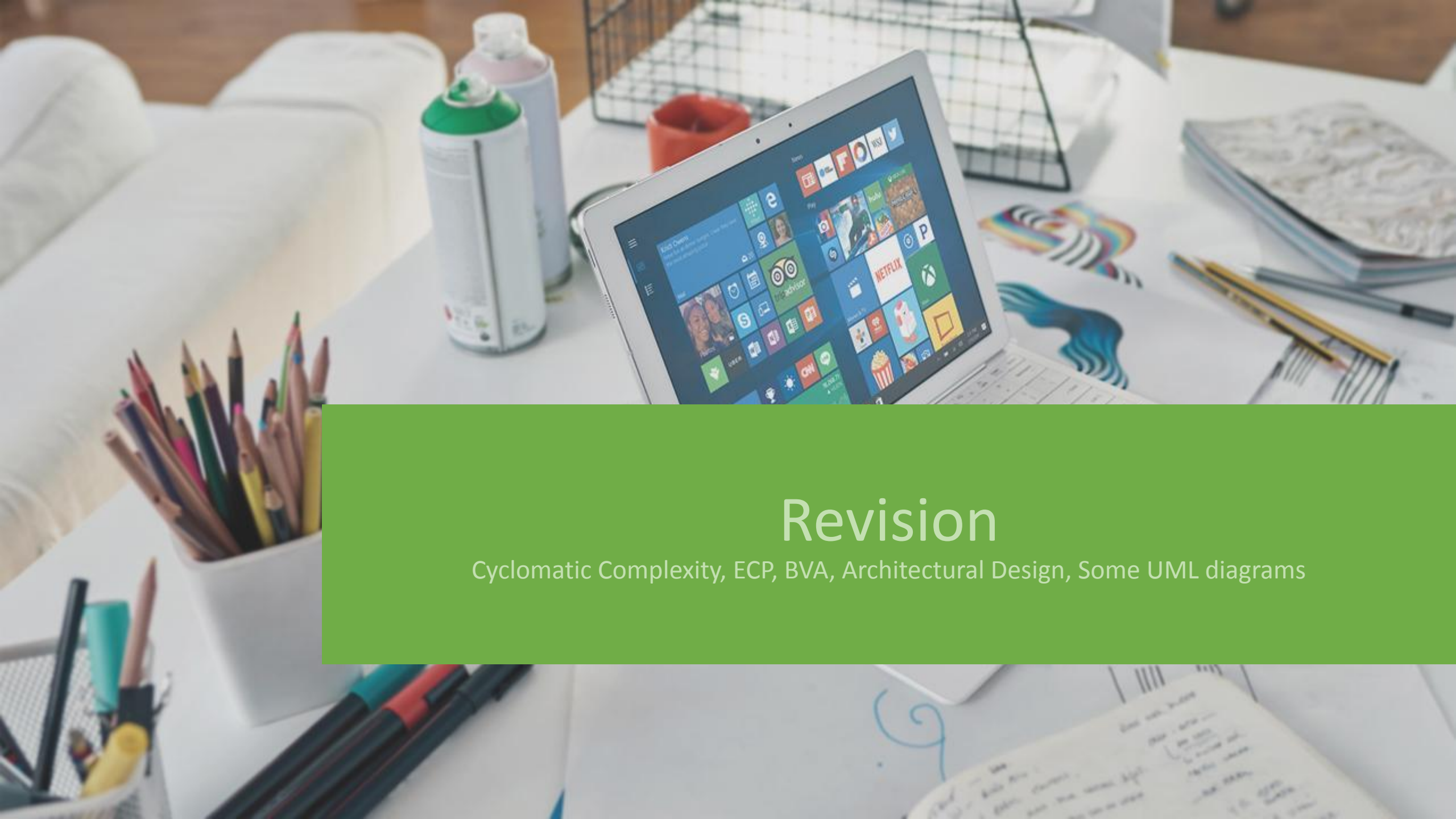


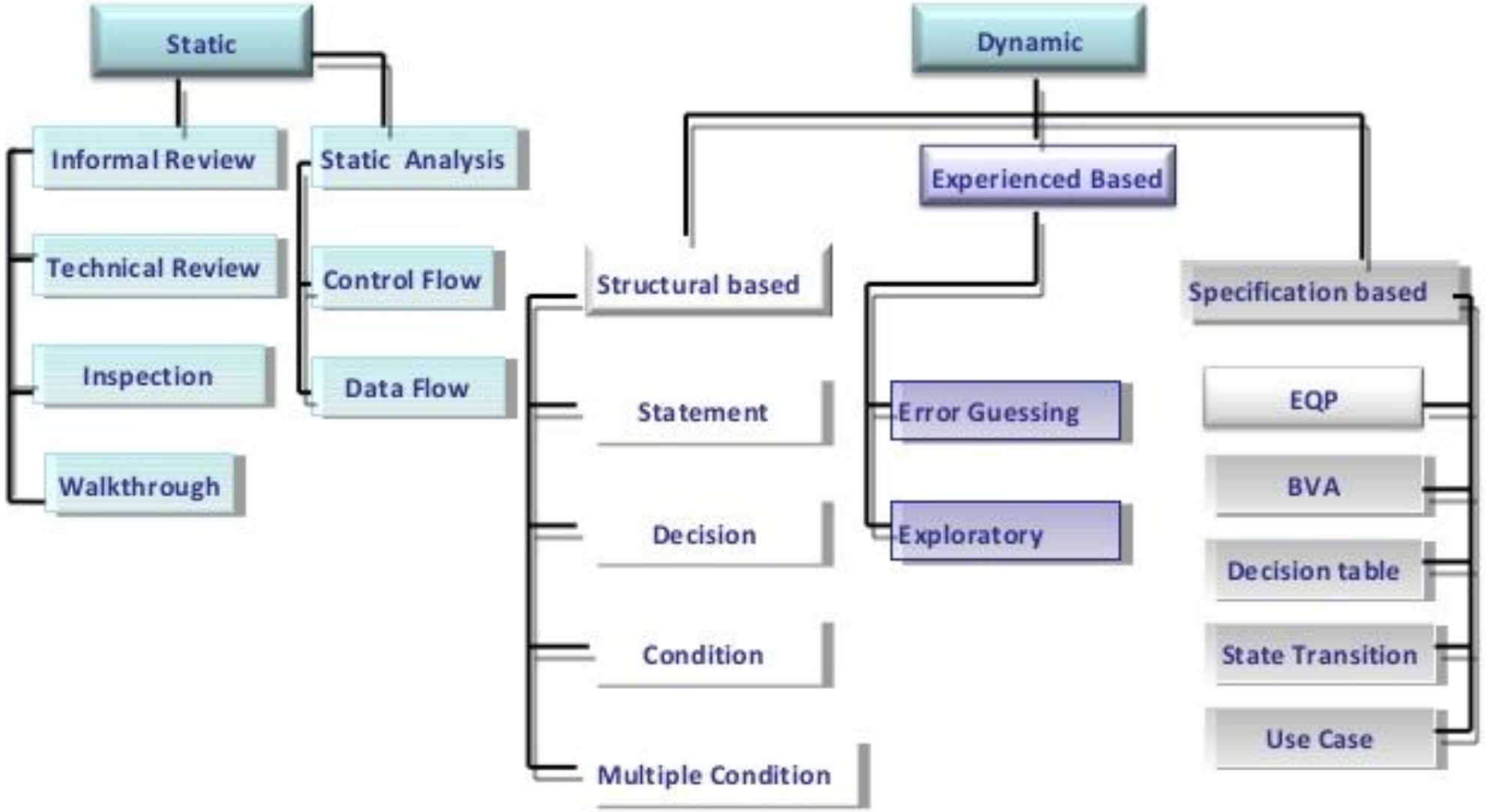
Revision

Cyclomatic Complexity, ECP, BVA, Architectural Design, Some UML diagrams

Verification

Definition:	The process of evaluating **work-products (not the actual final product)** of a development phase to determine whether they meet the specified requirements for that phase.
Objective:	To ensure that the product is being built according to the requirements and design specifications. In other words, to ensure that work products meet their specified requirements.
Question:	Are we building the product *right*?
Evaluation Items:	Plans, Requirement Specs, Design Specs, Code, Test Cases
Activities:	Static Testing (Reviews, Walkthroughs, Inspections)

Test Design Techniques



Work Products that Can Be Examined by Static Testing

- Specifications (SRS, FS, and Security Requirements etc)
- User stories, and Acceptance criteria
- Architecture and Design Specifications
- Code
- Testware (Test plans, Test Cases, Test Procedures, and Automated Test Scripts)
- User guides
- Web pages
- Contracts, Project Plans, Schedules, and Budgets

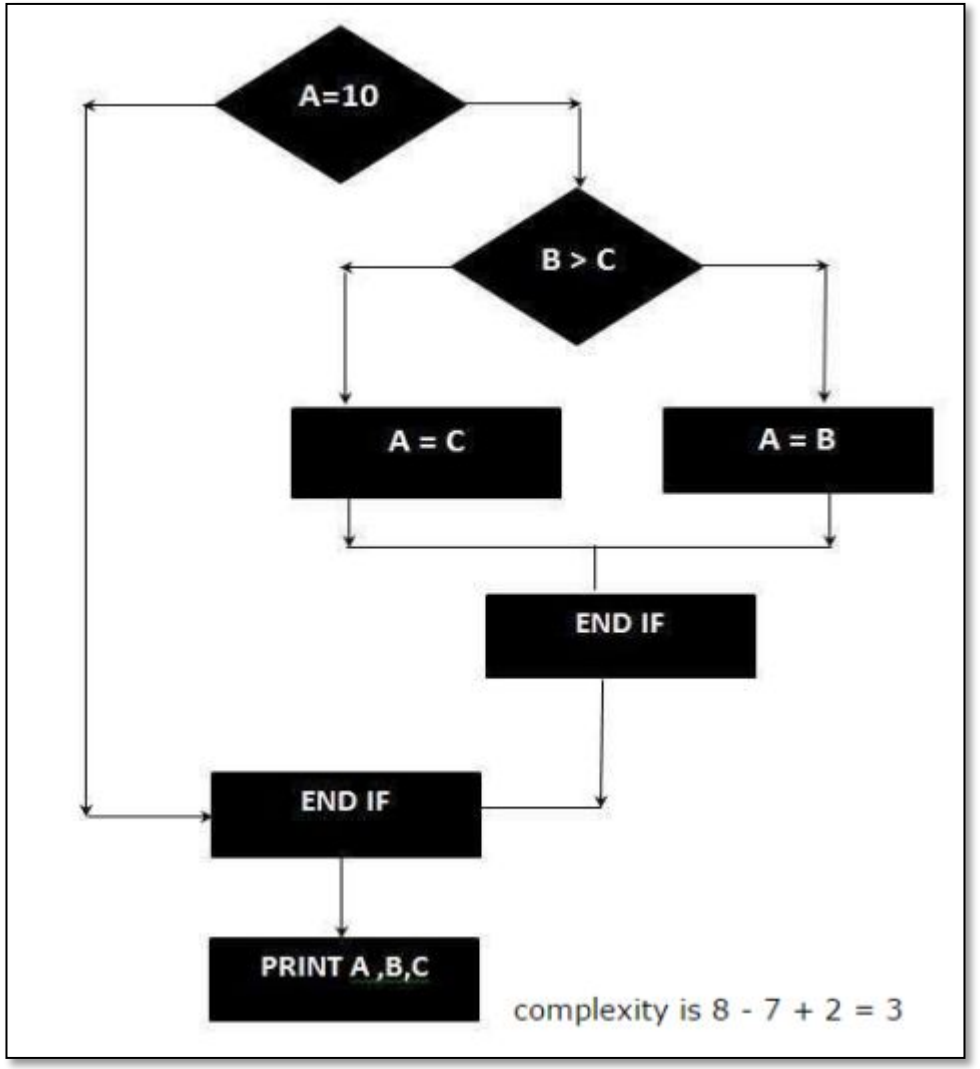
Static Analysis – CODE METRIC

Cyclomatic Code Complexity

CC = E – N + 2P

E = Number of Edges
N = Number of Nodes in Graph (Shapes)
P = Exit Paths

```
IF A = 10 THEN
 IF B > C THEN
   A = B
 ELSE
   A = C
 ENDIF
ENDIF
Print A
Print B
Print C
```



Cyclomatic Complexity

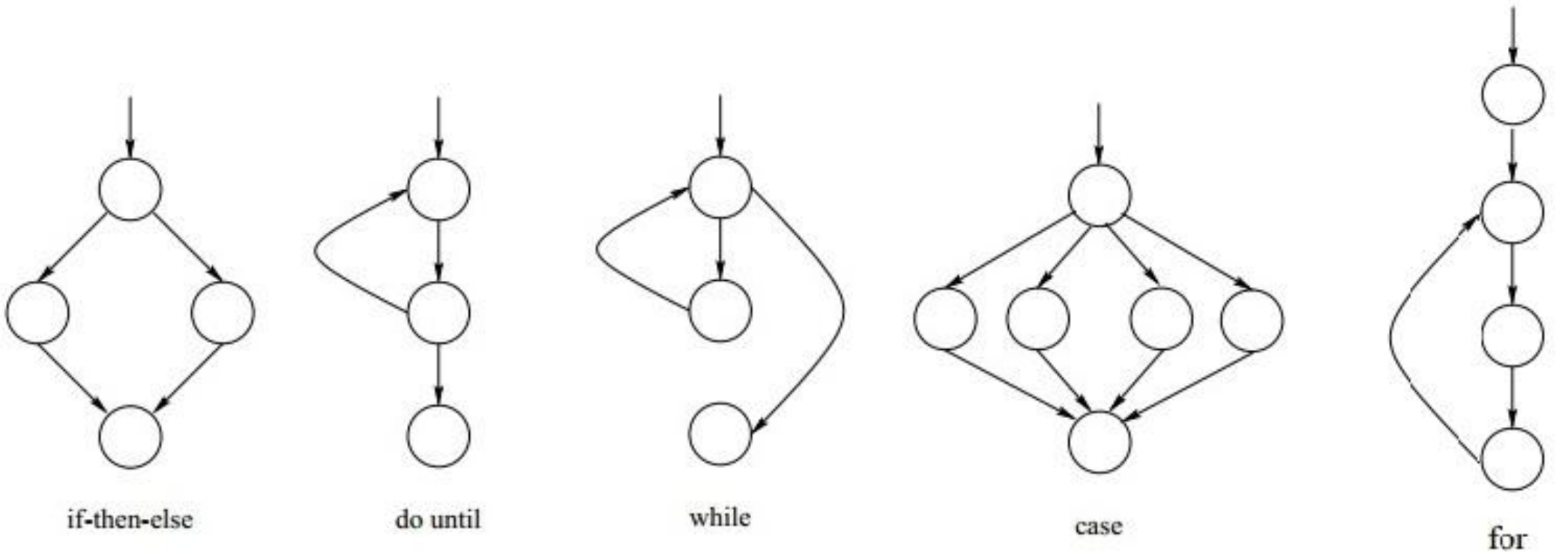


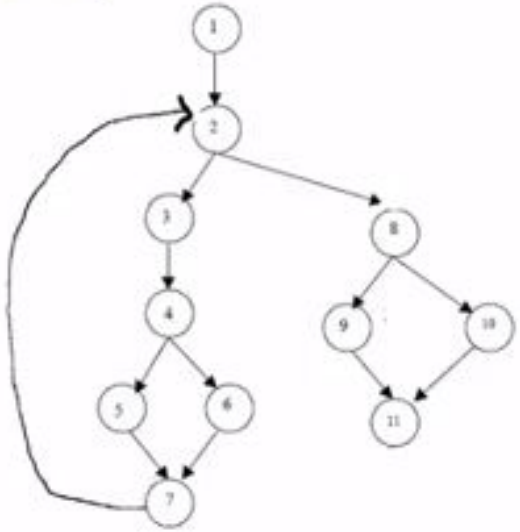
Figure 1: Flow graph representation.

Cyclomatic Complexity

```
1. INPUT A
   INPUT B
   INPUT C
2. WHILE ( (A<20) )
3. PRINT A+B
4. IF (A==C)
5. PRINT A
6. ELSE
   PRINT C
7. END OF DO
8. IF(C<=100)
9 . PRINT A+B
10 . ELSE
     PRINT A-B
11 . END OF PROGRAM
```

a) Draw the flow graph .
b) Determine the cyclomatic complexity
c) Determine a basis set of

solution



V(G) = E - N + 2
E number of edges
N number of nodes
V(G) = P + 1
P number of pridecate nodes

Path1: 1-2-8-9-11
Path2: 1-2-8-10-11
Path3: 1-2-3-4-5-7
Path4: 1-2-3-4-6-7

Cyclomatic Complexity Ranges

Value range	Code information
1 - 10	• Simple and easy to understand • High testability
10 - 20	• Code is complex but still comprehensible • Medium testability
> 20	• Code is very complex and difficult to understand • Low testability
> 50	• Code is very complex and difficult to understand • Not testable

General Testing Principle

1. Testing shows the presence of defects, not their absence
2. Exhaustive testing is impossible
3. Early testing saves time and money
4. Defects cluster together
5. Beware of the pesticide paradox
6. Testing is context dependent
7. Absence-of-errors is a fallacy

Error/Failure Reasons

- Errors may occur for many reasons, such as:

- Time pressure.
- Human fallibility.
- Inexperienced or insufficiently skilled project participants.
- Miscommunication between project participants.
- Complexity of the code, design, architecture.
- Misunderstandings about intra-system and inter-system interfaces, especially when such intra-system and inter-system interactions are large in numbers.
- New, unfamiliar technologies.

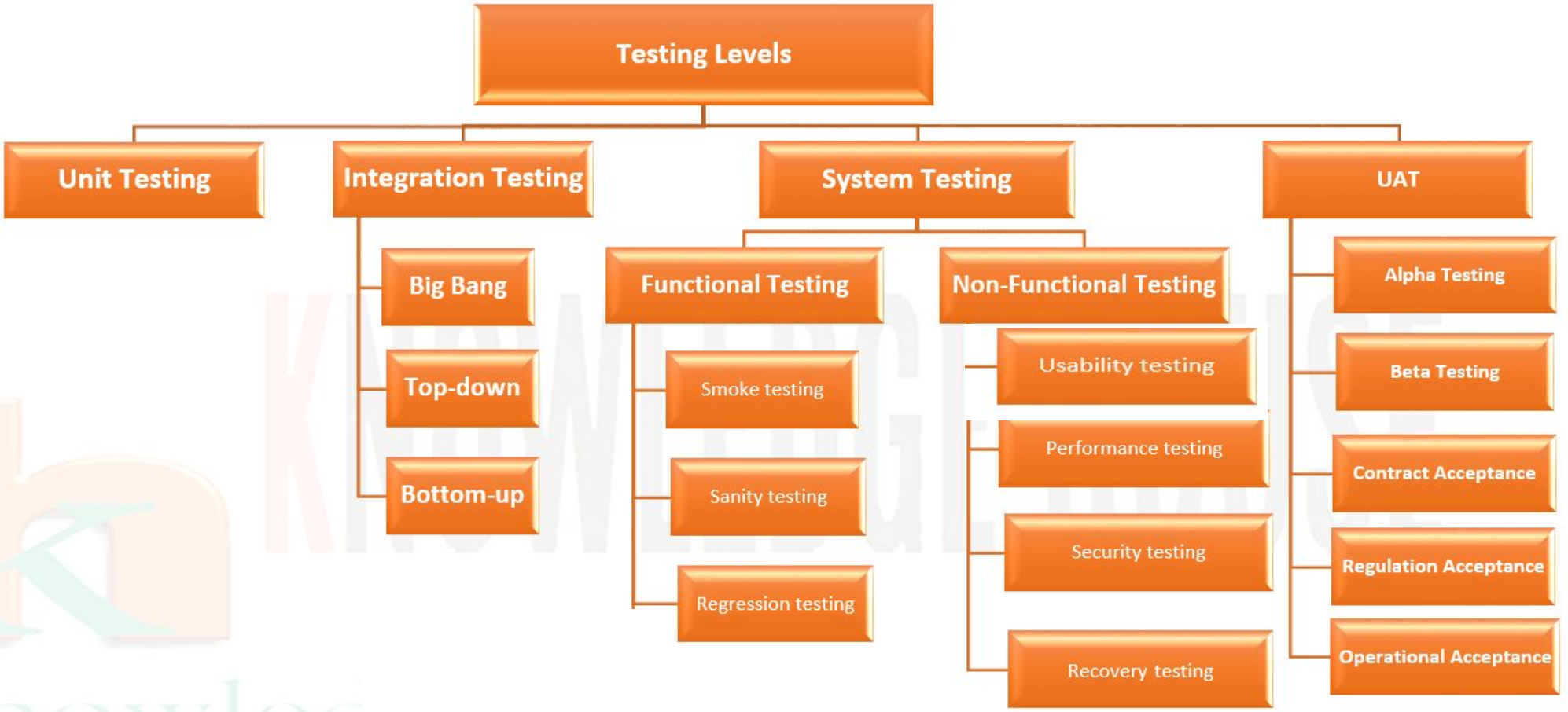
Testing Objectives

- To evaluate work products such as requirements, user stories, design, and code.
- To verify whether all specified requirements have been fulfilled.
- To validate whether the test object is completed and works as the users/stakeholders expect.
- To build confidence in the level of quality of the test object.
- To prevent defects.
- To find failures and defects.
- To reduce the level of risk of inadequate software quality (undetected failures in operation).
- To comply with contractual, legal, or regulatory requirements or standards.
- To provide sufficient information to stakeholders to allow them to make informed decisions, especially regarding the level of quality of the test object.

Test Levels, Type and Approaches

- Unit Test
- Integration Test
- System Test
- User Acceptance (UAT)
- Functional Testing
- Non-Functional Testing
- Smoke Testing
- Regression Testing
- Re-Testing
- Performance Testing
- White-BOX Testing
- Positive Testing
- Automated Testing
- Manual Testing
- Security Testing
- Sanity Testing
- Alpha Testing
- Beta Testing
- Factor Acceptance Test (FAT)
- Site Acceptance Test (SAT)
- Usability Testing
- Stress Testing
- Load Testing
- Compatibility Testing
- Black-BOX Testing
- Negative Testing
- Static Testing
- Dynamic Testing

Test Levels, Type and Approaches



Specification Based – Black Box

- **Boundary Value**
- **Equivalence Class Partitioning**

Example:

A bank add interest on the loan which is following

Loan Amount	Interest Rate	Test Inputs
0$ – 500$	3%	0, 250, 500
501$ – 1000$	5%	501, 750, 1000
1001$ – 1500$	10%	1001, 1250, 1500
1501$ – 2000$	20%	1501, 1750, 200

Specification Based – Black Box

Order numbers on a stock control system can range between 10000 and 99999 inclusive. Which of the following inputs might be a result of designing tests for only valid equivalence classes and valid boundaries:

A. 1000, 5000, 99999
B. 9999, 50000, 100000
C. 10000, 50000, 99999
D. 10000, 99999
E. 9999, 10000, 50000, 99999, 10000

Specification Based – Black Box

In a system designed to work out the tax to be paid:
An employee has £4000 of salary tax free.
The next £1500 is taxed at 10%
The next £28000 is taxed at 22%
Any further amount is taxed at 40%

To the nearest whole pound, which of these is a valid Boundary Value Analysis test case?

a) £1500
b) £32001
c) £33501
d) £28000

Specification Based – Black Box

A thermometer measures temperature in whole degrees only. If the temperature falls below 18 degrees, the heating is switched off. It is switched on again when the temperature reaches 21 degrees. What are the best values in degrees to cover all equivalence partitions?

A. 15, 19 and 25.
B. 17, 18 and19.
C. 18, 20 and22.
D. 16, 26 and 32.

Specification Based – Black Box

In a system designed to work out the tax to be paid:
An employee has £4000 of salary tax free.
The next £1500 is taxed at 10%
The next £28000 is taxed at 22%
Any further amount is taxed at 40%

Which of these groups of numbers would fall into the same equivalence class?

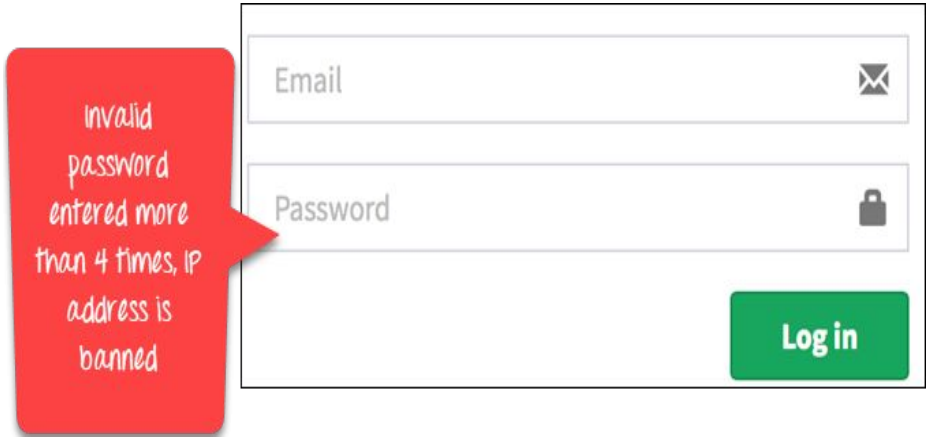
a) £4800; £14000; £28000
b) £5200; £5500; £28000
c) £28001; £32000; £35000
d) £5800; £28000; £32000

Specification Based – Black Box

Use Case Testing

In a use-case, an actor is represented by "A" and system by "S". We create Use for a login functionality of a Web Application as shown:

- Consider the first step of an end to end scenario for a login functionality for web application where the Actor enters email and password.
- In the next step, the system will validate the password
- Next, if the password is correct, the access will be granted
- There can be an extension of this use case. In case password is not valid system will display a message and ask for re-try four times
- If Password, not valid four times system will ban the IP address.



Main Scenario	Step	Description
A:Actor S:System	1	**A: Enter Agent Name & Password**
	2	**S: Validate Password**
	3	**S: Allow Account Access**
Extensions	2a	**Password not valid** S: Display Message and ask for re-try 4 times
	2b	**Password not valid 4 times** S: Close Application

Specification Based – Black Box

Use Case Testing

	Step	Description
Main Success Scenario **A: Actor** **S: System**	1	A: Inserts card
	2	S: Validates card and asks for PIN
	3	A: Enters PIN
	4	S: Validates PIN
	5	S: Allows access to account
Extensions	2a	Card not valid S: Display message and reject card
	4a	PIN not valid S: Display message and ask for re-try (twice)
	4b	PIN invalid 3 times S: Eat card and exit