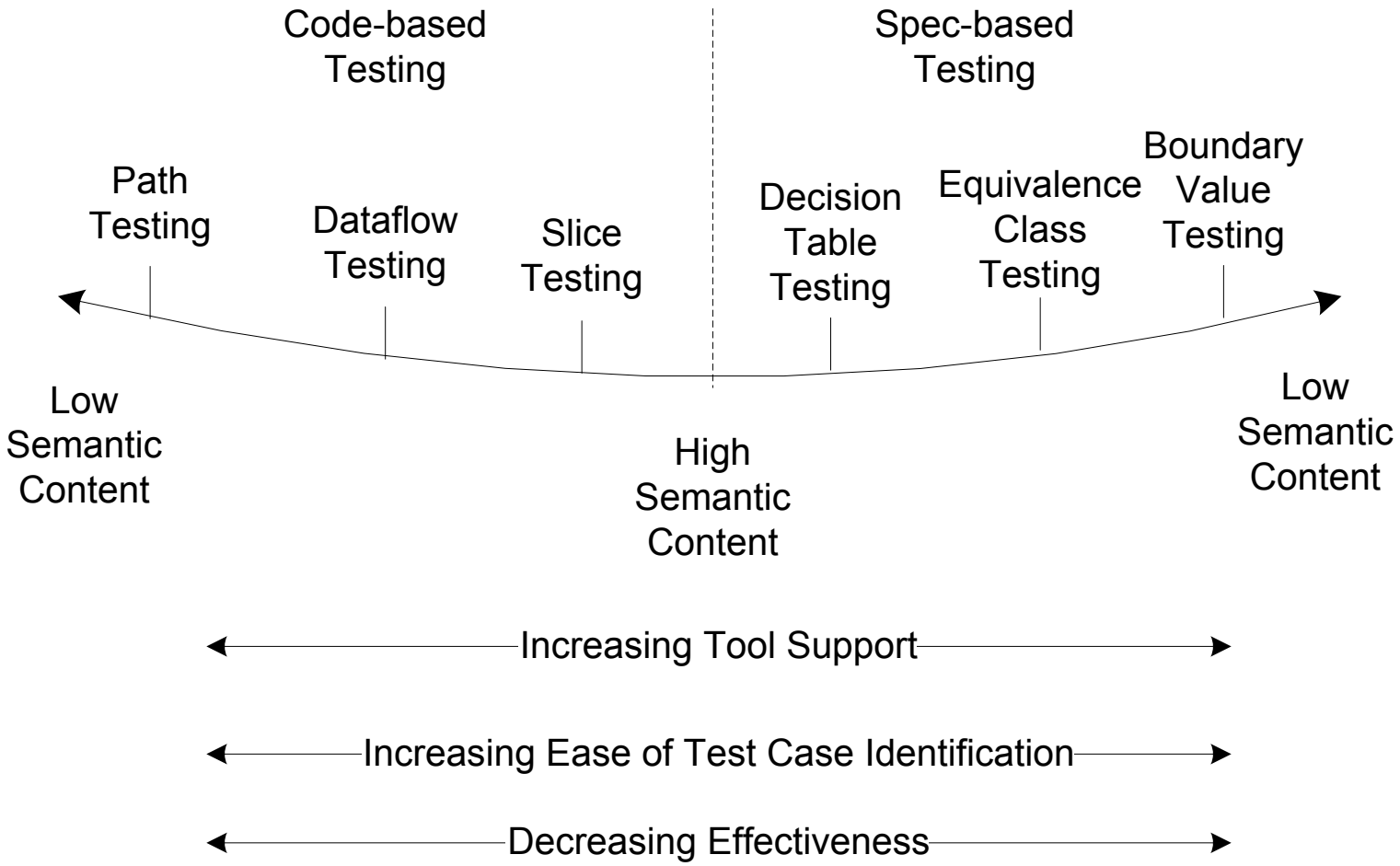
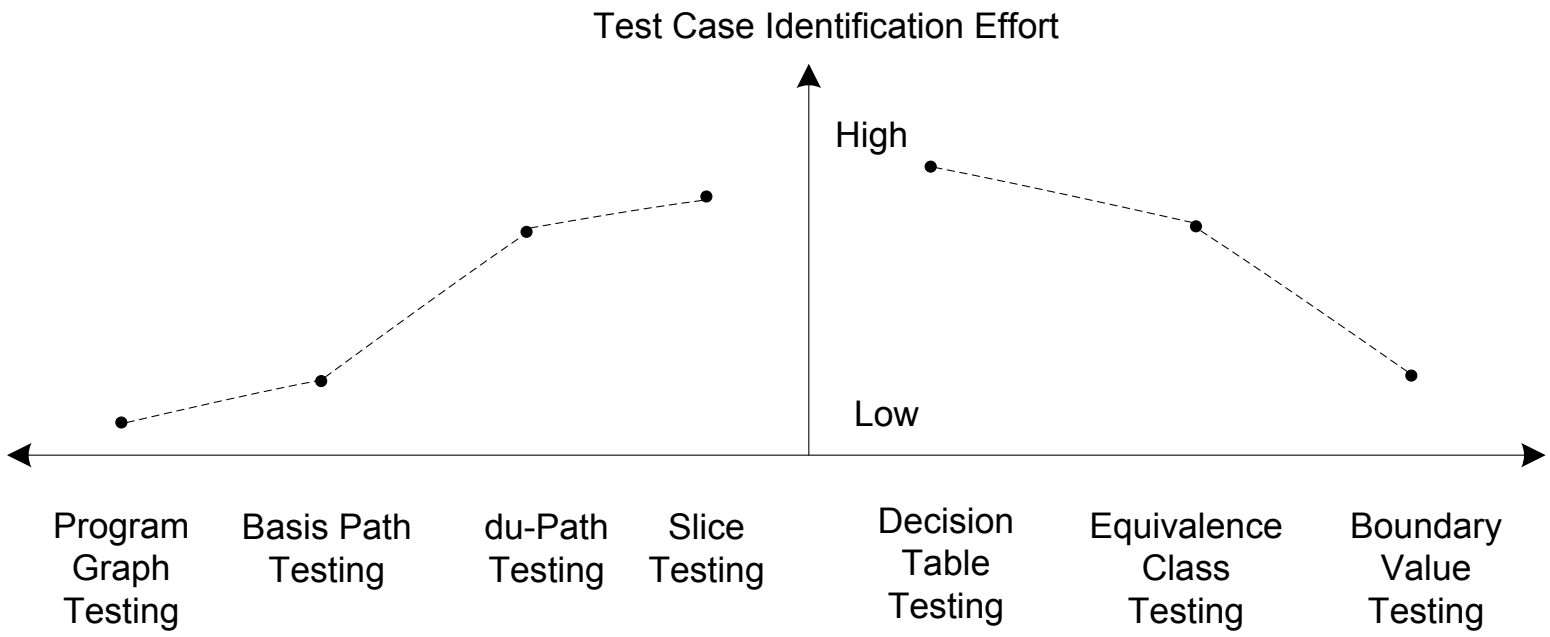
Chapter 10

Retrospective on Unit Testing

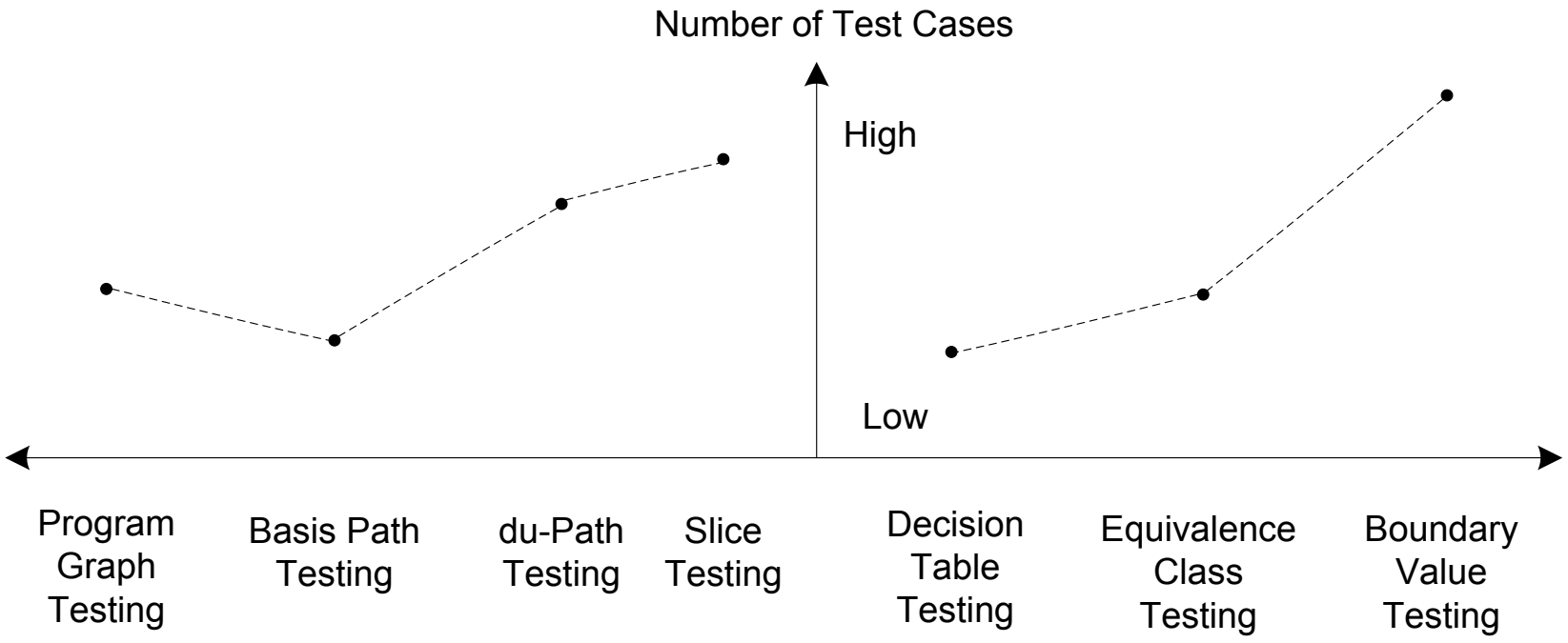
The Test Method Pendulum



Test Case Identification Effort



Number of Test Cases



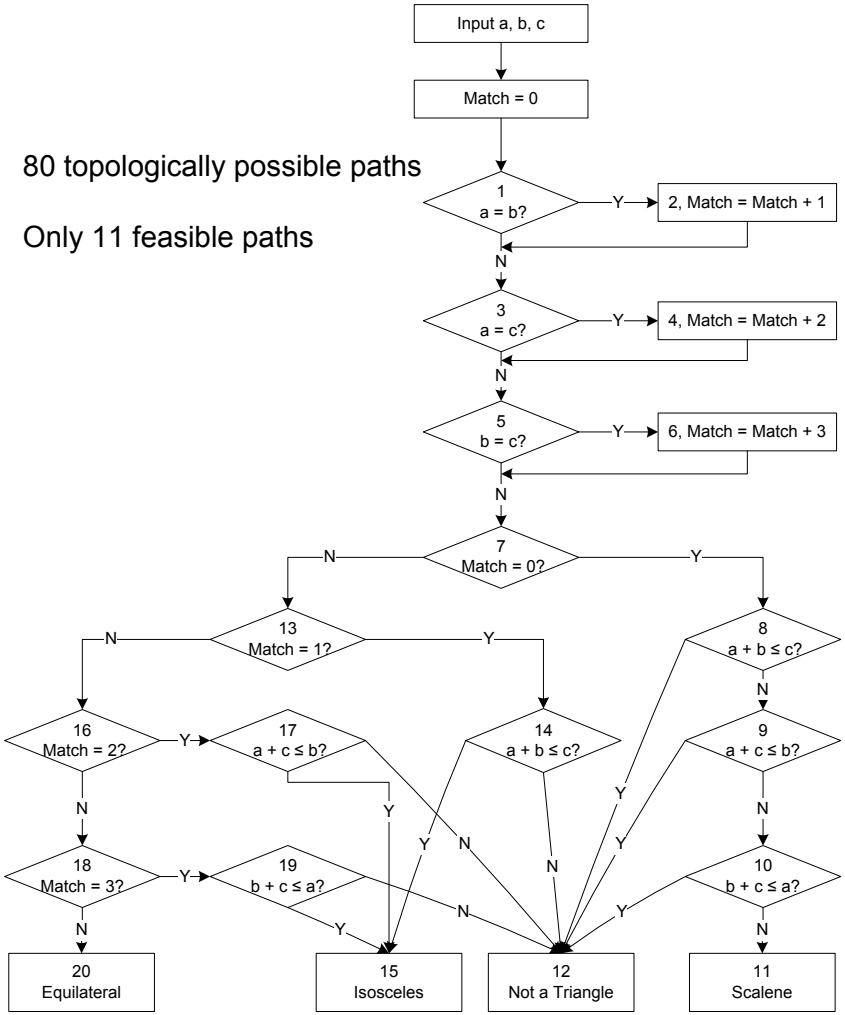
Traversing the Code-Based Side

- FORTRAN version of the Triangle Program
- Problems with path testing
- Basis path (McCabe) testing?
- Dataflow testing
- Slice testing
  - 80 topologically possible paths
  - Only 11 feasible paths
  - Cyclomatic complexity is 14
  - Basis path testing? (expect 14 basis paths, but only 11 paths are feasible)
  - Dataflow testing?
  - Slice testing?

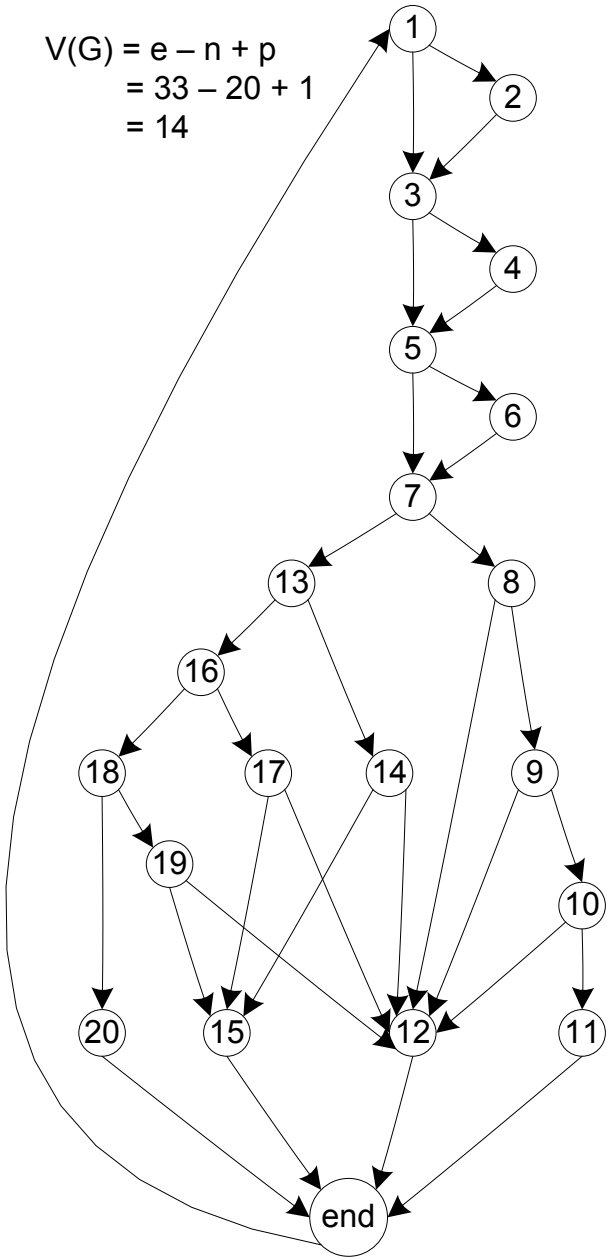
FORTRAN version of the Triangle Program

80 topologically possible paths

Only 11 feasible paths



Program Graph
(V(G) = 14)



Feasible Paths in the FORTRAN Version

Path	*Node Sequence*	*Description*
p1	1–2–3–4–5–6–7–13–16–18–20	Equilateral
p2	1–3–5–6–7–13–16–18–19–15	Isosceles (b = c)
p3	1–3–5–6–7–13–16–18–19–12	Not a Triangle (b = c)
p4	1–3–4–5–7–13–16–17–15	Isosceles (a = c)
p5	1–3–4–5–7–13–16–17–12	Not a Triangle (a = c)
p6	1–2–3–5–7–13–14–15	Isosceles (a = b)
p7	1–2–3–5–7–13–14–12	Not a Triangle (a = b)
p8	1–3–5–7–8–12	Not a Triangle (a + b ≤ c)
p9	1–3–5–7–8–9–12	Not a Triangle (b + c ≤ a)
p10	1–3–5–7–8–9–10–12	Not a Triangle (a + c ≤ b)
p11	1–3–5–7–8–9–10–11	Scalene

Basis Path Testing?

- Problematic!
- V(G) = 14 suggests 14 basis paths
- Only 11 logically feasible paths
- “The Method” generates infeasible paths

Dataflow Testing?

- Moving closer to "semantic reality"
- Implications of the match variable
  - FORTRAN style to reduce decisions
  - Only values of match are 0, 1, 2, 3, and 6
- The match variable has
  - four definition nodes
  - three computation uses
  - four predicate uses
  - 28 define/use paths
- The "side" variables a, b, and c have
  - 1 definition node
  - 9 use nodes
  - 9 definition-clear paths to test
- 37 dataflow test cases

Slice Testing?

- Even closer to “semantic reality”
- Make backward slices on
  - scalene at node 11
  - notATriangle at node 12
  - isosceles at node 15
  - equilateral at node 20

Slice Test Cases

- 3 test cases for S(scalene,11)
  - (3, 4, 5)
  - (4, 5, 3)
  - (5, 3, 4)
- 1 test case for S(equilateral,20)
- 3 test cases for S(isosceles,15)
- 18 test cases for S(notATriangle, 12)
  - six ways to fail the triangle test
  - each way has three permutations, as for S(scalene, 11)
- 25 slice-based test cases
- Compare with equivalence class test cases

Traversing the Specification-Based Side

- Boundary Value Testing
  - nominal values: 15 test cases
  - worst case values: 125 test cases
- Equivalence Class Testing
  - 11 equivalence classes
  - correspond exactly to the 11 feasible paths
- Decision Table Testing
  - 20 possible rules
  - 2 impossible rules
  - provides insight into the match variable

Path Coverage of Nominal Boundary Values

Case	*a*	*b*	*c*	*Expected*	*Path*
1	100	100	1	Isosceles	p6
2	100	100	2	Isosceles	p6
3	100	100	100	Equilateral	p1
4	100	100	199	Isosceles	p6
5	100	100	200	Not a Triangle	p7
6	100	1	100	Isosceles	p4
7	100	2	100	Isosceles	p4
8	100	100	100	Equilateral	p1
9	100	199	100	Isosceles	p4
10	100	200	100	Not a Triangle	p5
11	1	100	100	Isosceles	p2
12	2	100	100	Isosceles	p2
13	100	100	100	Equilateral	p1
14	199	100	100	Isosceles	p2
15	200	100	100	Not a Triangle	p3

Path Coverage of Nominal and Worst Case Boundary Value Test Cases

	p1	*p2*	*p3*	*p4*	*p5*	*p6*	*p7*	*p8*	*p9*	*p10*	*p11*
Nominal	3	3	1	3	1	3	1	0	0	0	0
Worst-case	5	12	6	11	6	12	7	17	18	19	12

Equivalence Class Testing

Classes from Chapter 6:

D1 = {<a, b, c> : a = b = c}
D2 = {<a, b, c> : a = b, a ≠ c}
D3 = {<a, b, c> : a = c, a ≠ b}
D4 = {<a, b, c> : b = c, a ≠ b}
D5 = {<a, b, c> : a ≠ b, a ≠ c, b ≠ c}
D6 = {<a, b, c> : a > b + c}
D7 = {<a, b, c> : b > a + c}
D8 = {<a, b, c> : c > a + b}
D9 = {<a, b, c> : a = b + c}
D10 = {<a, b, c> : b = a + c}
D11 = {<a, b, c> : c = a + b}

These classes will exactly execute the 11 feasible paths.

Decision Table Testing (continued)

c1. Match =	0				1				2			
c2. a + b <= c?	T	F!	F!	F	T	F!	F!	F	T	F!	F!	F
c3. a + c <= b?	F!	T	F!	F	F!	T	F!	F	F!	T	F!	F
c4. b + c <= a?	F!	F!	T	F	F!	F!	T	F	F!	F!	T	F
a1. Scalene				x								
a2. Not a Triangle	x	x	x		x	x	x		x	x	x	
a3. Isosceles								x				x
a4. Equilateral												
a5. Impossible												

(Extended Entry) Decision Table Testing

c1. Match =	3				4	5	6			
c2. a + b <= c?	T	F!	F!	F	--	--	T	F!	F!	F
c3. a + c <= b?	F!	T	F!	F	--	--	F!	T	F!	F
c4. b + c <= a?	F!	F!	T	F	--	--	F!	F!	T	F
a1. Scalene										
a2. Not a Triangle	x	x	x				x	x	x	
a3. Isosceles				x						
a4. Equilateral										x
a5. Impossible					x	x				

Notes on the EEDT

- the “match” variable notes which two sides (if any) are equal
- we are stuck with some reverse logic on the triangle inequalities
- there is a typo in Table 10.4. In the rule where
  - match = 1
  - c2 and c3 are F!
  - c4 = F is an isosceles triangle
- In the rule before that, the two x entries say that something is both Not a Trialgle and Isosceles. (Oppss!)

Redundancy Metrics Definitions

- Given a test method M that generates m test cases and a coverage metric S that defines s test cases for n structural elements...
  - The coverage of a methodology M with respect to a metric S, denoted C(M,S), is ratio of n to s.
  - The redundancy of a methodology M with respect to a metric S, denoted R(M,S), is ratio of m to s.
  - The net redundancy of a methodology M with respect to a metric S, denoted, NR(M,S) is ratio of m to n.

Redundancy Metrics for the Triangle Program

Method	*m*	*n*	*s*	*C(M, S) = n/s*	*R(M, S) = m/s*	*NR(M, S) = m/n*
Nominal	15	7	11	0.64	1.36	2.14
Worst-case	125	11	11	1	11.36	11.36
Goal	s	s	s	1	1	1

Case Study

A hypothetical Insurance Premium Program computes the semi-annual car insurance premium based on two parameters: the policy holder's age and driving record:

Premium = BaseRate*ageMultiplier – safeDrivingReduction

The ageMultiplier is a function of the policy holder's age, and the safe driving reduction is given when the current points (assigned by traffic courts for moving violations) on the policy holder's driver's license are below an age-related cutoff. Policies are written for drivers in the age range of 16 to 100. Once a policy holder has 12 points, his/her driver's license is suspended (hence there is no need for insurance). The BaseRate changes from time to time; for this example, it is $500 for a semi-annual premium.

Insurance Premium Program Data

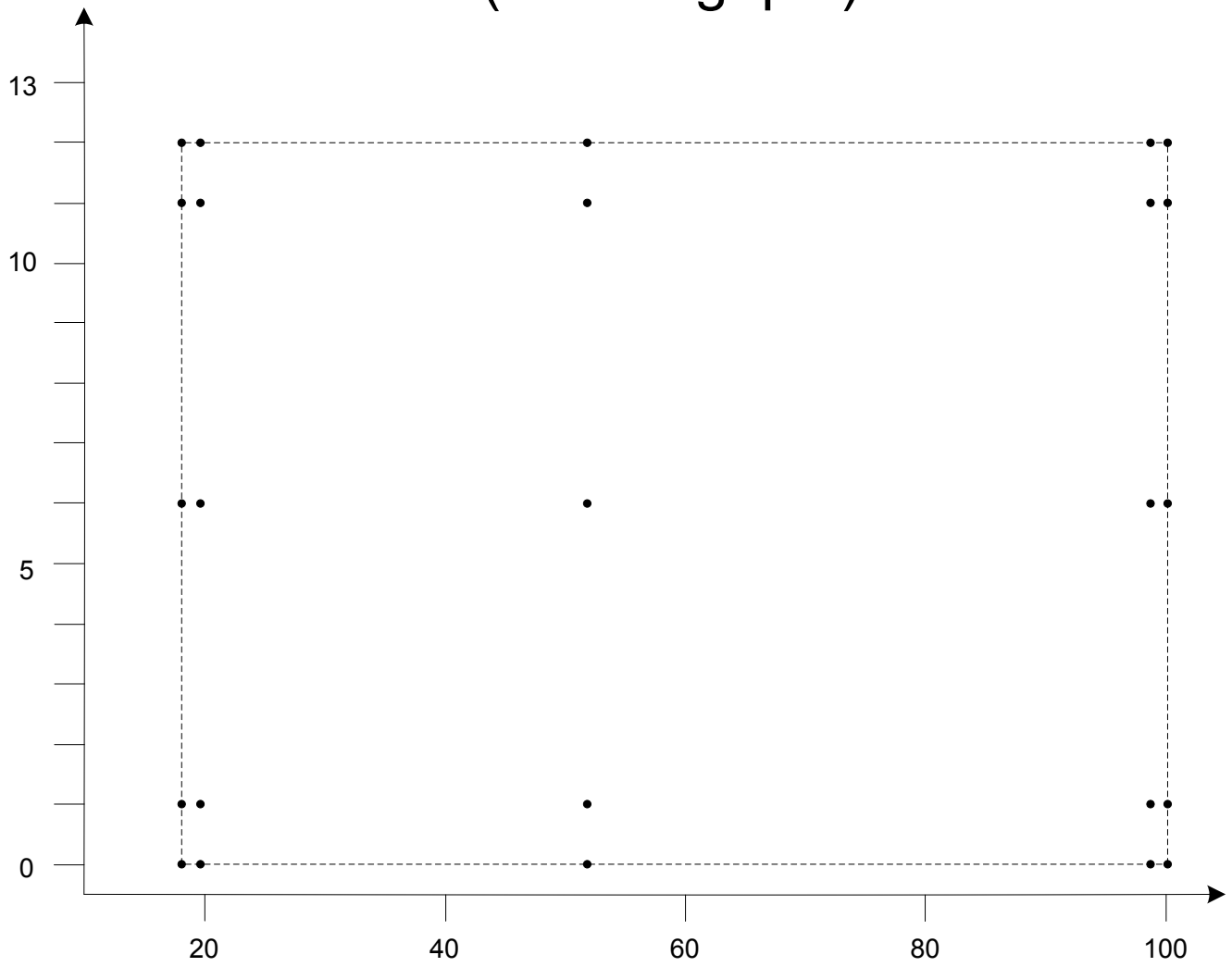
Age Range	*Age Multiplier*	*Points Cutoff*	*Safe Driving Reduction*
16<= age < 25	2.8	1	50
25<= age < 35	1.8	3	50
35<= age < 45	1.0	5	100
45<= age < 60	0.8	7	150
60<= age <= 100	1.5	5	200

Insurance Premium Program Calculations, Test Method Selection

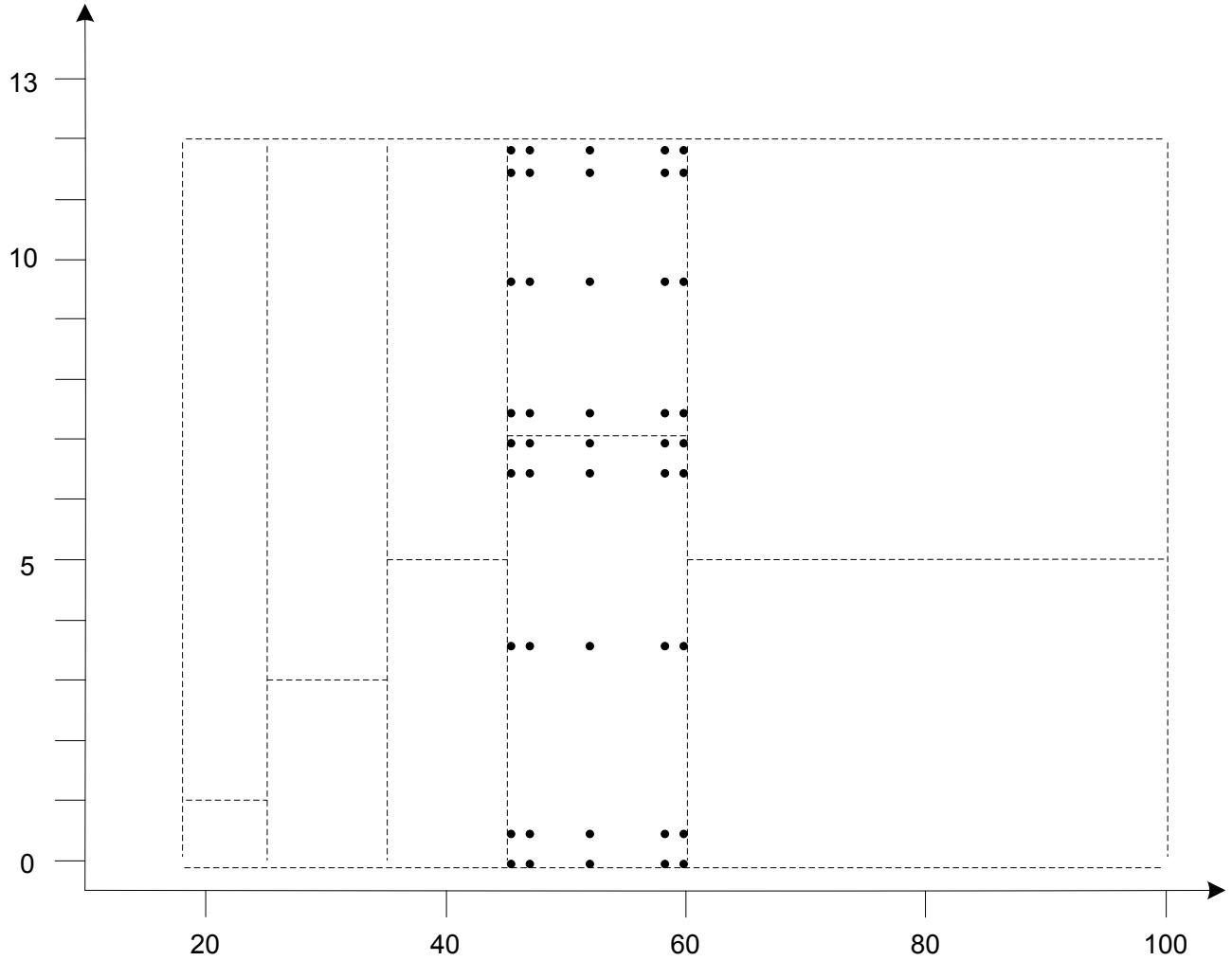
- Premium = BaseRate*ageMultiplier – safeDrivingReduction
- ageMultiplier = F1(age) [from table]
- safeDrivingReduction = F2(age, points) [from table]
- age and safeDrivingReduction are physical variables, with a dependency in F2.
- Boundary values for age: 16, 17, 54, 99, 100
- Boundary values for safeDrivingReduction: 0, 1, 6, 11, 12
- Robust values for age and safeDrivingReduction are not allowed by business rules.
- Worst case BVA yields 25 test cases, and many gaps, some redundancy. Need something better.

Worst Case Boundary Value Test Cases

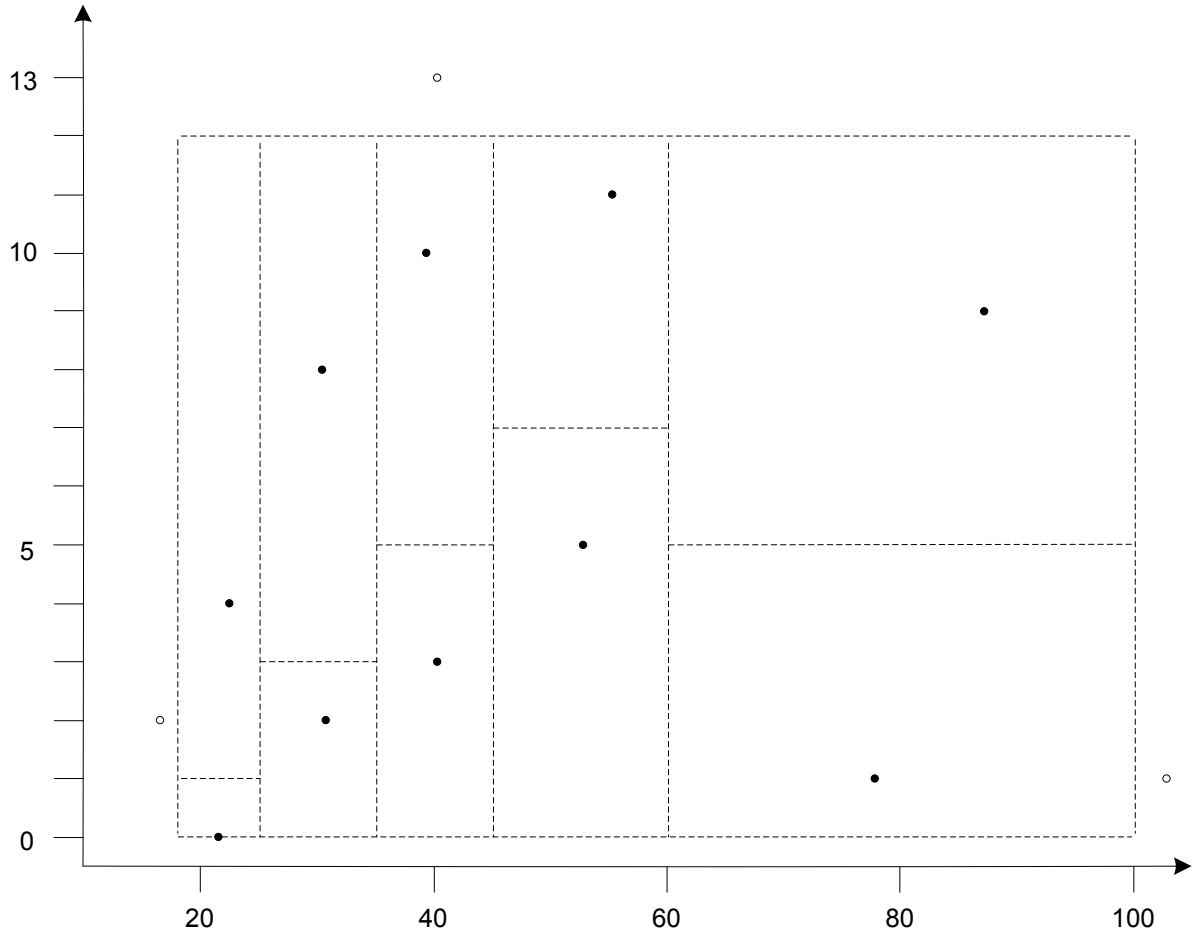
(severe gaps!)



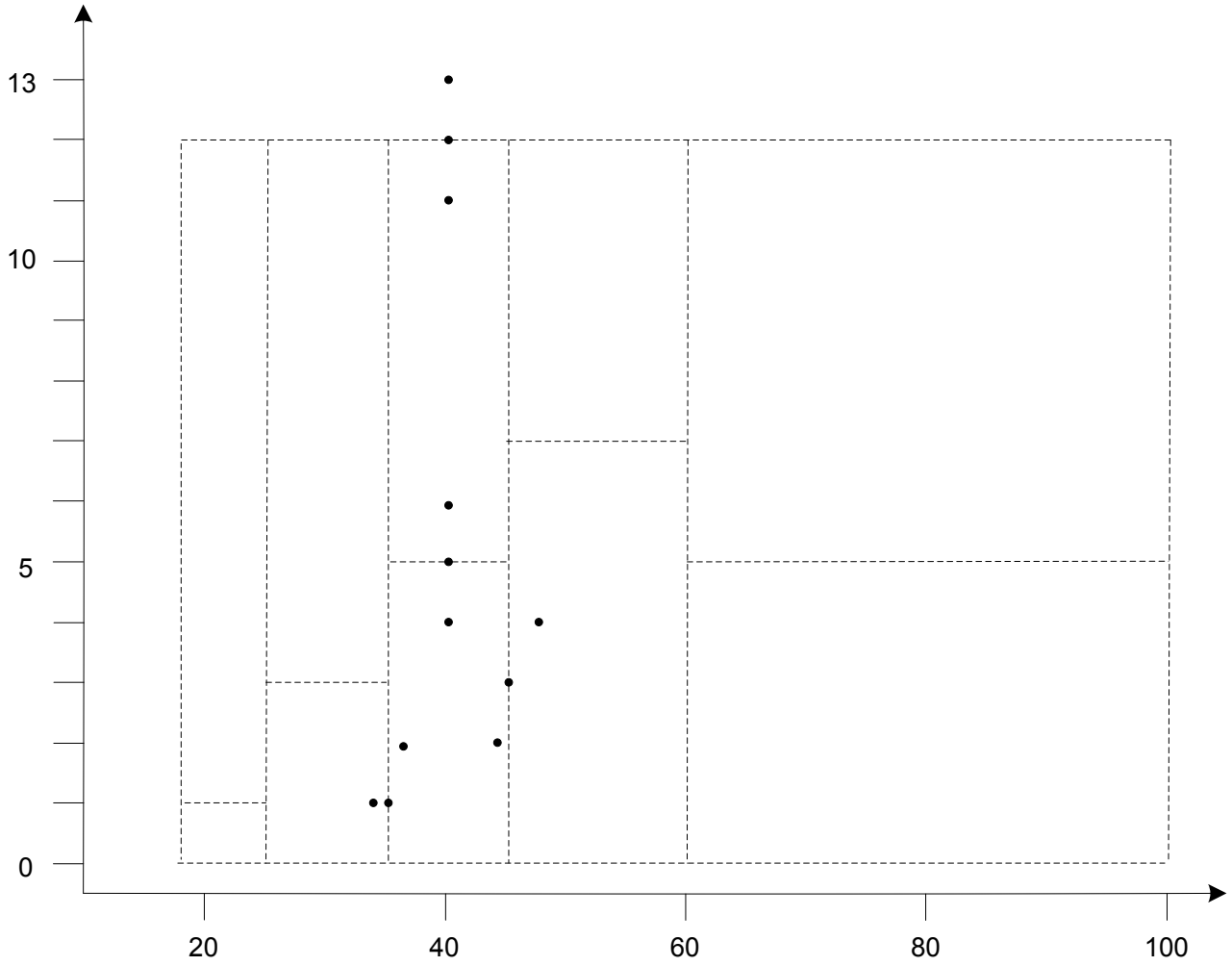
Closer Look At One Age Interval
(severe redundancy)



Weak and Robust Normal Equivalence Classes? (not much help)



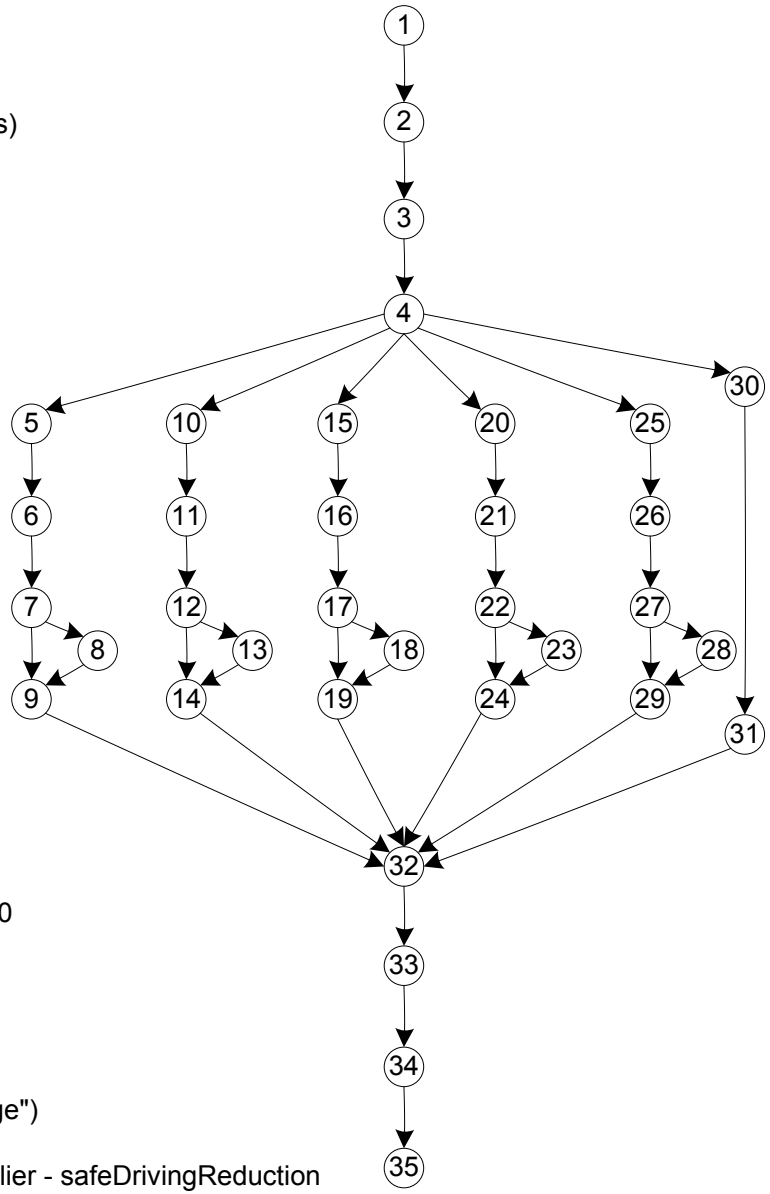
Hybrid Test Cases for One Age Class



Code-Based Testing for the Insurance Premium Program

- (Source pseudo-code and program graph on next slide)
- Cyclomatic complexity: V(G) = 11
- There are 11 feasible paths.

```
Program InsurancePremium
Dim driverAge, points As Integer
Dim baseRate, premium As Real
 1  Input(baseRate, driverAge, points)
 2  premium = 0
 3  safeDrivingReduction = 0
 4  Select Case  driverAge
 5    Case 1:  16<= driverAge  < 25
 6      ageMultiplier = 2.8
 7      If points < 1 Then
 8        safeDrivingReduction = 50
 9     EndIf
10    Case 2:  25<= driverAge  < 35
11      ageMultiplier = 1.8
12      If points < 3 Then
13        safeDrivingReduction = 50
14     EndIf
15    Case 3:  35<= driverAge  < 45
16      ageMultiplier = 1.0
17      If points < 5 Then
18        safeDrivingReduction = 100
19     EndIf
20    Case 4:  45<= driverAge  < 60
21      ageMultiplier = 0.8
22      If points < 7 Then
23        safeDrivingReduction = 150
24     EndIf
25    Case 5:  60<= driverAge  <= 100
26      ageMultiplier = 1.5
27      If points < 5 Then
28        safeDrivingReduction = 200
29     EndIf
30    Case 6:  Else
31       Output("Driver age out of range")
32  End Select
33  premium = baseRate * ageMultiplier - safeDrivingReduction
34  Output(premium)
35  End
```



Paths in the Insurance Premium Program

Path	*Node Sequence*
p1	1-2-3-4-5-6-7-9-32-33-34-35
p2	1-2-3-4-5-6-7-8-9-32-33- 34-35
p3	1-2-3-4-10-11-12-14-32-33- 34-35
p4	1-2-3-4-10-11-12-13-14-32-33-34-35
p5	1-2-3-4-15-16-17-19-32-33-34-35
p6	1-2-3-4-15-16-17-18-19-32-33- 34-35
p7	1-2-3-4-20-21-22-24-32-33- 34-35
p8	1-2-3-4-20-21-22-23-24-32-33-34-35
p9	1-2-3-4-25-26-27-29-32-33-34-35
p10	1-2-3-4-25-26-27-28-29-32-33-34-35
p11	1-2-3-4-30-31-32-33-34-35

Path Coverage of Spec-Based Methods

	Test Cases	*Paths covered*
Boundary value	25	p1, p2, p7, p8, p9, p10
Worst-case boundary value	103	p1, p2, p3, p4, p5, p6, p7, p8, p9, p10
Weak normal equivalence class	10	P1, p2, p3, p4, p5, p6, p7, p8, p9
Robust normal equivalence class	12	p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11
Decision table	12	p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11
Hybrid spec-based (extended to all age classes)	32	p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11