White-Box Software Testing Techniques

By Samer Zein, PhD

STUDENTS-HUB.com

Uploaded By: anonymous

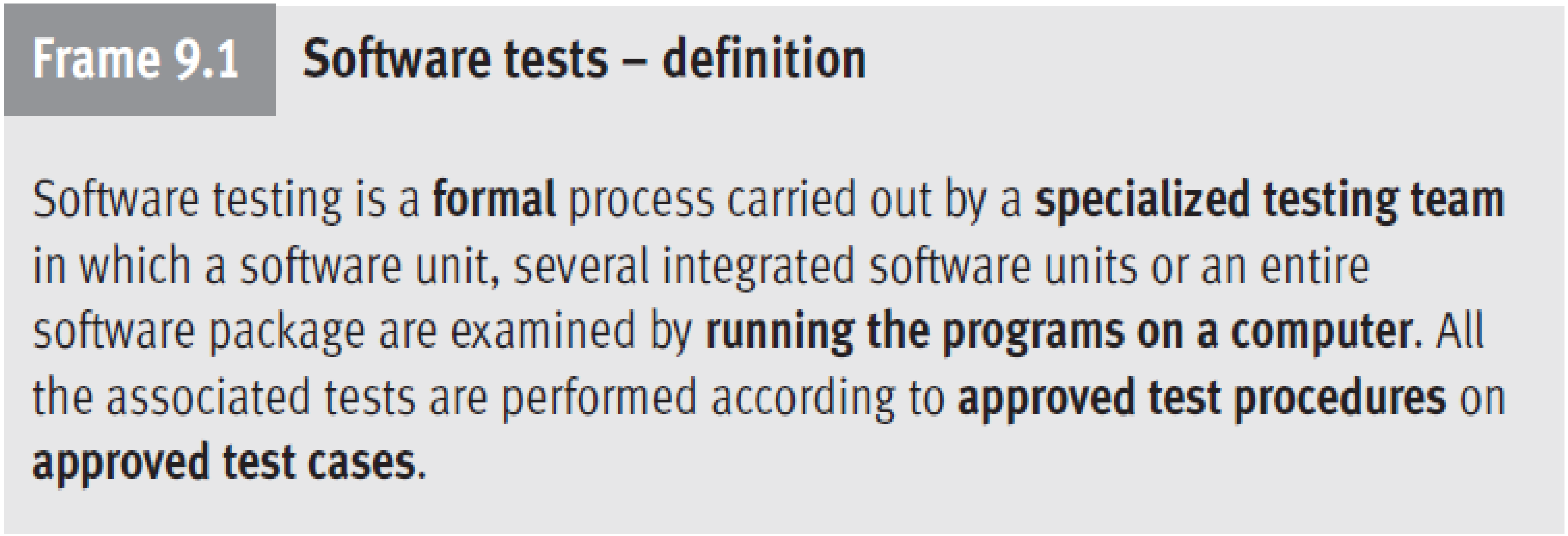
RECAP

- “At early times, ***Software Testing*** was confined to the final stage of development (Big Bang)”
- “But later on, as the importance of early detection of software defects penetrated quality assurance concepts, SQA professionals were encouraged to extend testing to the partial in-process products of coding, which led to software module (unit) testing and integration testing.”
- Hereafter, we will concentrate on test that are performed by running the code.

STUDENTS-HUB.com

Uploaded By: anonymous

Revisiting Definitions



Frame 9.1 **Software tests – definition**

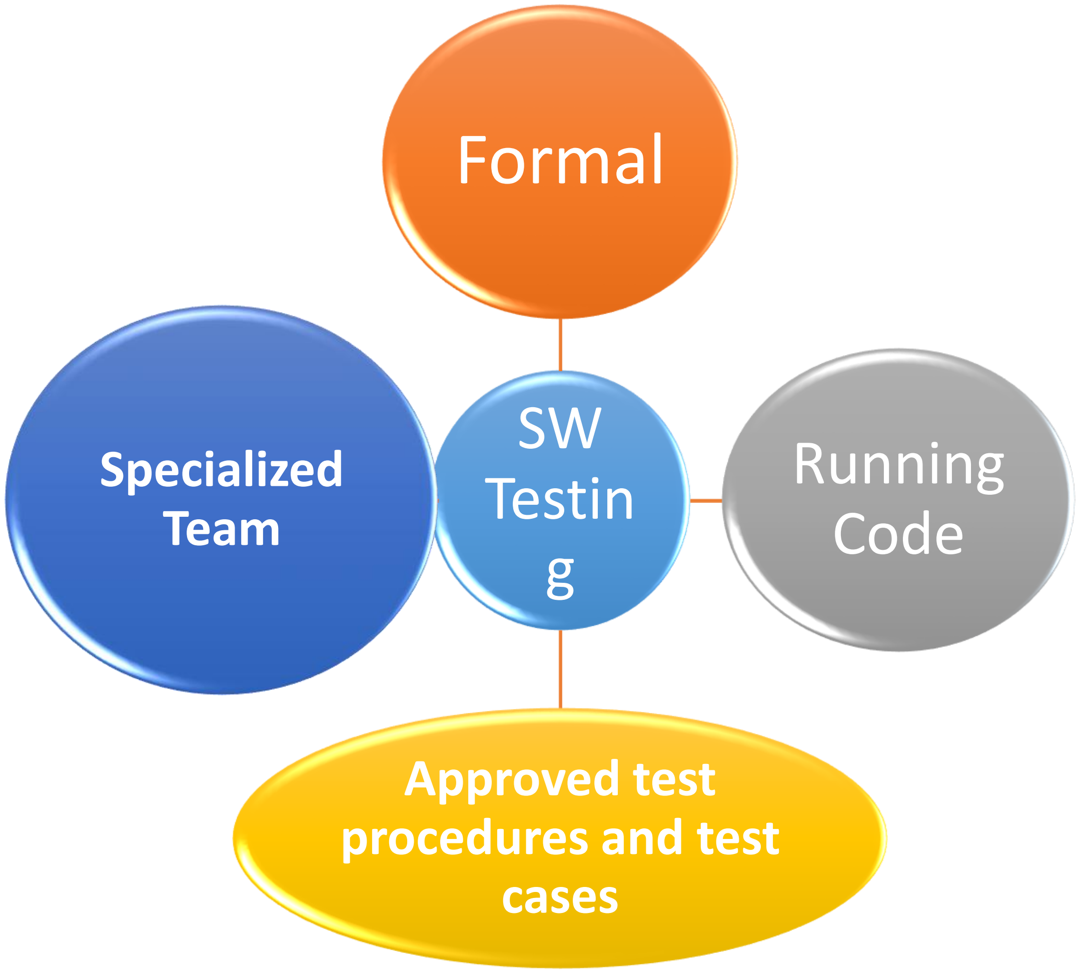
Software testing is a **formal** process carried out by a **specialized testing team** in which a software unit, several integrated software units or an entire software package are examined by **running the programs on a computer**. All the associated tests are performed according to **approved test procedures** on **approved test cases.**

Daniel, 2004

STUDENTS-HUB.com

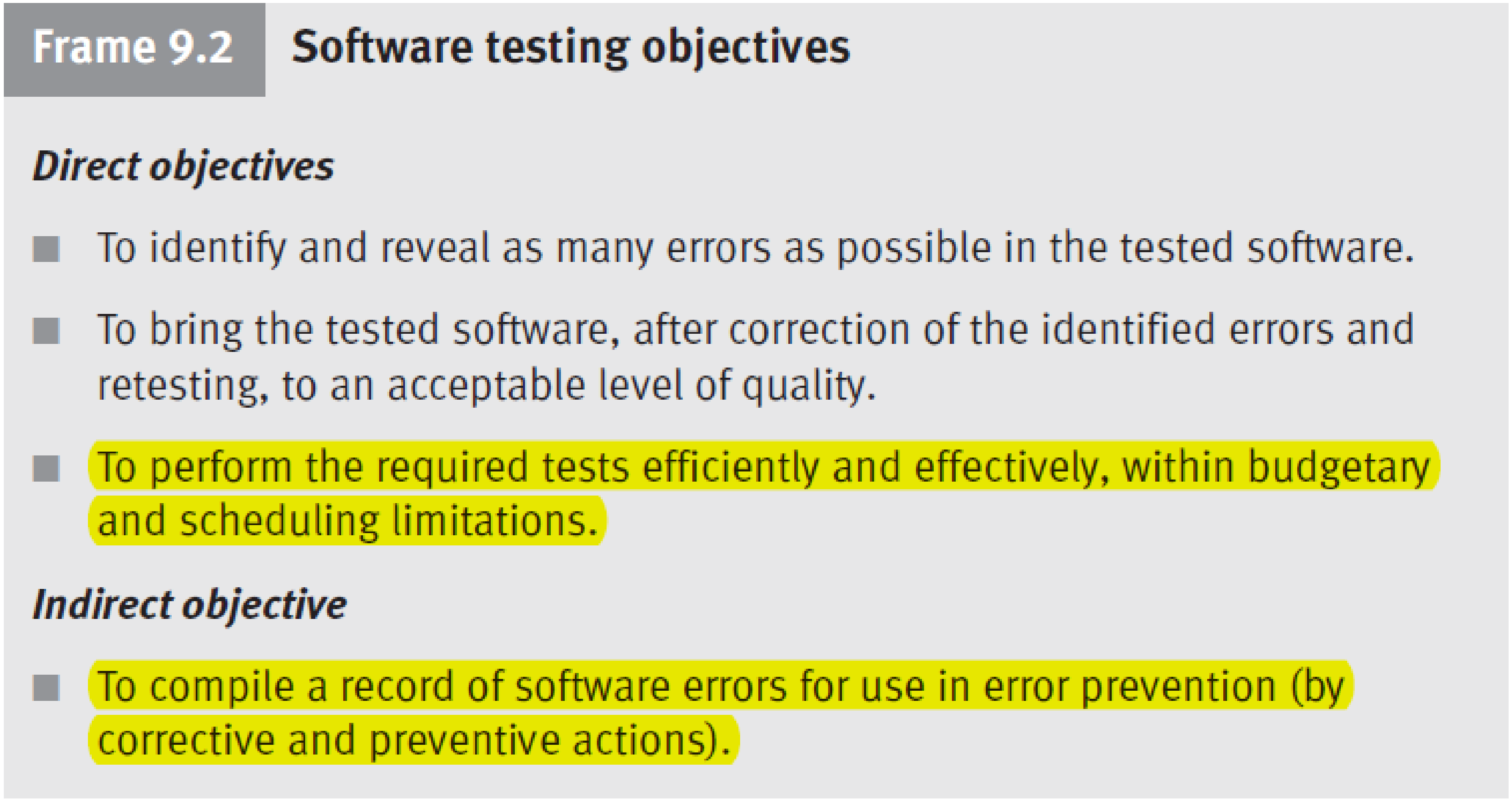
Uploaded By: anonymous

Software Testing is:



STUDENTS-HUB.com

Uploaded By: anonymous



Frame 9.2 **Software testing objectives**

Direct objectives

- To identify and reveal as many errors as possible in the tested software.
- To bring the tested software, after correction of the identified errors and retesting, to an acceptable level of quality.
- To perform the required tests efficiently and effectively, within budgetary and scheduling limitations.

Indirect objective

- To compile a record of software errors for use in error prevention (by corrective and preventive actions).

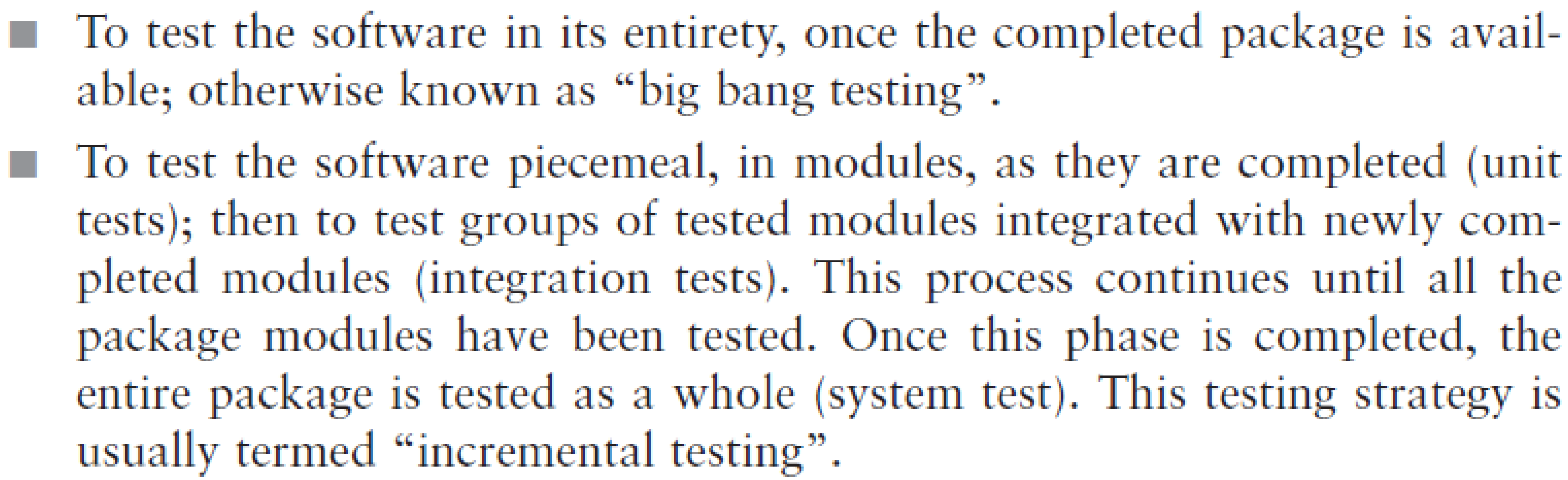
<u>Imp note:</u> *Therefore, we prefer the phrase "***acceptable, level of quality"***, meaning that a certain percentage of bugs, tolerable to the users, will remain unidentified upon installation of the software*

Daniel, 2004

STUDENTS-HUB.com

Uploaded By: anonymous

SW Testing: Two Strategies



- To test the software in its entirety, once the completed package is available; otherwise known as "big bang testing".
- To test the software piecemeal, in modules, as they are completed (unit tests); then to test groups of tested modules integrated with newly completed modules (integration tests). This process continues until all the package modules have been tested. Once this phase is completed, the entire package is tested as a whole (system test). This testing strategy is usually termed "incremental testing".

STUDENTS-HUB.com

Uploaded By: anonymous

White-Box Testing

- Realization of the white box testing concept requires verification of **every program statement** and **comment**.
- In order to perform *data processing and calculation correctness tests,* every computational operation in the sequence of operations created by each test case ("path") **<u>must be examined</u>**

STUDENTS-HUB.com

Uploaded By: anonymous

White-Box Testing: Path Coverage

- Different paths in a software module are created due to **IF–THEN–ELSE** or **DO WHILE** or **DO UNTIL**.
- Path testing: testing all its possible paths.

- Hence, the "**path coverage**" metrics: the percentage of the program paths executed during the test

STUDENTS-HUB.com

Uploaded By: anonymous

Path Coverage is Impractical!

- For example, one simple module can contain about 50 lines of code, but also contain lots of conditional statements, can reach **1024** different paths!!
- Another example of a software package containing 100 modules can reach **tens of thousands** of test cases!!
- But remember that this is accepted at **Critical Applications Testing**

STUDENTS-HUB.com

Uploaded By: anonymous

White-Box Testing:
Line Coverage (Basic Path Testing)

- The line coverage concept requires that, for full line coverage, ***every line of code be executed at least once*** during the process of testing.
- It is more affordable.
- But it is weaker: lots of paths will be left untested!
- All modern testing tools will show you a percentage of lines covered by your test cases.
- They even show you which lines exactly that are not yet executed.

STUDENTS-HUB.com

Uploaded By: anonymous

But First: CFG / Flow Graph

- A CFG captures the **flow of control** within a program.
- Assists testers in the analysis of a program to understand its behavior in terms of the flow of control.
- A CFG can be constructed manually without much difficulty for relatively small programs, say containing less than about **50 statements**.
- However, as the size of the program grows, so does the difficulty of constructing its CFG and hence arises the need for **tools**.

STUDENTS-HUB.com

Uploaded By: anonymous

Control Flow Graphs

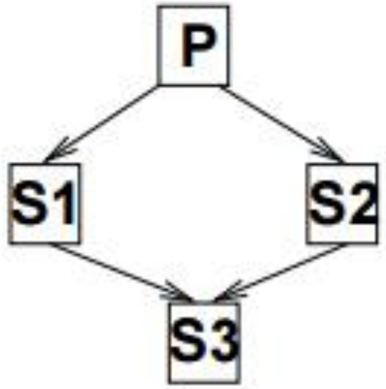
Nodes *Statements or Basic Blocks*

(Maximal sequence of code with branching only allowed at end)

Edges *Possible transfer of control*

Example:

if P
then S1
else S2
S3

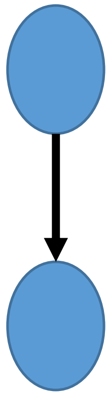

$$\begin{tikzcd} & \boxed{P} \arrow[dl] \arrow[dr] & \\ \boxed{S1} \arrow[dr] & & \boxed{S2} \arrow[dl] \\ & \boxed{S3} & \end{tikzcd}$$

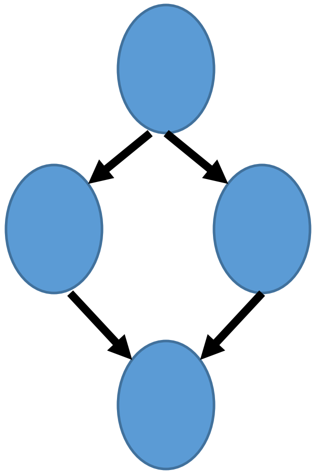
CFG

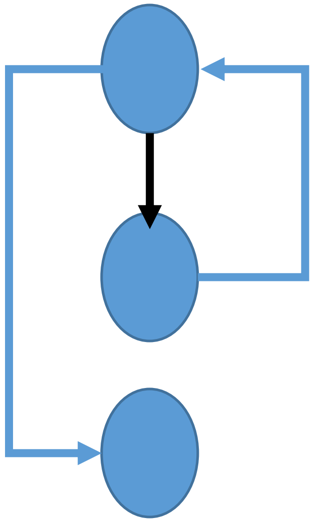
P predecessor of S1 and S2
S1, S2 sucessors of P

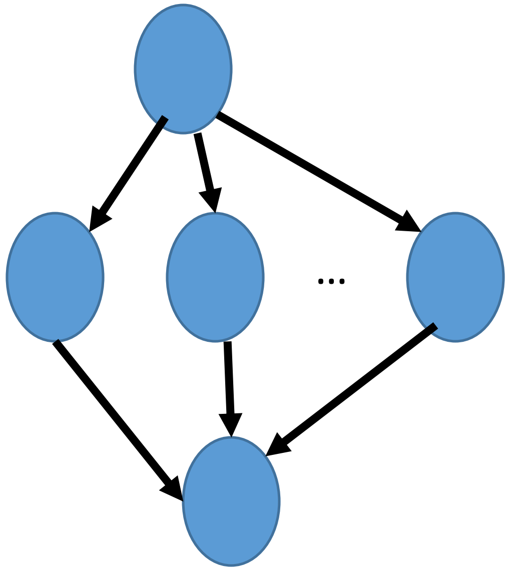
STUDENTS-HUB.com Uploaded By: anonymous

Basic Notation


Sequence


IF


While



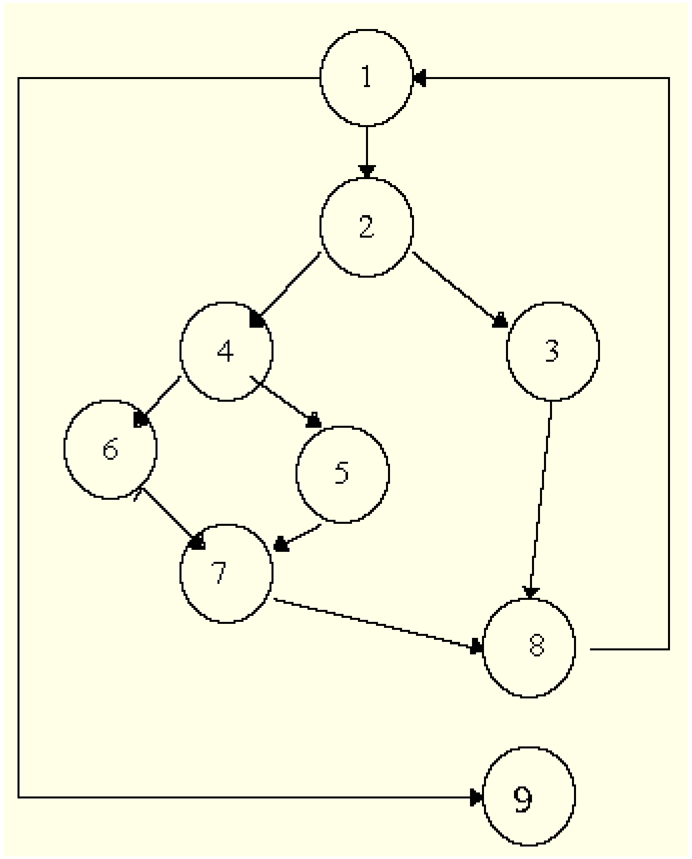
Case

STUDENTS-HUB.com

Uploaded By: anonymous

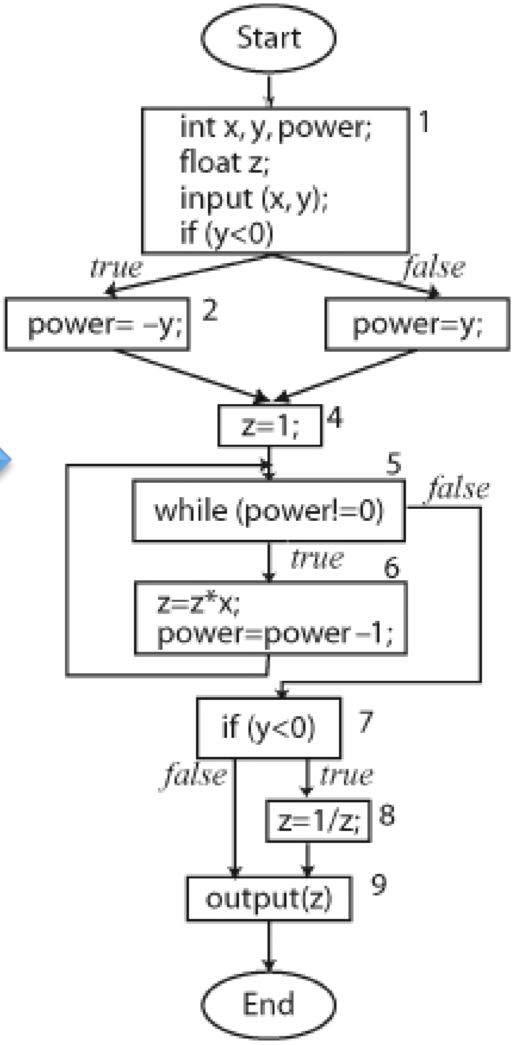
Basic Example

```
1:  WHILE NOT EOF LOOP
2:     Read Record;
2:     IF field1 equals 0 THEN
3:        Add field1 to Total
3:        Increment Counter
4:     ELSE
4:         IF field2 equals 0 THEN
5:             Print Total, Counter
5:             Reset Counter
6:         ELSE
6:             Subtract field2 from Total
7:         END IF
8:     END IF
8:     Print "End Record"
9:  END LOOP
9:  Print Counter
```

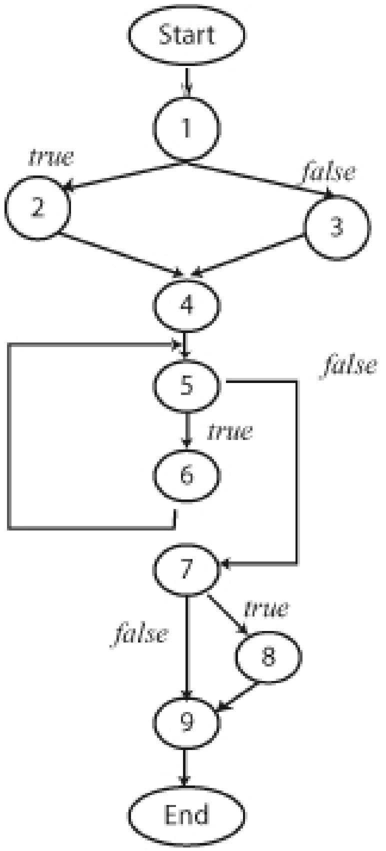


Another Example

```
begin
   int x, y, power;
   float z;
   input (x, y);
   if (y<0)
     power=-y;
   else
     power=y;
   z=1;
   while (power!=0){
     z=z*x;
   power=power-1;
   }
   if (y<0)
     z=1/z;
   output(z);
end
```



(a)



(b)

STUDENTS-HUB.com

Uploaded By: anonymous

A Path:

- A path through a flow graph is considered **Complete** if the first node along the path is **Start** and the terminating node is **End**.
- A path *p* through a flow graph for program *P* is considered **feasible** if there exists at least one test case which when input to *P* causes *p* to be traversed.
- If no such test case exists, then *p* is considered **infeasible**.

STUDENTS-HUB.com

Uploaded By: anonymous

Exercise

P1, P2 are complete, feasible

p_1 = (Start, 1, 2, 4, 5, 6, 5, 7, 8, 9, End)

p_2 = (Start, 1, 3, 4, 5, 6, 5, 7, 9, End)

p_3 = (5, 7, 8, 9)

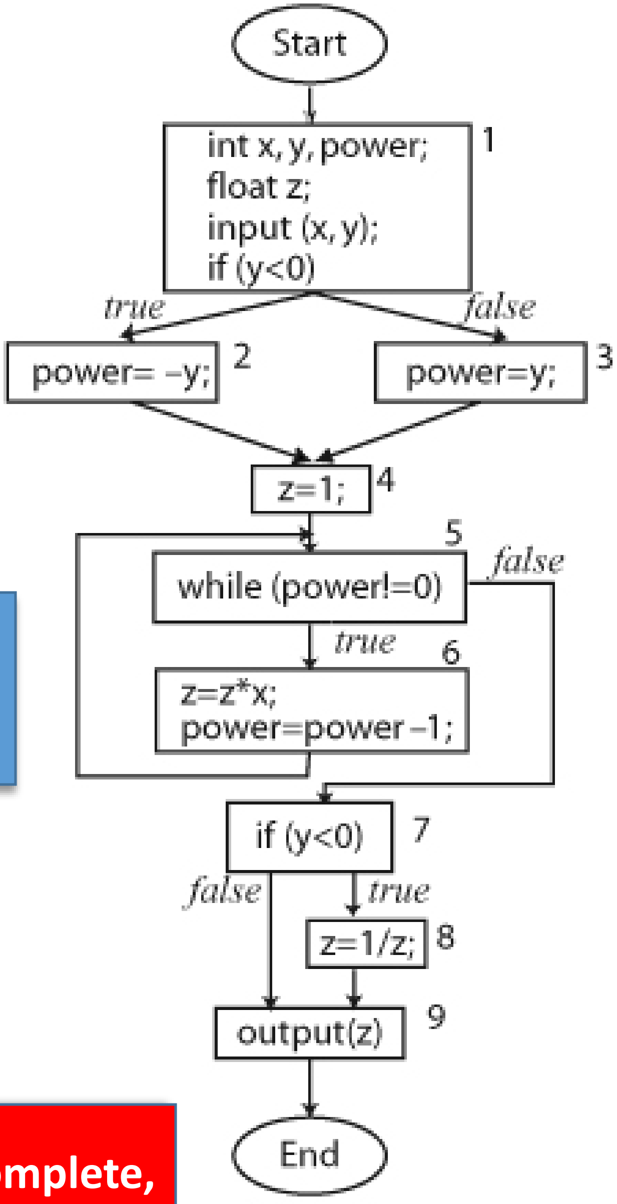
p_4 = (6, 5, 7, 9, End)

P3, P4 are incomplete,

p_5 = (Start, 1, 3, 4, 5, 6, 5, 7, 8, 9, End)

p_6 = (Start, 1, 2, 4, 5, 7, 9, End)

P5, P6 are complete, But infeasible

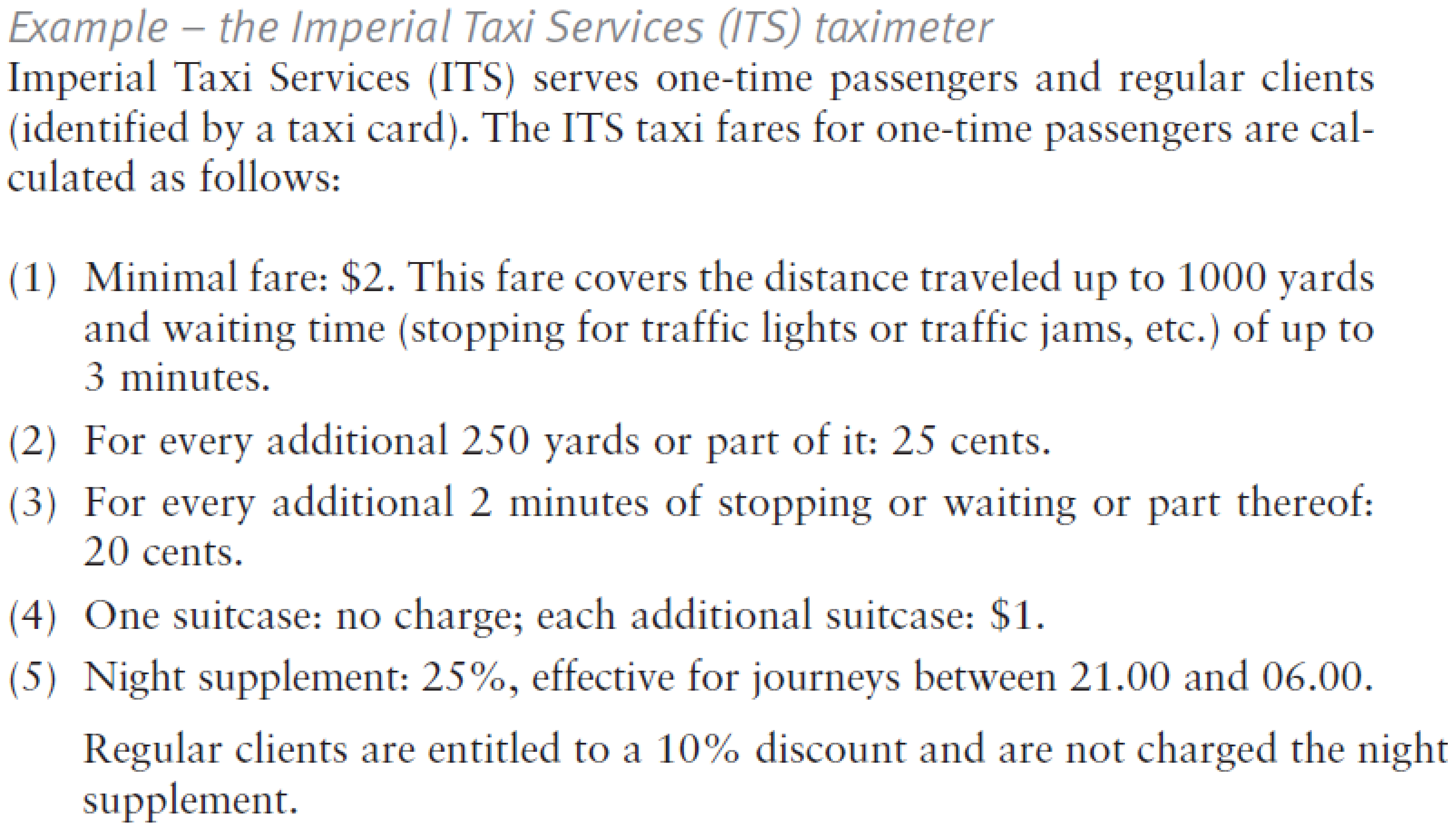


(a)

STUDENTS-HUB.com

Uploaded By: anonymous

In-Class Example Workshop



Example – the Imperial Taxi Services (ITS) taximeter

Imperial Taxi Services (ITS) serves one-time passengers and regular clients (identified by a taxi card). The ITS taxi fares for one-time passengers are calculated as follows:

(1) Minimal fare: $2. This fare covers the distance traveled up to 1000 yards and waiting time (stopping for traffic lights or traffic jams, etc.) of up to 3 minutes.

(2) For every additional 250 yards or part of it: 25 cents.

(3) For every additional 2 minutes of stopping or waiting or part thereof: 20 cents.

(4) One suitcase: no charge; each additional suitcase: $1.

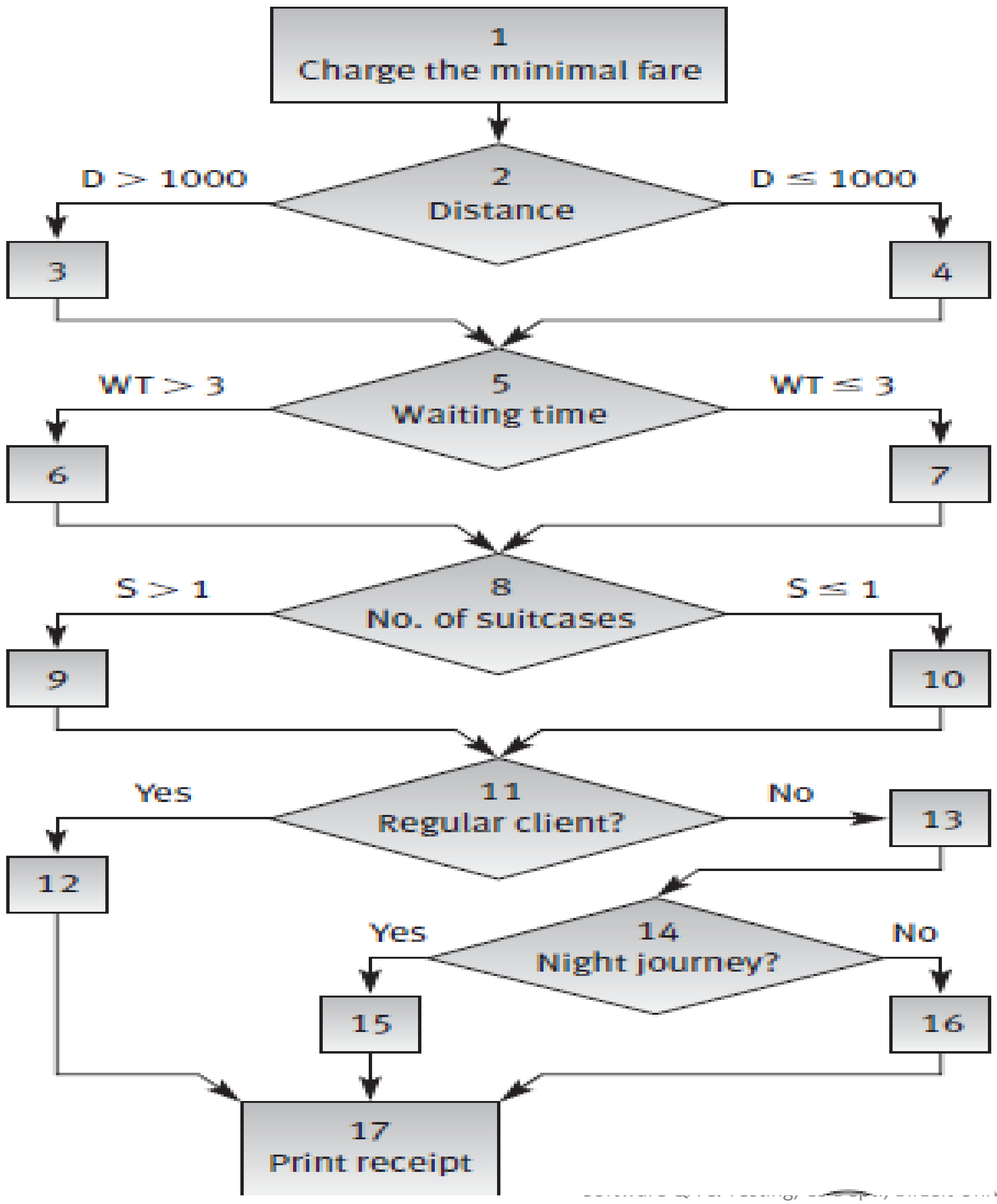
(5) Night supplement: 25%, effective for journeys between 21.00 and 06.00.

Regular clients are entitled to a 10% discount and are not charged the night supplement.

STUDENTS-HUB.com

Uploaded By: anonymous

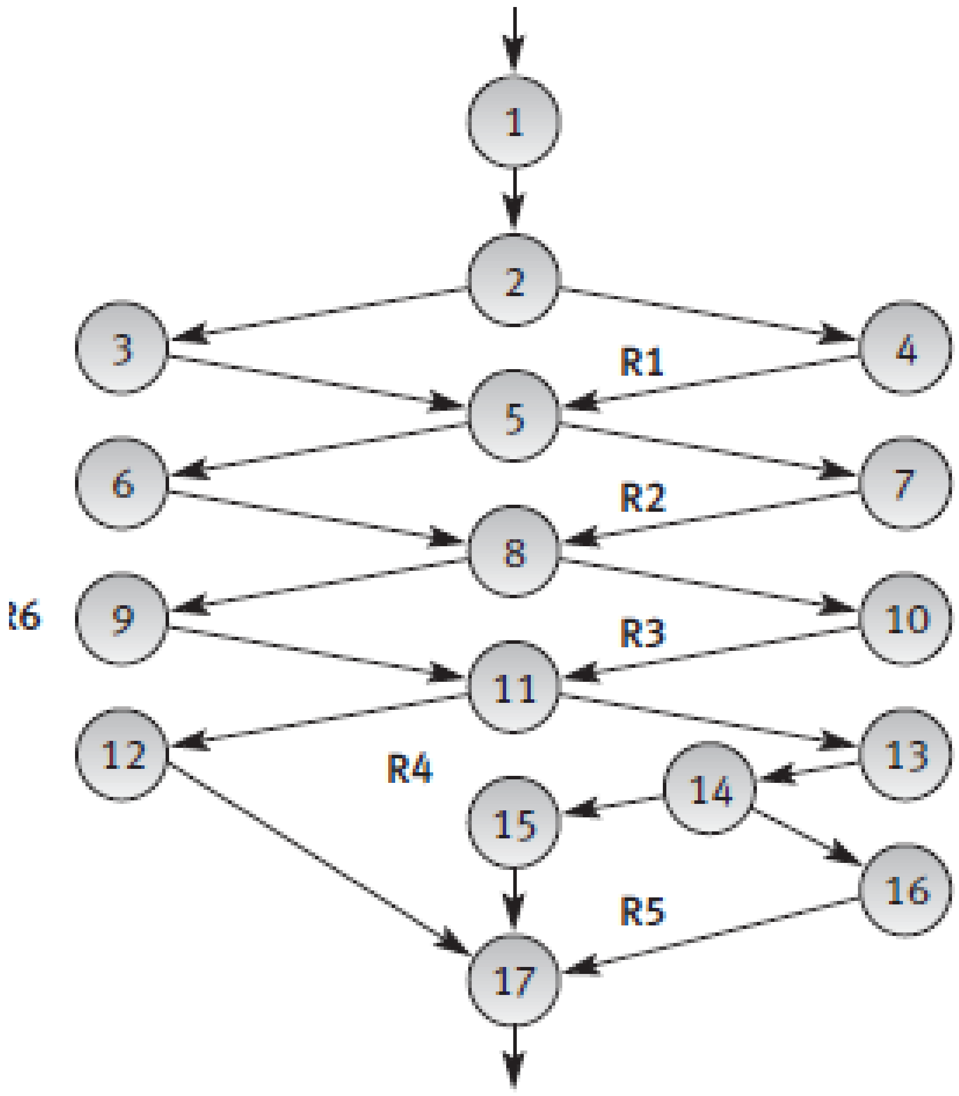
Flow Chart



STUDENTS-HUB.com

Uploaded By: anonymous

Flow Graph:



STUDENTS-HUB.com

Uploaded By: anonymous

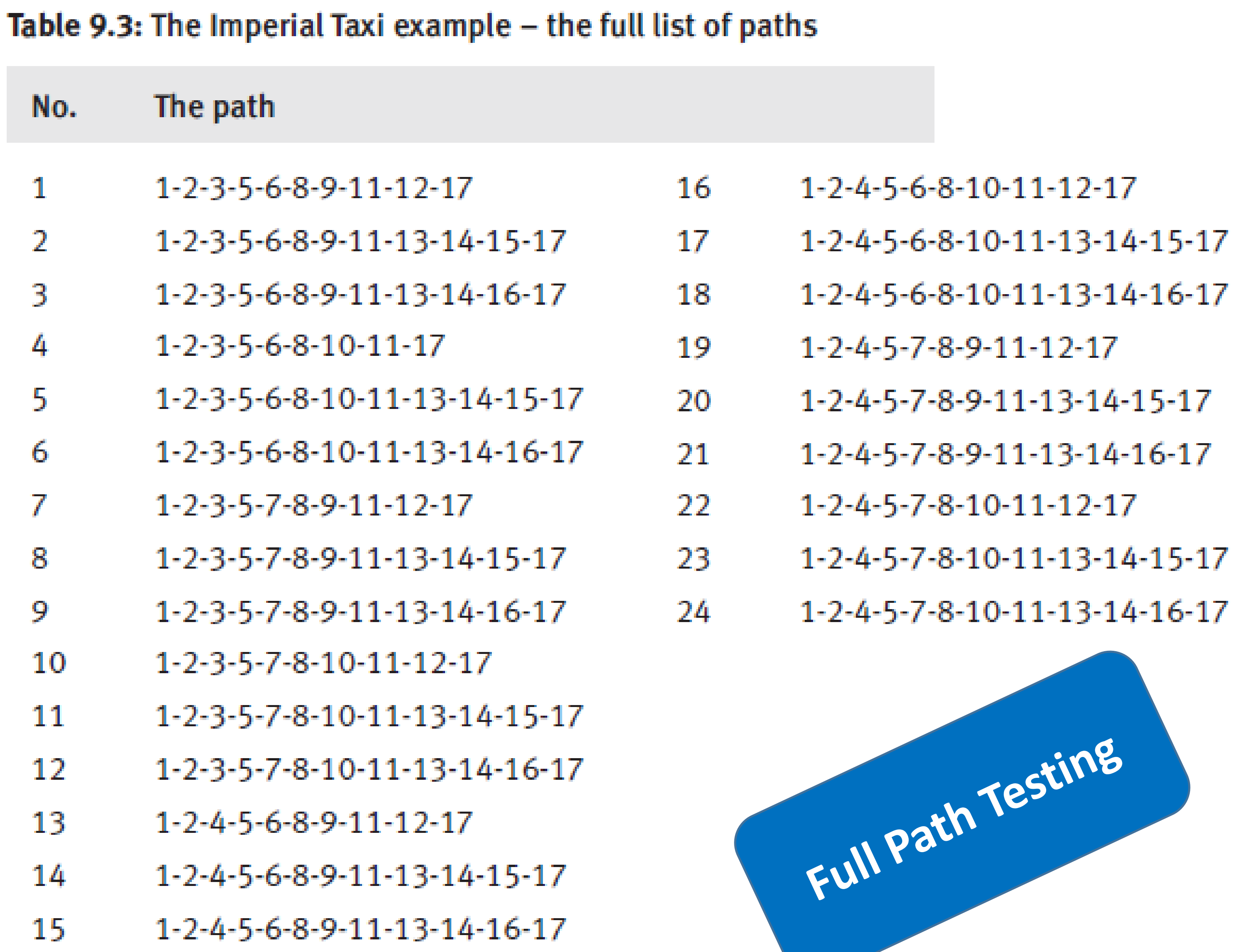


Table 9.3: The Imperial Taxi example – the full list of paths

No.	The path
1	1-2-3-5-6-8-9-11-12-17
2	1-2-3-5-6-8-9-11-13-14-15-17
3	1-2-3-5-6-8-9-11-13-14-16-17
4	1-2-3-5-6-8-10-11-17
5	1-2-3-5-6-8-10-11-13-14-15-17
6	1-2-3-5-6-8-10-11-13-14-16-17
7	1-2-3-5-7-8-9-11-12-17
8	1-2-3-5-7-8-9-11-13-14-15-17
9	1-2-3-5-7-8-9-11-13-14-16-17
10	1-2-3-5-7-8-10-11-12-17
11	1-2-3-5-7-8-10-11-13-14-15-17
12	1-2-3-5-7-8-10-11-13-14-16-17
13	1-2-4-5-6-8-9-11-12-17
14	1-2-4-5-6-8-9-11-13-14-15-17
15	1-2-4-5-6-8-9-11-13-14-16-17
16	1-2-4-5-6-8-10-11-12-17
17	1-2-4-5-6-8-10-11-13-14-15-17
18	1-2-4-5-6-8-10-11-13-14-16-17
19	1-2-4-5-7-8-9-11-12-17
20	1-2-4-5-7-8-9-11-13-14-15-17
21	1-2-4-5-7-8-9-11-13-14-16-17
22	1-2-4-5-7-8-10-11-12-17
23	1-2-4-5-7-8-10-11-13-14-15-17
24	1-2-4-5-7-8-10-11-13-14-16-17

Full Path Testing

STUDENTS-HUB.com

Uploaded By: anonymous

Table 9.4: The Imperial Taxi example – the minimum number of paths

No.	The path
1	1-2-3-5-6-8-9-11-12-17
23	1-2-4-5-7-8-10-11-13-14-15-17
24	1-2-4-5-7-8-10-11-13-14-16-17

Line Coverage Testing

STUDENTS-HUB.com

Uploaded By: anonymous

Line Coverage vs Full Path

- The proportion of test cases required to test the system by full line coverage of three test cases (by basic path testing) versus full path coverage **of 24 test cases is 1:8!**

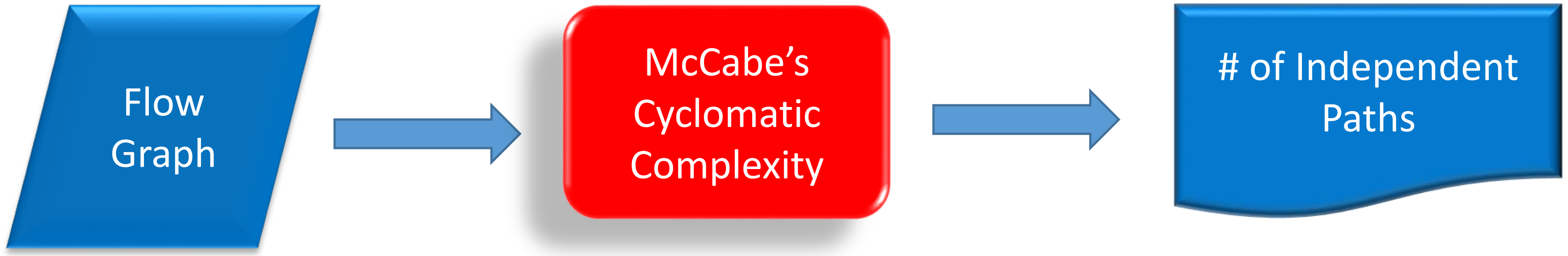
- This ratio grows rapidly with program complexity.

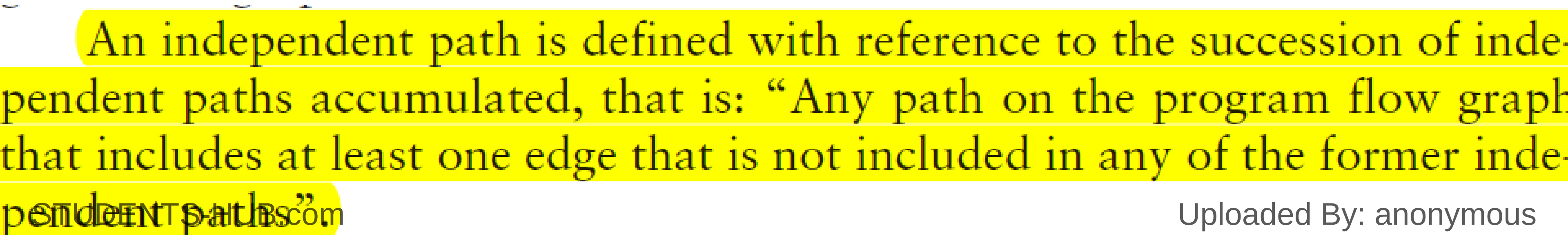
STUDENTS-HUB.com

Uploaded By: anonymous

McCabe’s Cyclomatic complexity metrics

- Support for **Basic Path Testing** is provided by McCabe’s Cyclomatic Complexity Metrics
- Measures the complexity of a program.
- Measures the maximum number of paths needed to achieve full line coverage



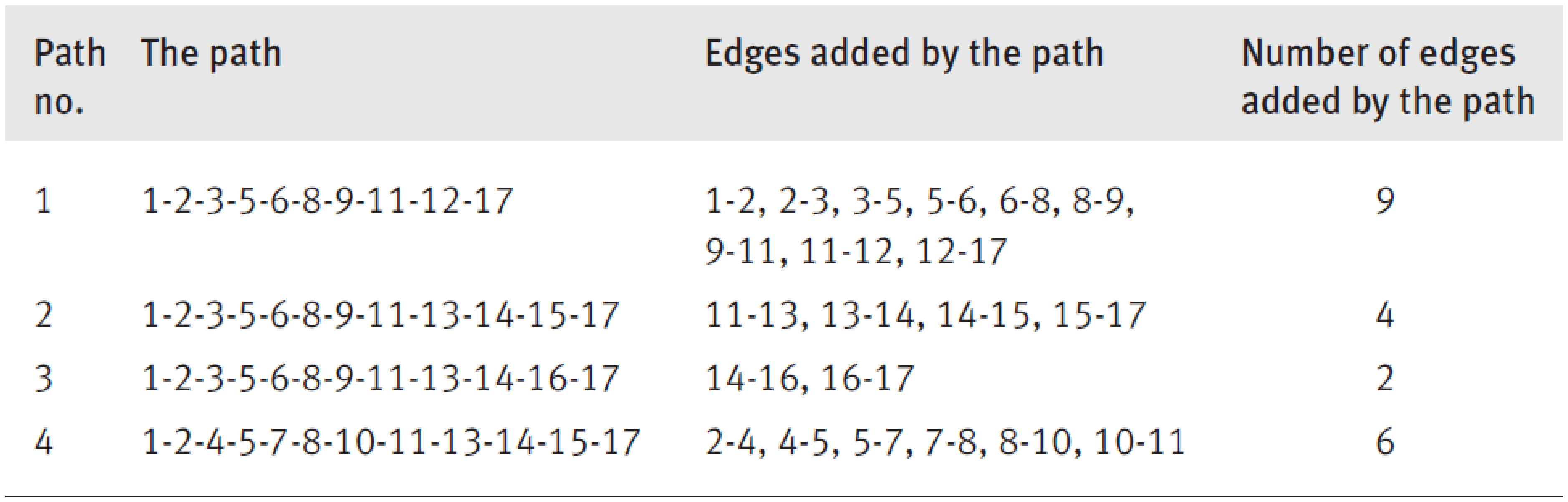

An independent path is defined with reference to the succession of inde-
pendent paths accumulated, that is: “Any path on the program flow graph
that includes at least one edge that is not included in any of the former inde-
pendent paths”.

STUDENTS-HUB.com

Uploaded By: anonymous

McCabe’s Cyclomatic Complexity Example:

Table 9.5: The ITS example – the set of independent paths to achieve full coverage



Path no.	The path	Edges added by the path	Number of edges added by the path
1	1-2-3-5-6-8-9-11-12-17	1-2, 2-3, 3-5, 5-6, 6-8, 8-9, 9-11, 11-12, 12-17	9
2	1-2-3-5-6-8-9-11-13-14-15-17	11-13, 13-14, 14-15, 15-17	4
3	1-2-3-5-6-8-9-11-13-14-16-17	14-16, 16-17	2
4	1-2-4-5-7-8-10-11-13-14-15-17	2-4, 4-5, 5-7, 7-8, 8-10, 10-11	6

STUDENTS-HUB.com

Uploaded By: anonymous

McCabe’s Cyclomatic Complexity Calculation

- V(G) = R, where R→ #of Regions
- V(G) = E – N + 2 where E → # of Edges, N → # of Nodes
- V(G) = P + 1 where P → number of decisions contained in the graph

STUDENTS-HUB.com

Uploaded By: anonymous

McCabe’s Cyclomatic Complexity Calculation Example

Example

Applying the above to the ITS taximeter module example described above, we can obtain the values of the above parameters from Figure 9.3. We find that $R = 6$, $E = 21$, $N = 17$, and $P = 5$. Substituting these values into the metrics formulae we obtain:

(1) $V(G) = R = 6$

(2) $V(G) = E - N + 2 = 21 - 17 + 2 = 6$

(3) $V(G) = P + 1 = 5 + 1 = 6$

STUDENTS-HUB.com

Uploaded By: anonymous

Table 9.6: The ITS example – the maximum set of independent paths

Path no.	The path	Edges added by the path	Number of edges added by the path
1	1-2-3-5-6-8-9-11-12-17	1-2, 2-3, 3-5, 5-6, 5-8, 8-9, 9-11, 11-12, 12-17	9
2	1-2-4-5-6-8-9-11-12-17	2-4, 4-5	2
3	1-2-3-5-7-8-9-11-12-17	5-7, 7-8	2
4	1-2-3-5-6-8-10-11-12-17	8-10, 10-11	2
5	1-2-3-5-6-8-9-11-13-14-15-17	11-13, 13-14, 14-15, 15-17	4
6	1-2-3-5-6-8-9-11-13-14-16-17	14-16, 16-17	2

STUDENTS-HUB.com

Uploaded By: anonymous

McCabe’s Cyclomatic Complexity and Testability

- <= 5 → Testable
- <= 10 → Not too difficult
- >20 → High complexity
- > 50 → Untestable

STUDENTS-HUB.com

Uploaded By: anonymous

White-Box Testing: Branch/ Decision Testing

- *Branch coverage is a measure of the percentage of the decision points (Boolean expressions) of the program have been evaluated as both true and false in test cases.*

- In this technique, we evaluate an **Entire** Boolean Expression as **one** true and false expression.
- **Even if it contains multiple logical operators**

STUDENTS-HUB.com

Uploaded By: anonymous

Example:

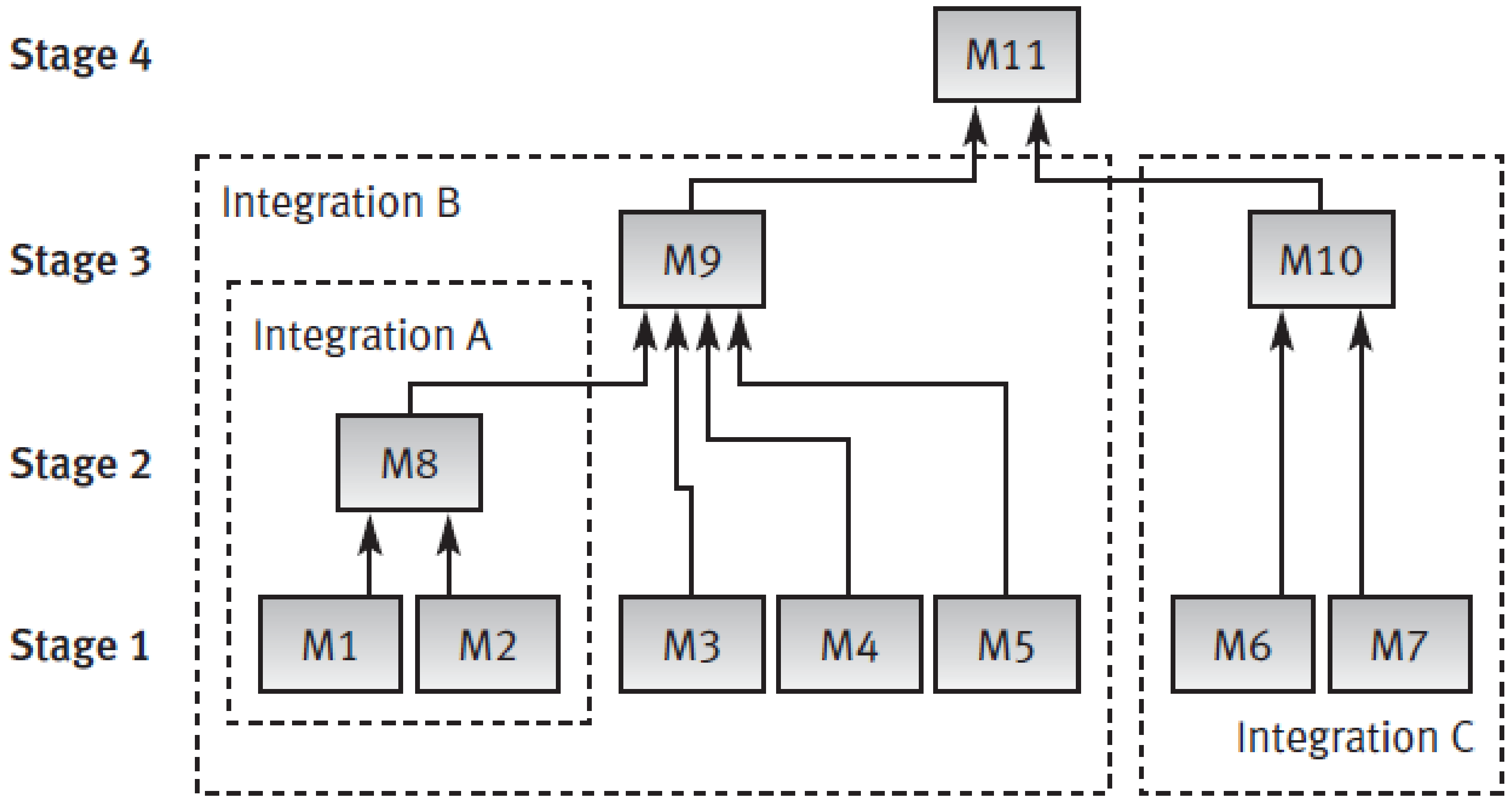
```
1     int foo (int a, int b, int c, int d, float e)  {
2           float e;
3           if (a == 0)  {
4                return 0;
5           }
6           int x = 0;
7           if ((a==b) OR ((c == d) AND bug(a) )) {
8                x=1;
9           }
10           e = 1/x;
11     return e;
12     }
```

Line #	Predicate	True	False
3	(a == 0)	Test Case 1 **foo(0, 0, 0, 0, 0)** **return 0**	Test Case 2 **foo(1, 1, 1, 1, 1)** **return 1**
7	((a==b) OR ((c == d) AND bug(a)))	Test Case 2 **foo(1, 1, 1, 1, 1)** **return 1**	

STUDENTS-HUB.com

Uploaded By: anonymous

Integration Testing: Bottom-Up

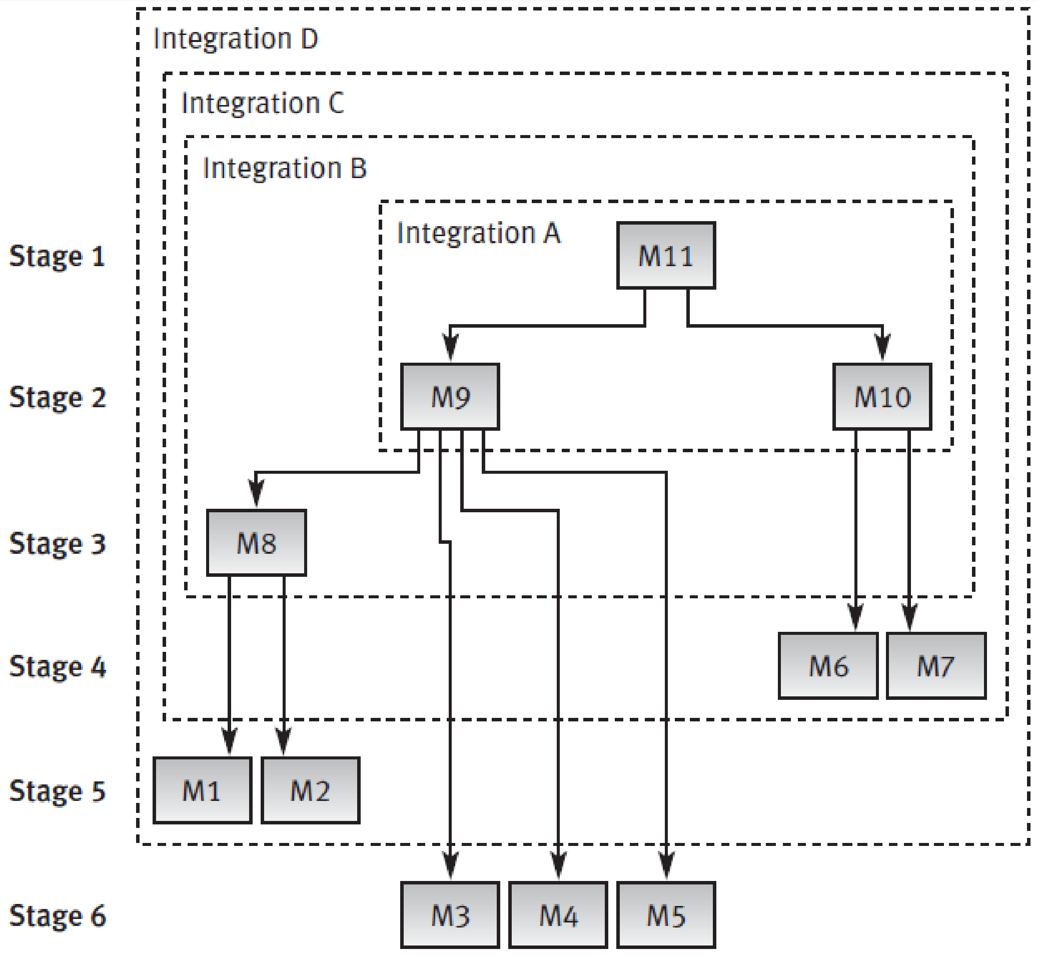


(a) Bottom-up testing

STUDENTS-HUB.com

Uploaded By: anonymous

Integration Testing: Top-Down



(b) Top-down testing

STUDENTS-HUB.com

Uploaded By: anonymous

White-Box Testing: Condition Coverage

- *Condition coverage is a measure of percentage of Boolean sub-expressions of the program that have been evaluated as both true or false outcome [applies to compound predicate] in test cases.*
- *Notice that in* line 7 there are three sub-Boolean expressions to the larger statement (a==b), (c==d), and bug(a).
- Condition coverage measures the outcome of each of these sub-expressions
- independently of each other.

STUDENTS-HUB.com

Uploaded By: anonymous

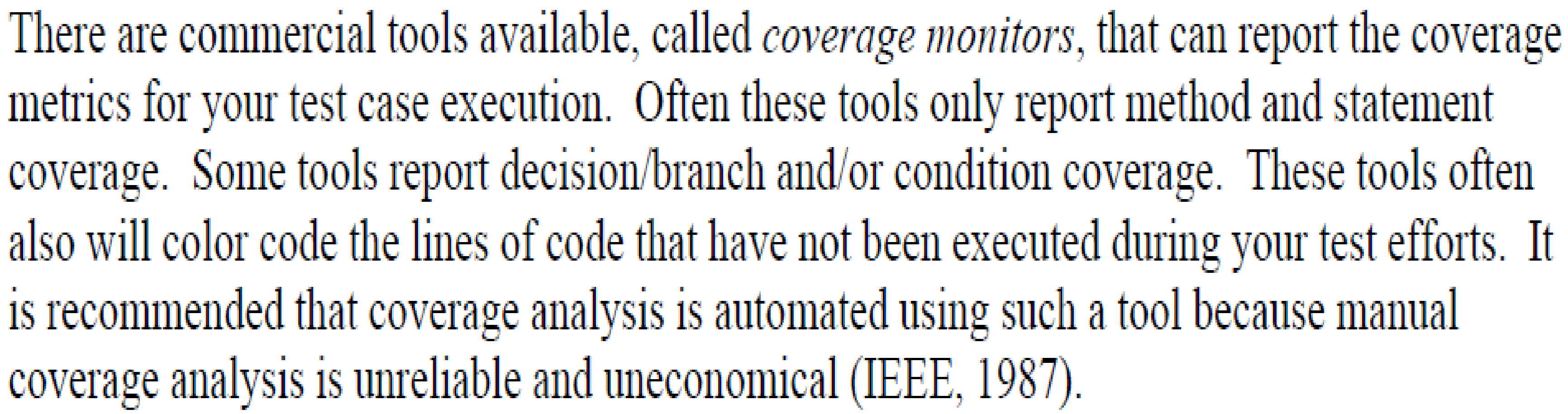
White-Box Testing Verdict

- The characteristics of white box testing limit its use to software modules of :
  - very high risk and very high cost of failure,
  - where it is highly important to identify and fully correct as many of the software errors as possible.

STUDENTS-HUB.com

Uploaded By: anonymous

White-Box Testing: Verdict



There are commercial tools available, called *coverage monitors*, that can report the coverage metrics for your test case execution. Often these tools only report method and statement coverage. Some tools report decision/branch and/or condition coverage. These tools often also will color code the lines of code that have not been executed during your test efforts. It is recommended that coverage analysis is automated using such a tool because manual coverage analysis is unreliable and uneconomical (IEEE, 1987).

STUDENTS-HUB.com

Uploaded By: anonymous