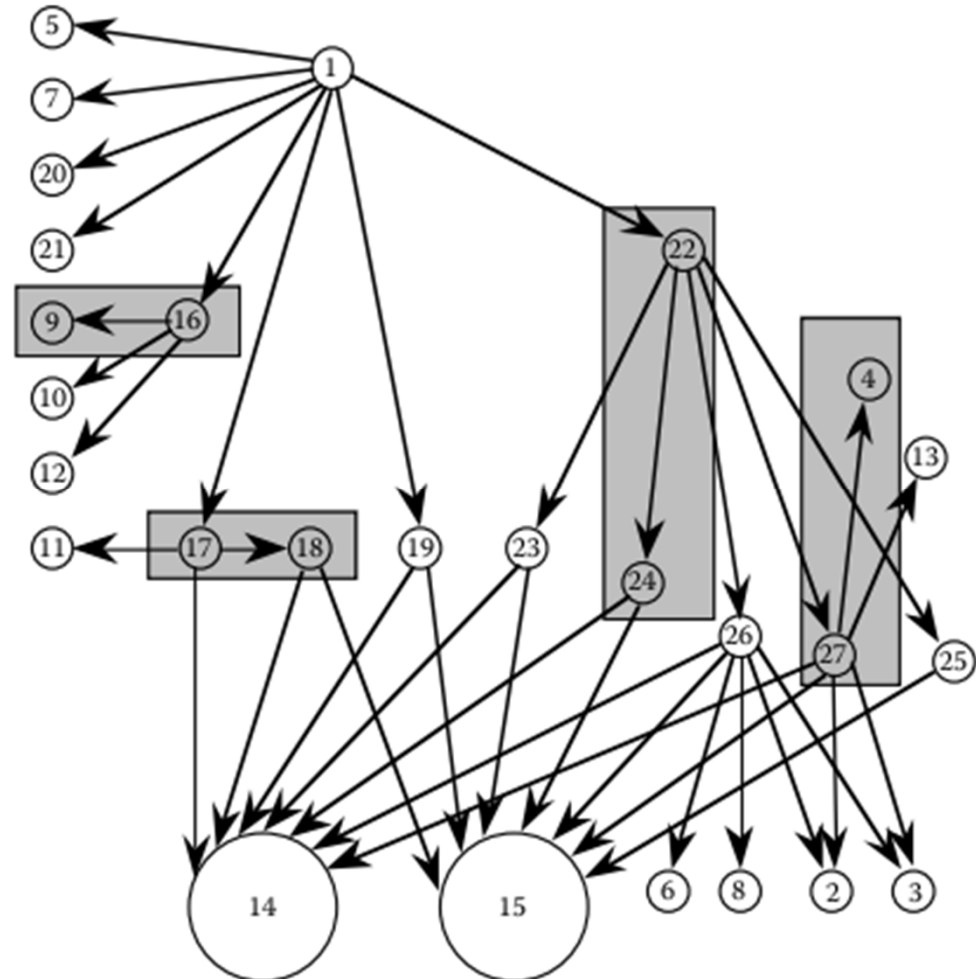
Sandwich Integration

- John Montagu, 4th Earl of Sandwich



Call Based Graph Integration

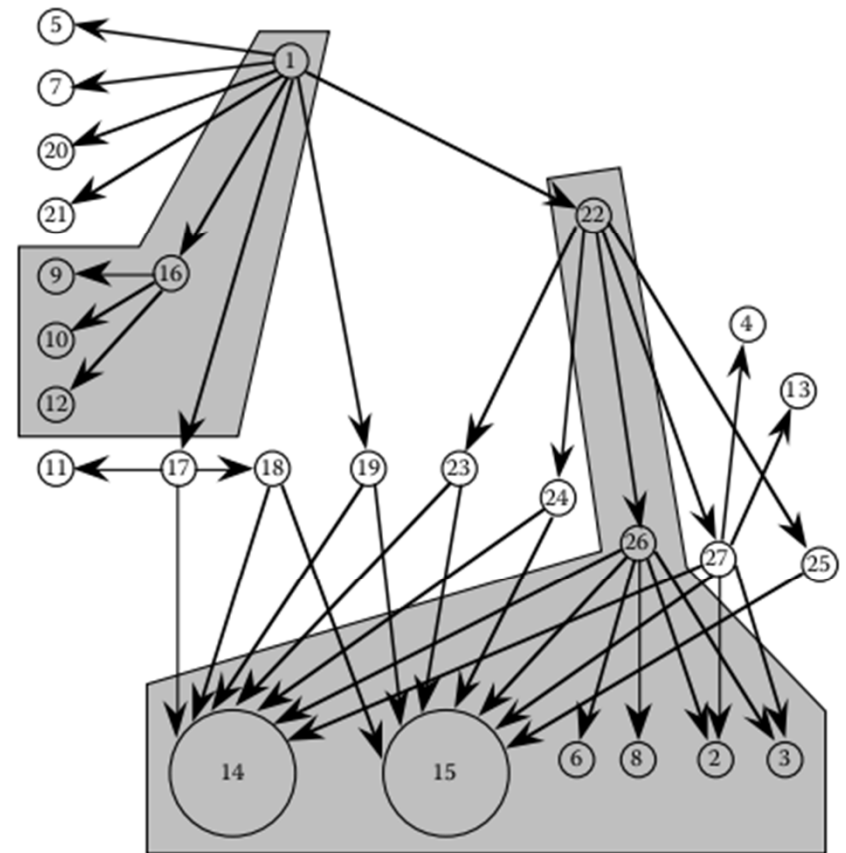
- Pairwise integration
- Neighborhood integration
 - InteriorNodes = nodes – sources – sinks
 - Neighborhoods
 - Interior+source node



Neighborhood Integration

Table 13.3 SATM Neighborhoods

<i>Node</i>	<i>Predecessors</i>	<i>Successors</i>
16	1	9, 10, 12
17	1	11, 14, 18
18	17	14, 15
19	1	14, 15
23	22	14, 15
24	22	14, 15
26	22	14, 15, 6, 8, 2, 3
27	22	14, 15, 2, 3, 4, 13
25	22	15
22	1	23, 24, 26, 27, 25
1	n/a	5, 7, 2, 21, 16, 17, 19, 22



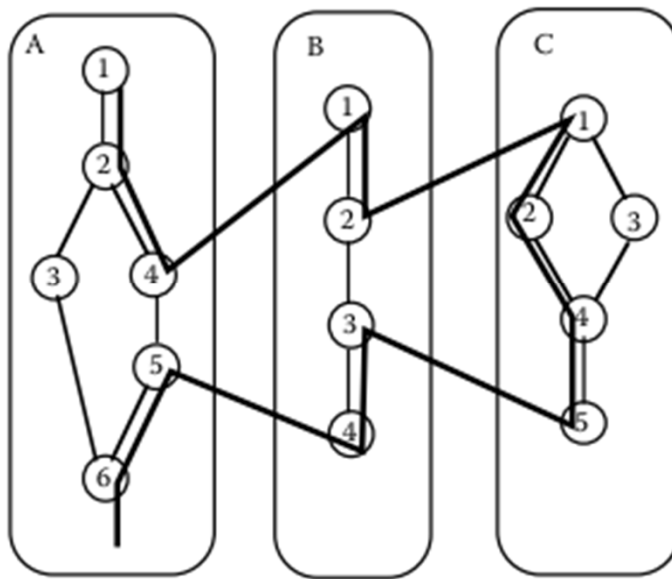
Path Based Integration

New and Extended Concepts

- Definition: A **source node** in a program is a statement at which program execution begins or resumes.
- Definition: A **sink node** in a program is a statement at which program execution stops.
- Definition: A **module execution path** is a sequence of statements that begins with a source node and ends with a sink node, with no intervening sink nodes.

- Definition: A **message** is a programming language mechanism by which one unit transfers control to another unit.
- Definition: An **MM-Path** is an interleaved sequence of module execution paths and messages.

Fig 13.8 MM-Paths across three Units



MEP(A,1) = <1, 2, 3, 6>

MEP(A,2) = <1, 2, 4>

MEP(A,3) = <5, 6>

MEP(B,1) = <1, 2>

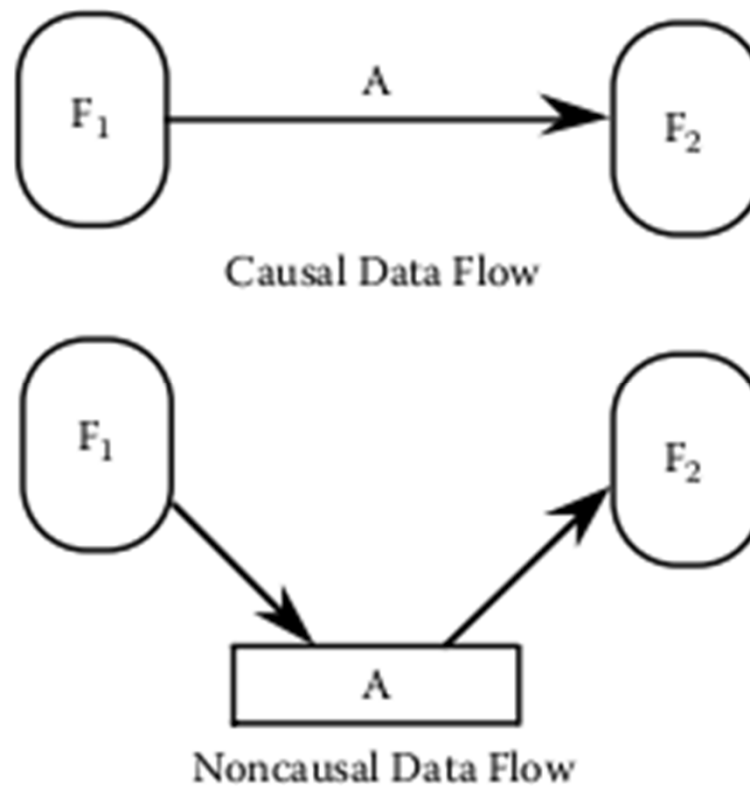
MEP(B,2) = <3, 4>

MEP(C,1) = <1, 2, 4, 5>

MEP(C,2) = <1, 3, 4, 5>

- **Definition:** Given a set of units, their **MM-Path graph** is the directed graph in which nodes are module execution paths and edges correspond to messages and returns from one unit to another.

Figure 13.10 Data Quiescence



SATM code

```
1. Main Program
2. State = AwaitCard
3. Do      'Main loop
4. Case State
5.   Case 1: AwaitCard
6.     ScreenDriver(1, null)           msg1
7.     WatchCardSlot(CardSlotStatus)  msg2
8.     Do While CardSlotStatus is Idle
9.       WatchCardSlot(CardSlotStatus) msg3
10.    End While
11.    ControlCardRoller(accept)       msg4
12.    ValidateCard(CardOK, PAN)       msg5
13.    If CardOK
14.      Then State = AwaitPIN
15.      Else ControlCardRoller(eject)  msg6
16.    EndIf
17.    State = AwaitCard
18. Case 2: AwaitPIN
19.   ValidatePIN(PINok, PAN)          msg7
```

20.	If PINok	-
21.	Then ScreenDriver(2, null)	msg8
22.	State = AwaitTrans	
23.	ElseScreenDriver(4, null)	msg9
24.	State = AwaitCard	
25.	EndIf	
26.	Case 3: AwaitTrans	
27.	ManageTransaction	msg10
28.	State = CloseSession	
29.	Case 4: CloseSession	
30.	If NewTransactionRequest	
31.	Then State = AwaitTrans	
32.	ElsePrintReceipt	msg11
33.	EndIf	
34.	PostTransactionLocal	msg12
35.	CloseSession	msg13
36.	ControlCardRoller(eject)	msg14
37.	State = AwaitCard	
38.	End Case (State)	
39.	Until 'Forever	



40.	End. (Main program SATM)	
41.	Procedure ValidatePIN(PINok, PAN)	
42.	GetPINforPAN(PAN, ExpectedPIN)	msg15
43.	Try = First	
44.	Case Try of	
45.	Case 1: First	
46.	ScreenDriver(2, null)	msg16
47.	GetPIN(EnteredPIN, CancelHit)	msg17
48.	If EnteredPIN = ExpectedPIN	
49.	Then PINok = True	
50.	Else ScreenDriver(3, null)	msg18
51.	Try = Second	
52.	EndIf	
53.	Case 2: Second	
54.	ScreenDriver(2, null)	msg19
55.	GetPIN(EnteredPIN, CancelHit)	msg20
56.	If EnteredPIN = ExpectedPIN	
57.	Then PINok = True	
58.	Else ScreenDriver(3, null)	msg21
59.	EndIf	
60.	Try = Third	

```

60.          Try = Third
61.      Case 3: Third
62.          ScreenDriver(2, null)                msg22
63.          GetPIN(EnteredPIN, CancelHit)         msg23
64.          If EnteredPIN = ExpectedPIN
65.              Then PINok = True
66.              Else ScreenDriver(4, null)         msg24
67.                  PINok = False
68.          EndIf
69.      EndCase (Try)
70.  End.      (Procedure ValidatePIN)

71.  Procedure GetPIN(EnteredPIN, CancelHit)
72.      Local Data: DigitKeys = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
73.      CancelHit = False
74.      EnteredPIN = null string
75.      digitsRcvd=0
76.      Do While NOT(DigitsRcvd=4 OR CancelHit)
77.          KeySensor(KeyHit)msg25
78.          If KeyHit IN DigitKeys
79.              Then
80.                  EnteredPIN = EnteredPIN + KeyHit
81.                  digitsRcvd = digitsRcvd + 1
82.                  If digitsRcvd = 1
83.                      Then ScreenDirver (2, 'X---') msg26
84.                  EndIf

```

```

85.         If digitsRcvd = 2
86.             Then ScreenDirver (2, 'XX--')    msg27
87.         EndIf
88.         If digitsRcvd = 3
89.             Then ScreenDirver (2, 'XXX-')    msg28
90.         EndIf
91.         If digitsRcvd = 4
92.             Then ScreenDirver (2, 'XXXX')    msg29
93.         EndIf
94.     Else
95.         CancelHit = True
96.     EndIf
97. End While
98. End. (Procedure GetPIN)

```

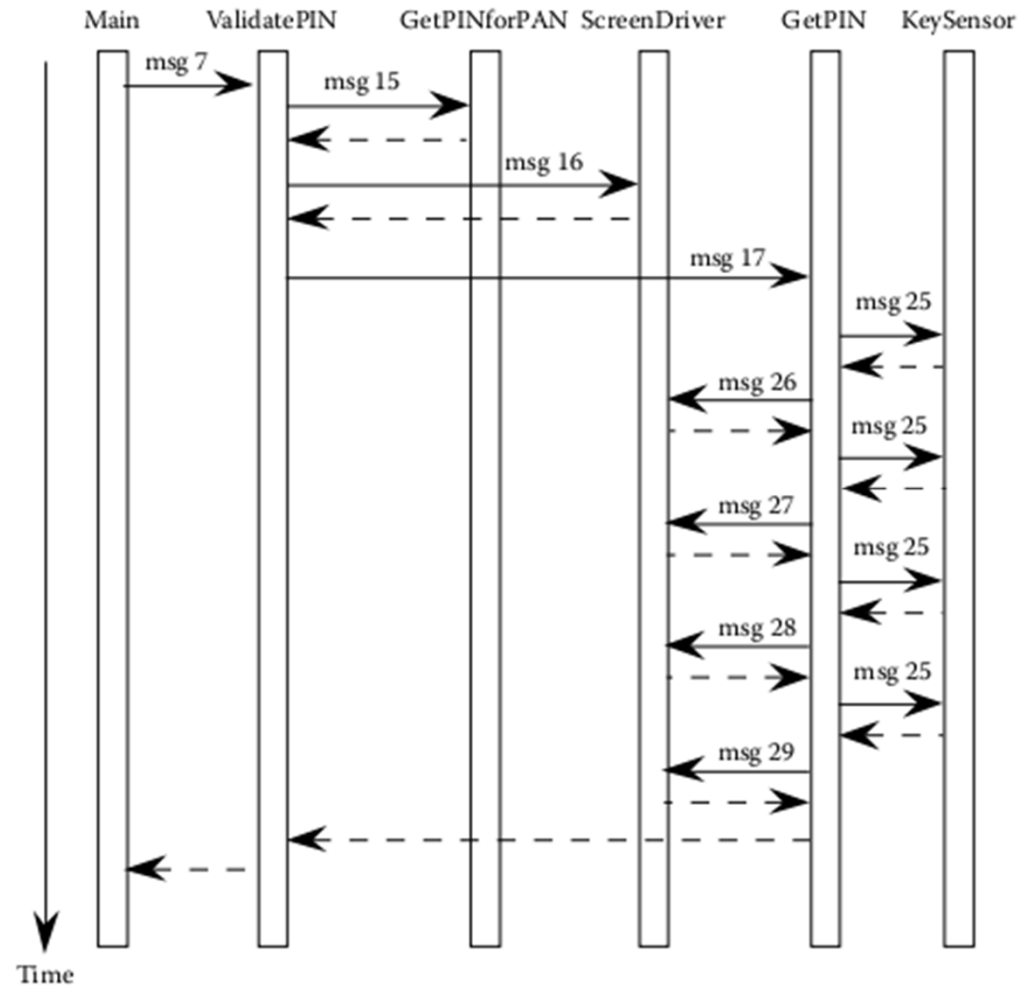
```
Main (1, 2, 3, 18, 19)
msg 7
ValidatePIN (41, 42)
msg 15
GetPINforPAN (no pseudo-code given)
ValidatePIN (43, 44, 45, 46)
msg 16
ScreenDriver (no pseudo-code given)
ValidatePIN (47)
msg 17
GetPIN(71, 72, 73, 74, 75, 76, 77)
msg 25
KeySensor (no pseudo-code given)  `first digit
GetPIN (78, 79, 80, 81, 82, 83)
msg 26
ScreenDriver (no pseudo-code given)
GetPIN (84, 85, 87, 88, 90, 91, 93, 96, 97, 76, 77)
```

```

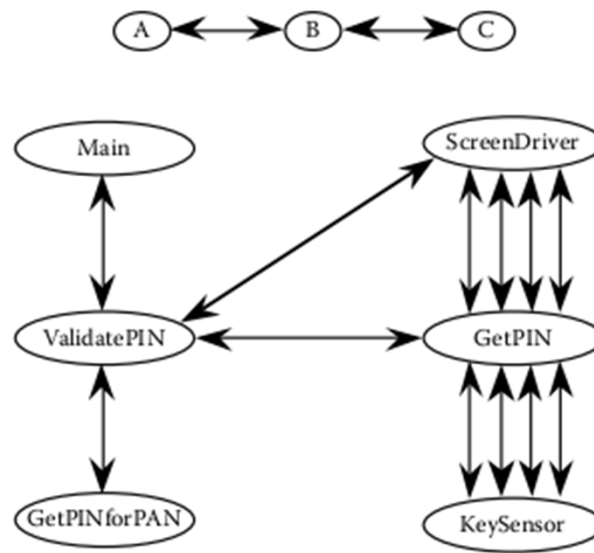
msg 25
KeySensor (no pseudo-code given) )      'second digit
GetPIN (78, 79, 80, 81, 82, 84, 85, 86)
msg 27
ScreenDriver (no pseudo-code given)
GetPIN (87, 88, 90, 91, 93, 96, 97, 76, 77)
msg 25
KeySensor (no pseudo-code given) )      'third digit
GetPIN (78, 79, 80, 81, 82, 84, 85, 87, 88, 89)
msg 28
ScreenDriver (no pseudo-code given)
GetPIN (90, 91, 93, 96, 97, 76, 77)
msg 25
KeySensor (no pseudo-code given) )      'fourth digit
GetPIN (78, 79, 80, 81, 82, 84, 85, 87, 88, 90, 91, 92)
msg 29
ScreenDriver (no pseudo-code given)
GetPIN (93, 96, 97, 76, 98)
ValidatePIN (48, 49, 52, 69, 70)
Main(20)

```

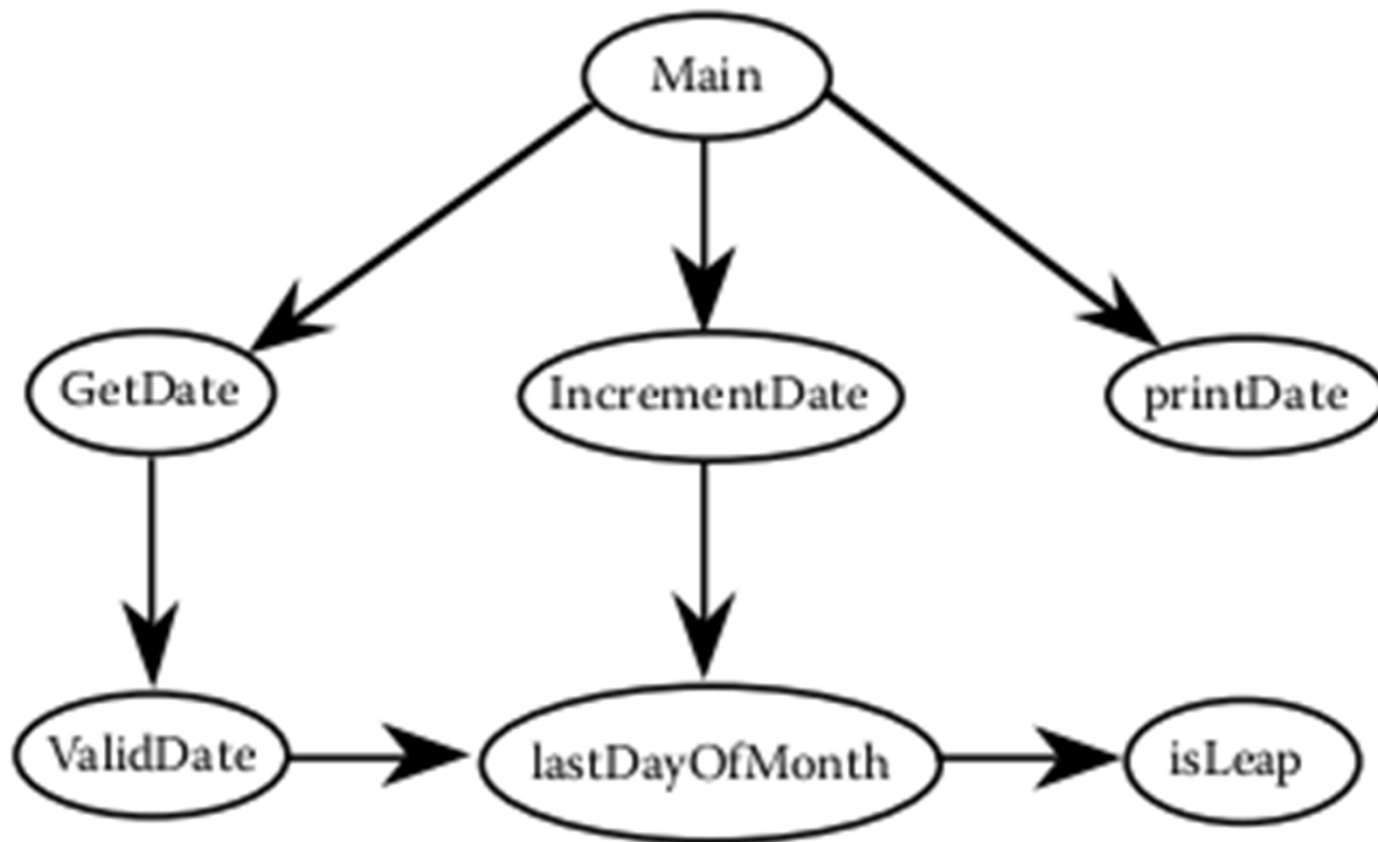
UML Sequence Diagram a sample MM-Path



MM-Path directed graphs



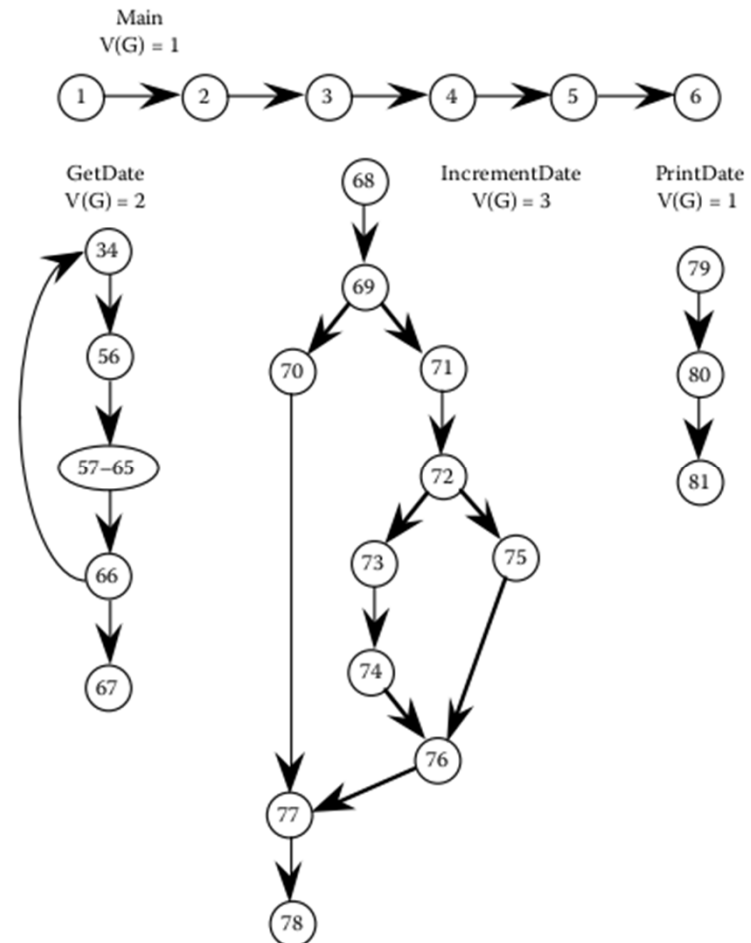
Call Graph of “Integration” Version



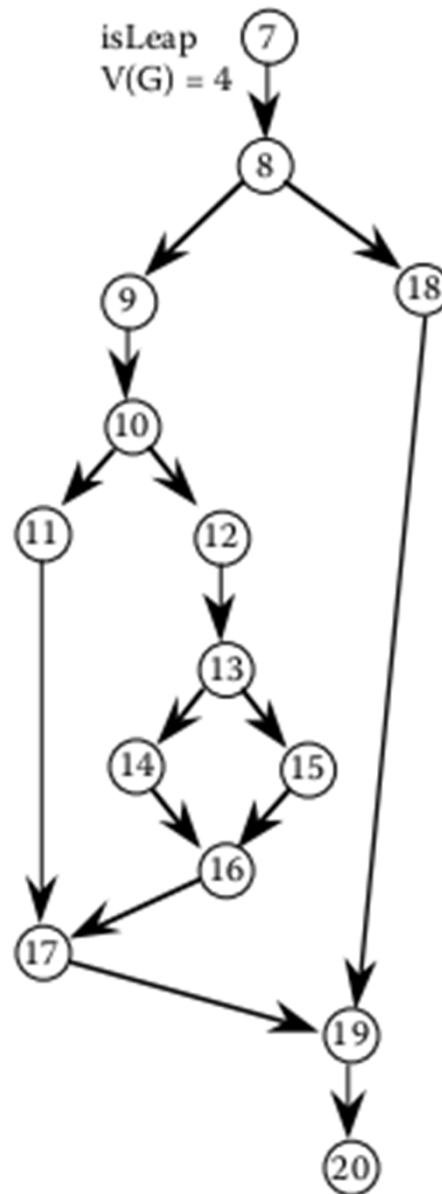
SATM Integration Version

■ Function

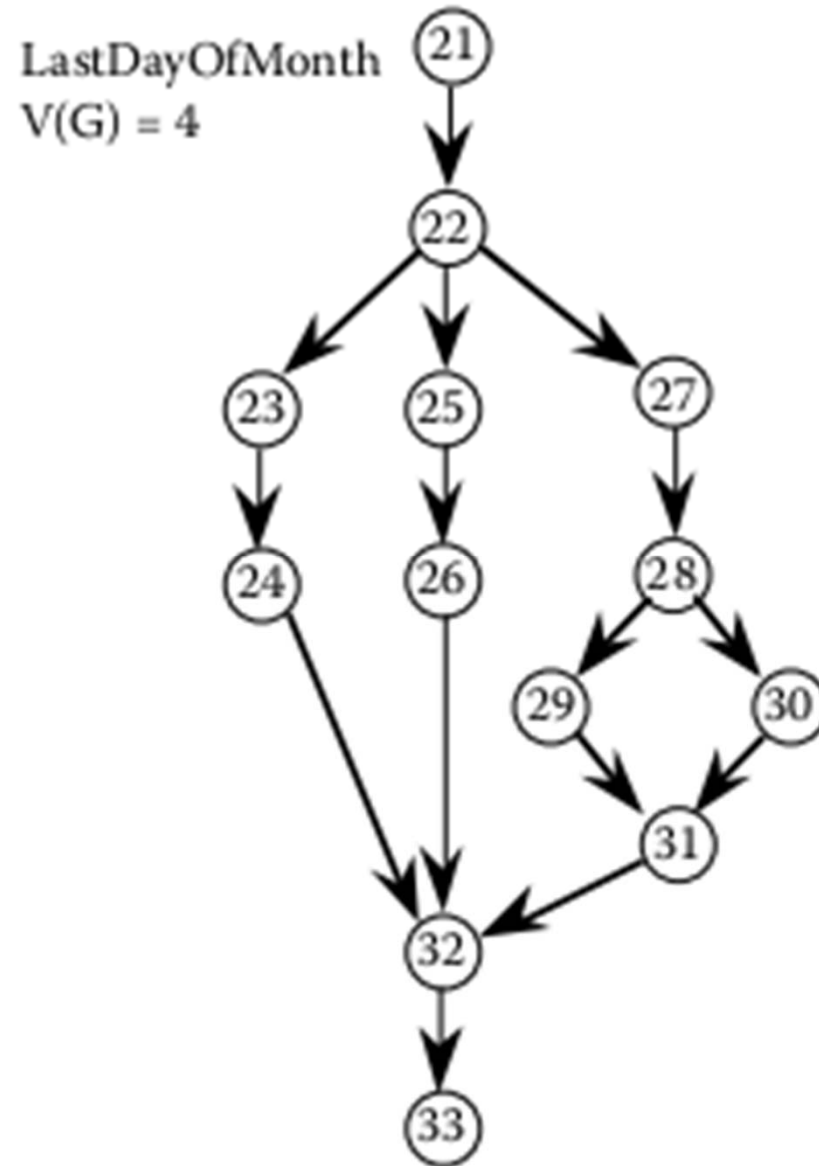
- `isLeapYear(year)`
- `lastDayOfMonth( )`
- `getDate()`



isLeap()

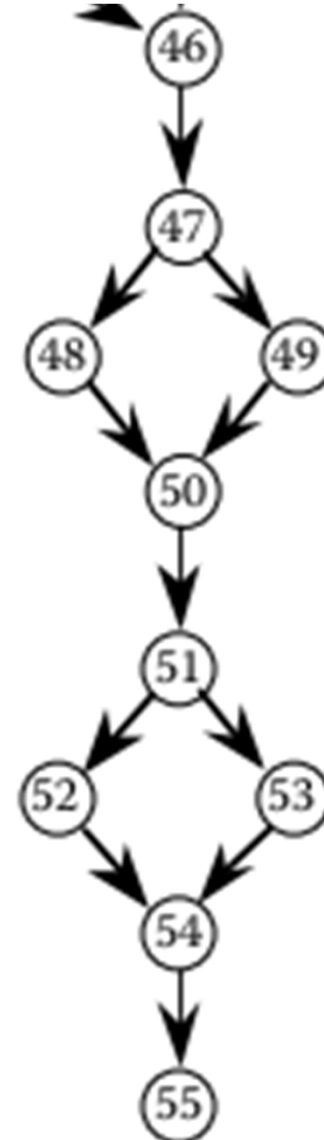
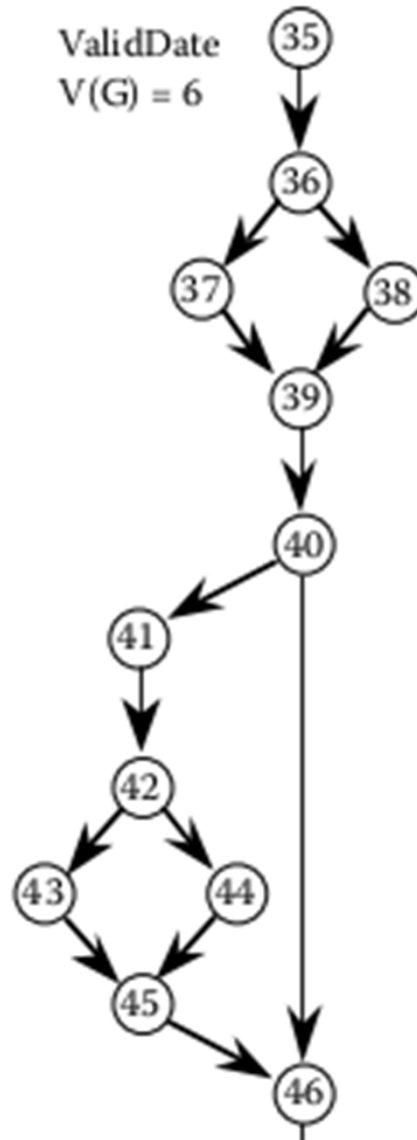


lastDayOfMonth



validDate()

ValidDate
 $V(G) = 6$



Decomposition Based Integration

Call Graph Based Integration

MM-Path-Based Integration

```
Main (1, 2)
  msg1
  GetDate (34, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66)
    msg7
    ValidateDate (35, 36, 37, 39, 40, 41, 42))
      msg6
      lastDayOfMonth (21, 22, 23, 24, 32, 33)
        'point of
        message quiescence
      ValidateDate (43, 45, 46, 47, 48, 50, 51, 52, 54, 55)
    GetDate (67)
Main (3)
```

Table 13.4 Comparison of Integration Testing Strategies

<i>Strategy Basis</i>	<i>Ability to Test Interfaces</i>	<i>Ability to Test Co-Functionality</i>	<i>Fault Isolation Resolution</i>
Functional decomposition	Acceptable but can be deceptive	Limited to pairs of units	Good, to faulty unit
Call graph	Acceptable	Limited to pairs of units	Good, to faulty unit
MM-Path	Excellent	Complete	Excellent, to faulty unit execution path

References

- Deutsch, M.S., *Software Verification and Validation-Realistic Project Approaches*, Prentice-Hall, Englewood Cliffs, NJ, 1982.
- Fordahl, M., Elementary Mistake Doomed Mars Probe, Associated Press, October 1, 1999, www.fas.org/mars/991001/-mars01.htm.
- Hetzel, B., *The Complete Guide to SOFTWARE TESTING*, 2nd ed., QED Information Sciences, Inc., Wellesley, MA, 1988.
- Jorgensen, P.C., The Use of MM-Paths in Constructive Software Development, Ph.D. dissertation, Arizona State University, Tempe, 1985.
- Jorgensen, P.C. and Erickson, C., Object-oriented integration testing, *Communications of the ACM*, September 1994.
- Kaner, C., Falk, J., and Nguyen, H.Q., *Testing Computer Software*, 2nd ed., Van Nostrand Reinhold, New York, 1993.
- Mosley, D.J., *The Handbook of MIS Application Software Testing*, Yourdon Press, Prentice-Hall, Englewood Cliffs, NJ, 1993.