Software Engineering

Dr. Fatma ElSayed

Dr. Ahmed Yousry

Computer Science Department

Course Information

Contents

- Introduction to Software Engineering
- Software Processes
- Requirements Engineering
- System Modelling Part 1
- System Modelling Part 2
- System Modelling Part 3
- Software Testing Part 1
- Software Testing Part 2
- System Architecture

Course Information

Contents

- Introduction to Software Engineering
- Software Processes
- Requirements Engineering
- System Modelling Part 1
- System Modelling Part 2
- System Modelling Part 3
- Software Testing Part 1
- Software Testing Part 2
- System Architecture

Chapter 8: Software Testing Part 2

White Box Testing Techniques

Basis Path Testing

(1)

Construct the Control Flow Graph(CFG)

(2)

Compute the Cyclomatic Complexity Metric(CCM) of the Graph

(3)

Identify the Independent Paths

Independent Program Paths

- An **independent path** is any path through the program that introduces at **least one new set of processing statement or new condition**.
- When stated in terms of a flow graph, an independent path must move along at least one edge that **has not been traversed** before the path is defined.
- If a path has **one new node compared** to all other linearly independent paths, then the path is also linearly independent.
- **Every statement** in the program should be executed **at least once** and every condition will have been executed **on its true and false**.

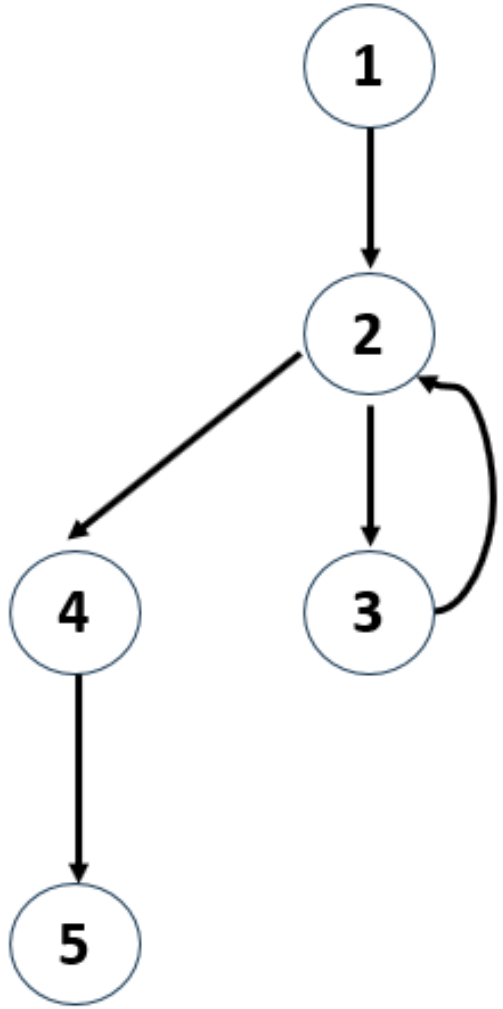
Cyclomatic Complexity

Cyclomatic Complexity **2**

Basis set of independent paths

- Path 1: 1-2-4-5
- Path 2: 1-2-3-2-4-5

- Each new path introduces a new edge.

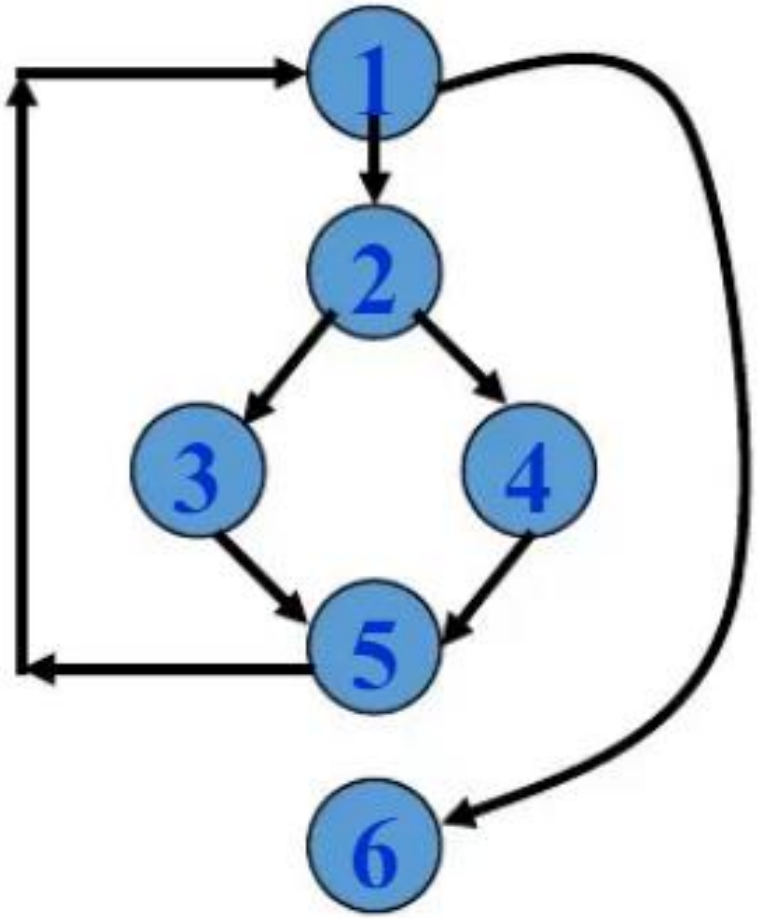


Cyclomatic Complexity

Cyclomatic Complexity **3**

Basis set of independent paths

- Path 1: 1-6
- Path 2: 1-2-3-5-1-6
- Path 3: 1-2-4-5-1-6

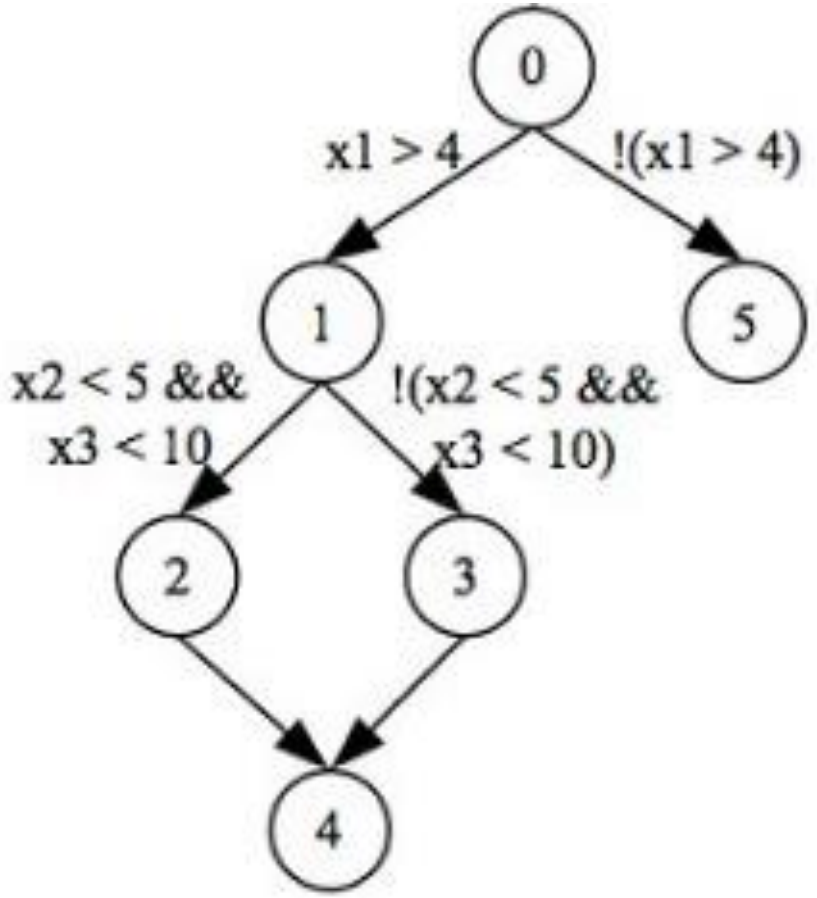


Cyclomatic Complexity

Cyclomatic Complexity **3**

Basis set of independent paths

- Path 1: 0-5
- Path 2: 0-1-2-4
- Path 3: 0-1-3-4



Independent Program Paths

Cyclomatic Complexity **4**

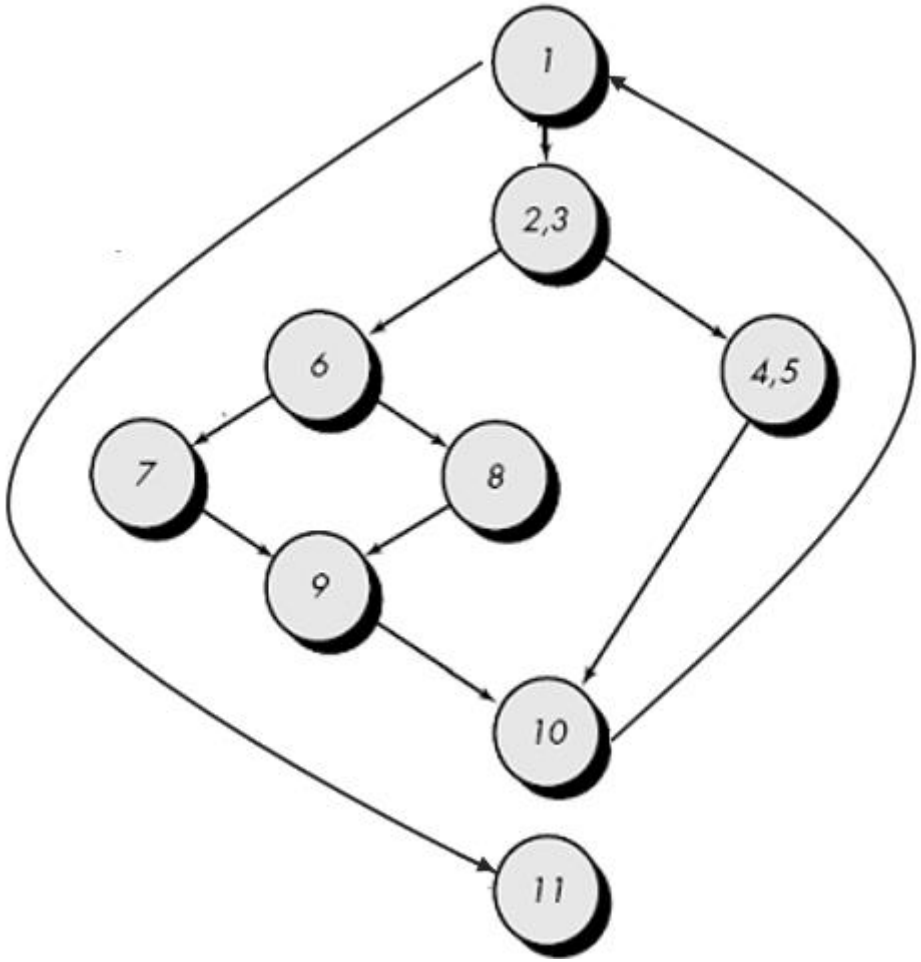
Basis set of independent paths

- Path 1: 1-11
- Path 2: 1-2-3-4-5-10-1-11
- Path 3: 1-2-3-6-8-9-10-1-11
- Path 4: 1-2-3-6-7-9-10-1-11

What about this path?

1-2-3-4-5-10-1-2-3-6-8-9-10-1-11

Path is not independent



White Box Testing Techniques

Basis Path Testing

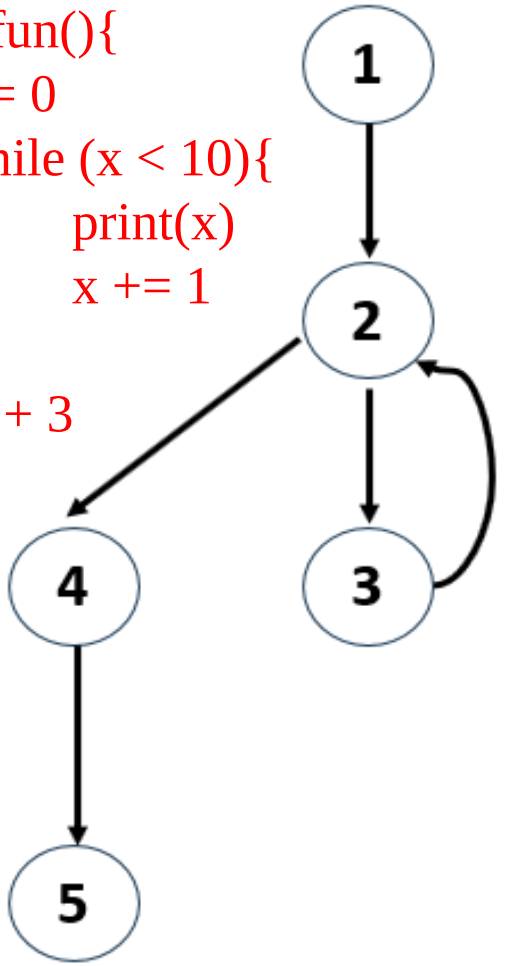
(1)	(2)	(3)	(4)
Construct the Control Flow Graph(CFG)	Compute the Cyclomatic Complexity Metric(CCM) of the Graph	Identify the Independent Paths	Design Test cases from Independent Paths

Deriving Test Cases

- **Path 1**: 1-2-4-5
- **Path 2**: 1-2-3-2-4-5

Case #	Path	Input	Expected outcomes
1	1-2-3-2-4-5	x = 0	Print(0), Y=13
2	1-2-4-5	x=10	Y=13

```
void fun(){
    x = 0
    while (x < 10){
        print(x)
        x += 1
    }
y = x + 3
}
```



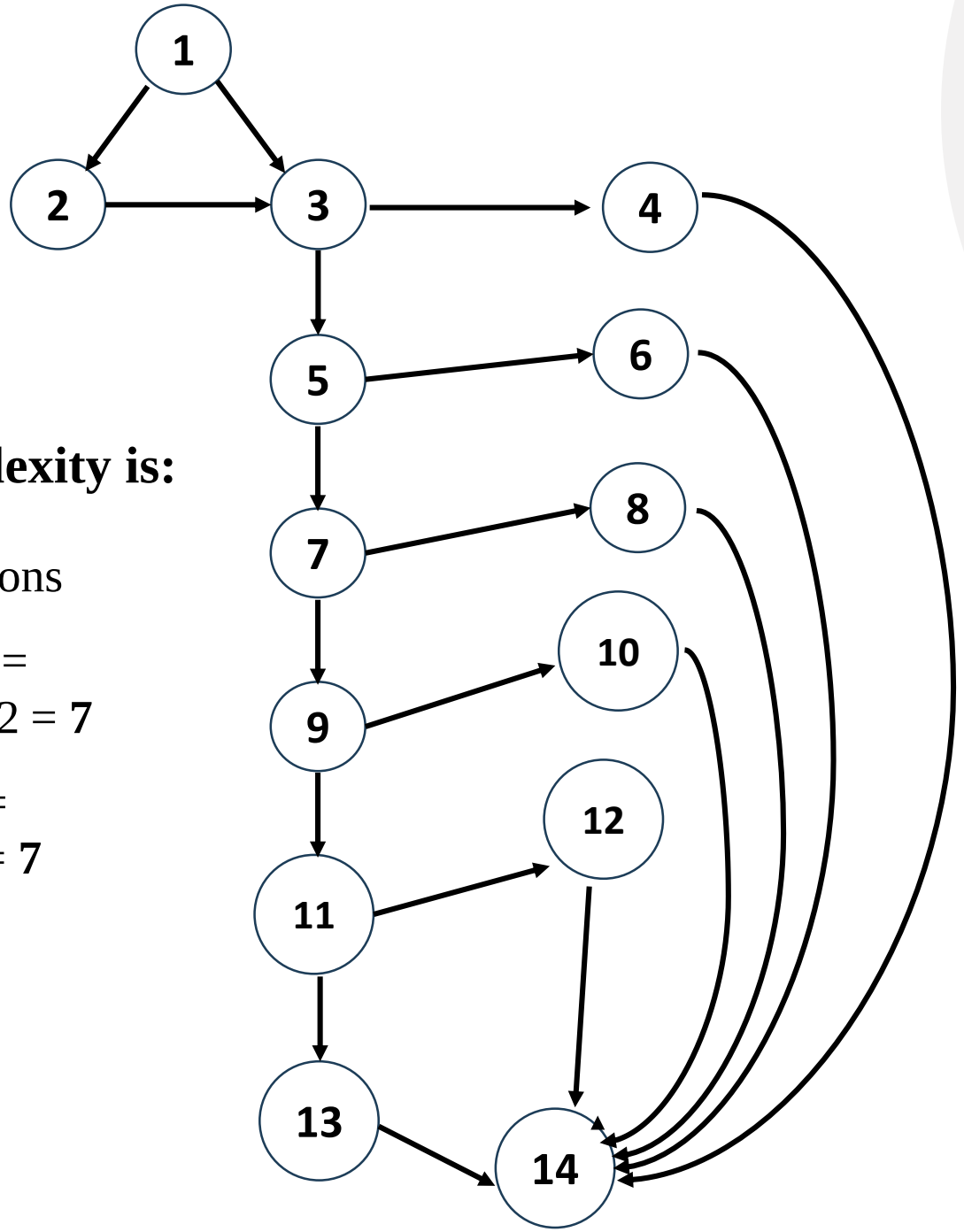
- Test cases should be designed to **force execution** of the independent paths *(basis set)*.

```
double calculate(int amount)
 {
     double rushCharge = 0;
     if (nextday=="yes" )
     {
         rushCharge = 14.50;
     }
    double tax = amount * .0725;
    if (amount >= 1000)
     {
        shipcharge = amount * .06 + rushCharge;
     }
    else if (amount >= 200)
     {
         shipcharge = amount * .08 + rushCharge;
     }
    else if (amount >= 100)
     {
        shipcharge = 13.25 + rushCharge;
     }
```

1

```
    else if (amount >= 50)
     {
        shipcharge = 9.95 + rushCharge;
     }
    else if (amount >= 25)
     {
        shipcharge = 7.25 + rushCharge;
     }
    else
    {
        shipcharge = 5.25 + rushCharge;
    }
    total = amount + tax + shipcharge;
    return total;
} //end calculate
```

2



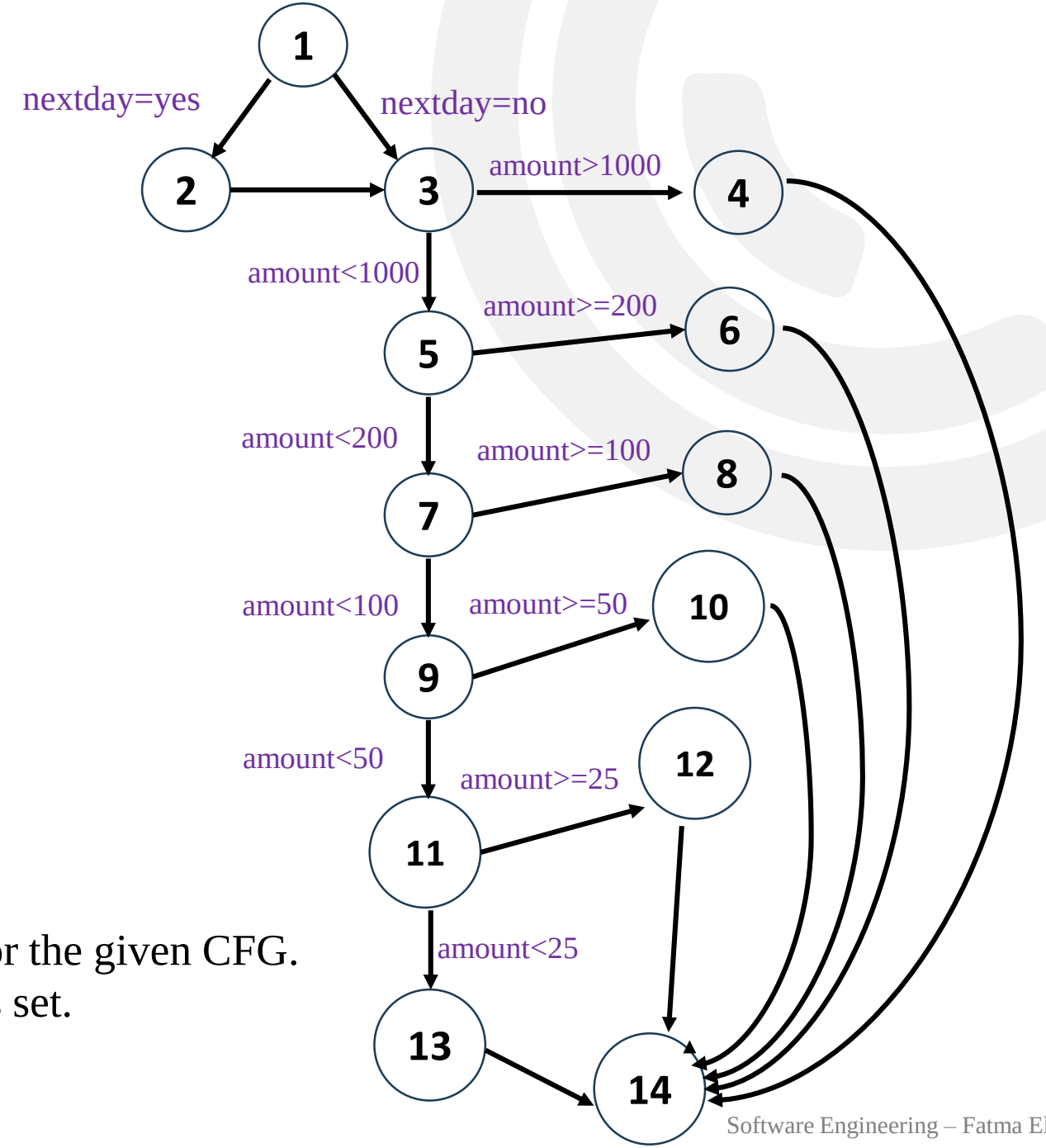
Cyclomatic complexity is:

1. V(G) = **7** Regions
2. V(G) = E-N+2 =
   19-14+2 = **7**
3. V(G) = P+1 =
   6+1 = **7**

```
public double calculate(int amount)
{
1   double rushCharge = 0;
    if (nextday=="yes" )
    {
2       rushCharge = 14.50;
    }
3   double tax = amount * .0725;
    if (amount >= 1000)
    {
4       shipcharge = amount * .06 + rushCharge;
    }
5   else if (amount >= 200)
    {
6       shipcharge = amount * .08 + rushCharge;
    }
7   else if (amount >= 100)
    {
8       shipcharge = 13.25 + rushCharge;
    }
9   else if (amount >= 50)
    {
10      shipcharge = 9.95 + rushCharge;
    }
11  else if (amount >= 25)
    {
12      shipcharge = 7.25 + rushCharge;
    }
    else
    {
13      shipcharge = 5.25 + rushCharge;
    }
14  total = amount + tax + shipcharge;
    return total;
} //end calculate
```

Basis set of independent paths:

1. 1-2-3-4-14
2. 1-2-3-5-6-14
3. 1-2-3-5-7-8-14
4. 1-2-3-5-7-9-10-14
5. 1-2-3-5-7-9-11-12-14
6. 1-2-3-5-7-9-11-13-14
7. 1-3-4-14



Note:

- This basis set is not unique.
- There are several different basis sets for the given CFG.
- You may have derived a different basis set.

Deriving Test Cases

- Test cases should be designed to **force execution** of the independent paths *(basis set)*.

Case #	Path	Input	Expected outcomes
1	1-2-3-4-14	yes, amount = 1500	1713.25
2	1-2-3-5-6-14	yes, amount = 300	360.25
3	1-2-3-5-7-8-14	yes, amount = 120	156.45
4	1-2-3-5-7-9-10-14	yes, amount = 90	120.98
5	1-2-3-5-7-9-11-12-14	yes, amount = 40	64.65
6	1-2-3-5-7-9-11-13-14	yes, amount = 10	30.47
7	1-3-4-14	no, amount = 1300	1472.25

Testing Stages

Integration Testing

- When distinct modules are combined, we need to verify they interact correctly. We can approach this from two perspectives:

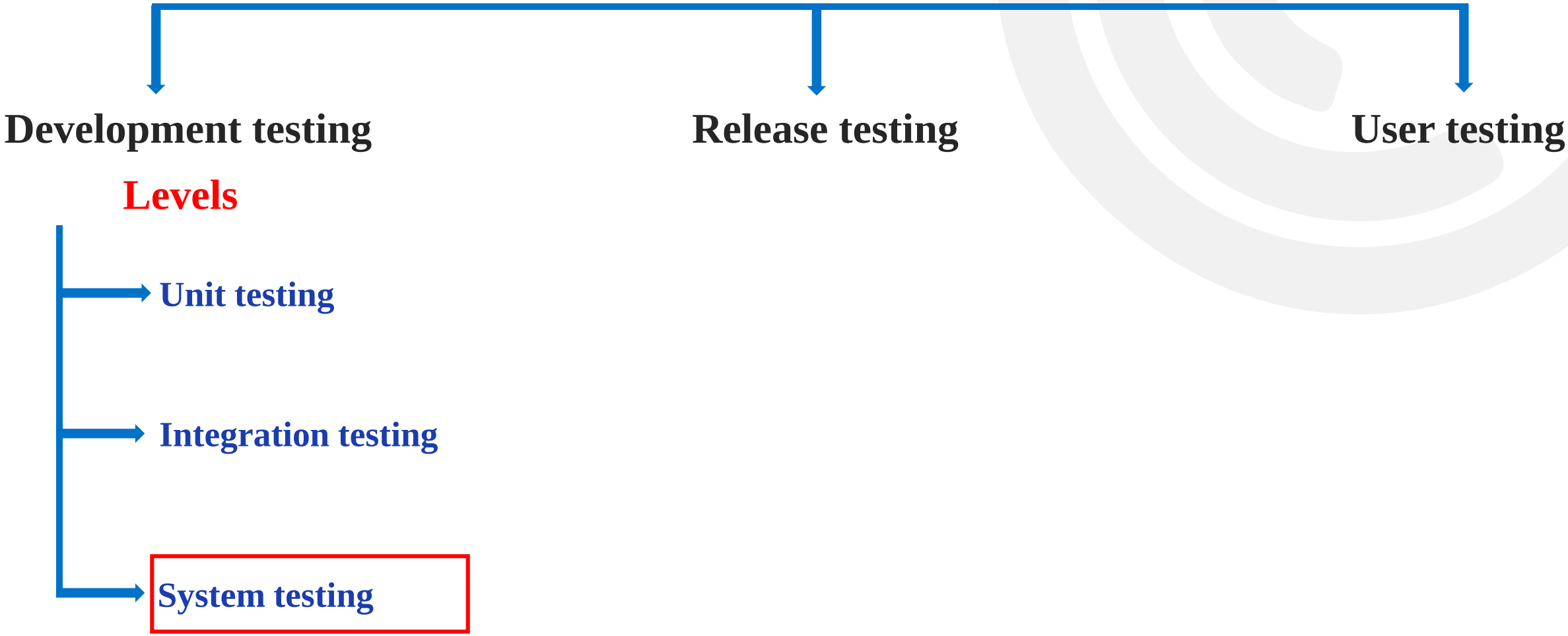
Black-Box

- Focus: Validating interfaces and external behavior.
- No need to know internal code
- The user view: We only care that specific Inputs produce the Expected Outputs.

White-Box

- Focus: How modules connect internally
- Tests data flow, and code-level interactions.
- The developer view: Examine code to ensure that modules communicate correctly and use the right commands and data formats.

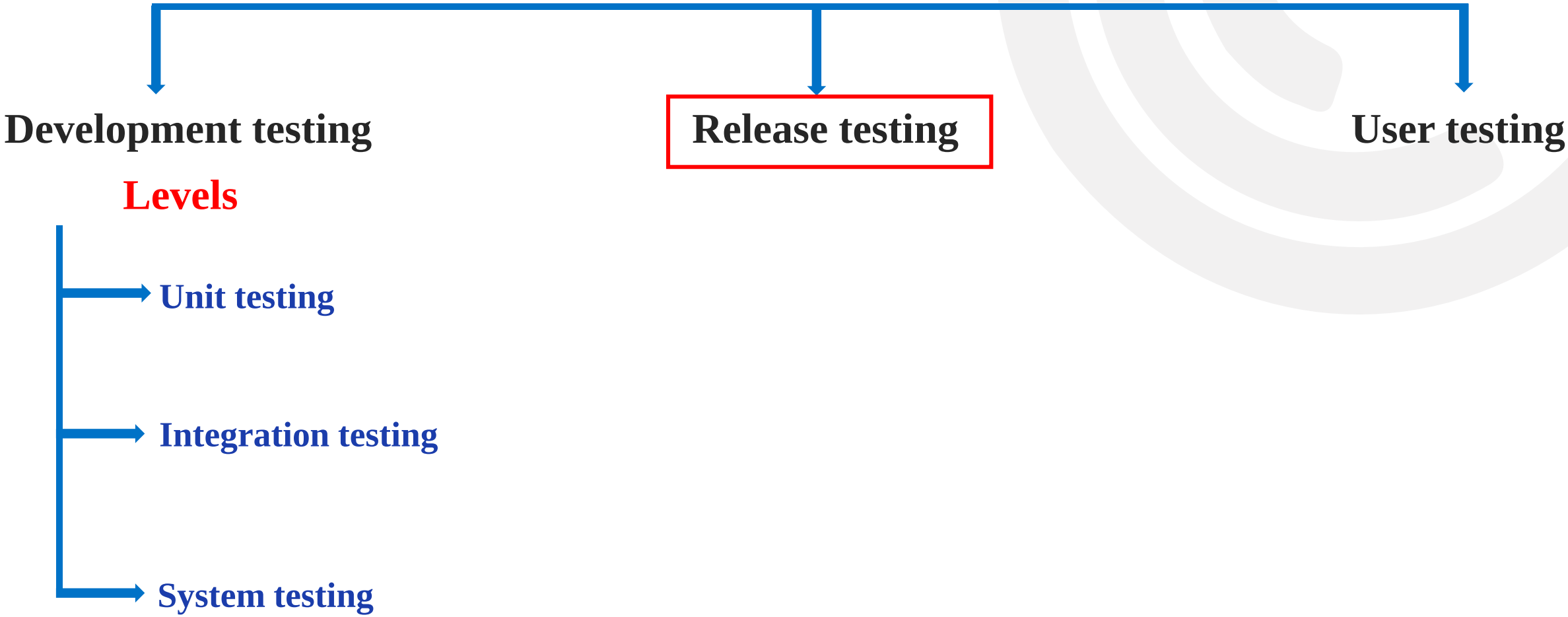
Testing Stages



System Testing

- System testing checks that **components are compatible, interact correctly and transfer the right data at the right time across their interfaces**.
- It is where all of the components in a system are integrated and the system is tested as a **whole**.
- Software is incorporated with other system elements (e.g., hardware, people, information), and a series of system integration and validation tests are conducted.

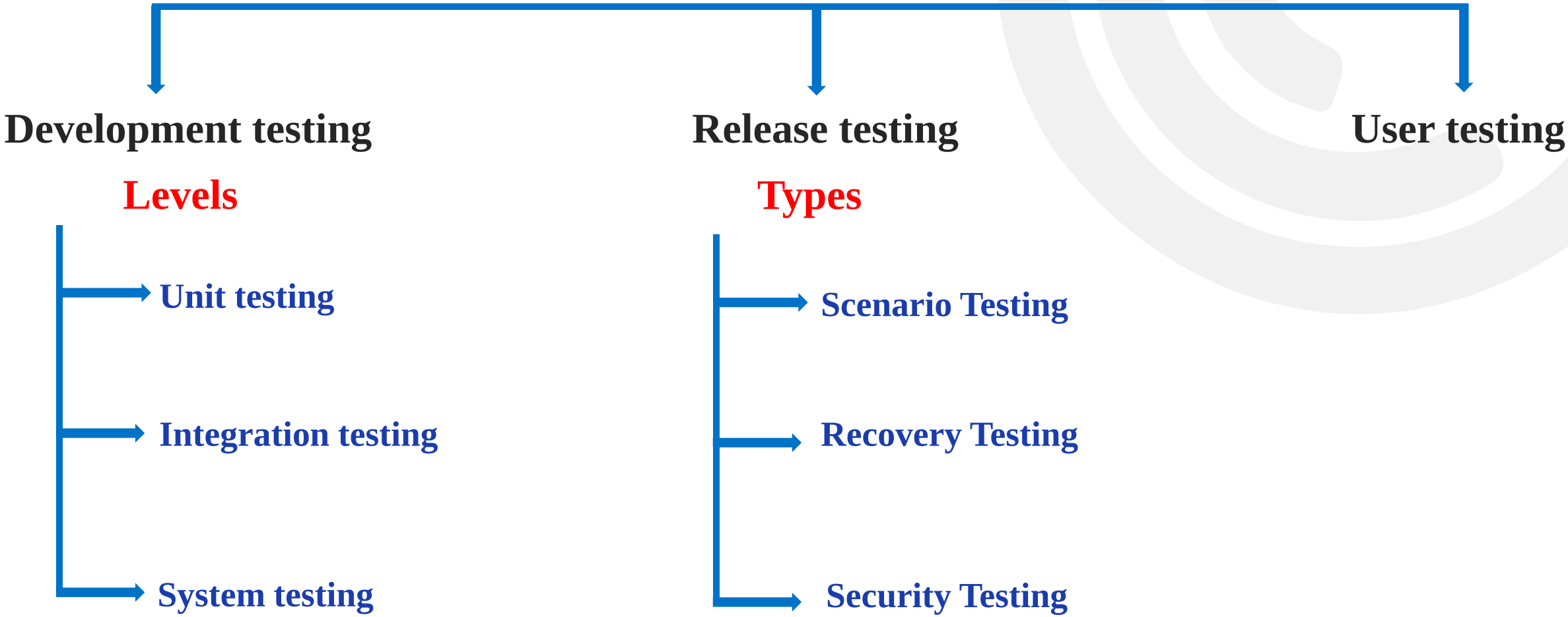
Testing Stages



Release Testing

- There are two important distinctions between **release** testing and **system** testing during the development process:
- A **separate team** that has not been involved in the system development should be responsible for release testing.
- System testing by the development team should focus on **discovering bugs in the system** (defect testing). The objective of release testing is to check that the system **meets its requirements** and is good enough for external use (validation testing).

Testing Stages



Testing Stages

Development testing

Levels

- Unit testing
- Integration testing
- System testing

Release testing

Types

- Scenario Testing
- Recovery Testing
- Security Testing

User testing

Types

- Alpha Testing
- Beta Testing

Alpha Testing

- It is conducted at the **developer's site by a customer.**
- It is done under **supervision** of the developer.
  - The software is used in a natural setting with the developer *"looking over the shoulder"* of the user and recording errors and usage problems

Beta Testing

- It is conducted at **customer sites** by the end-user of the software.
- Beta test is a "live" application of the software in an environment that cannot be controlled by the developer.
- The customer **records all problems** that are encountered during beta testing and **reports** these to the developer at regular intervals.
- As a result of problems reported during beta tests, software engineers make modifications and then prepare for release of the software product to the entire customer base.

Thank You