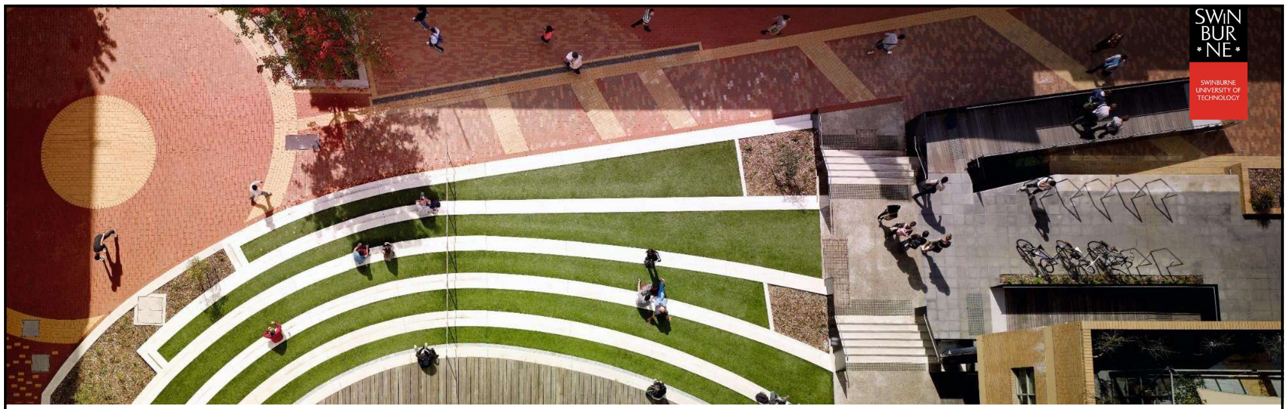


# COS80022 – Software Quality and Testing

*Week 9 – White-Box Testing*

## Outline

- White-box testing
  - Statement coverage
  - Branch coverage
  - Path coverage
- Experience-based testing

SWINBURNE UNIVERSITY OF TECHNOLOGY

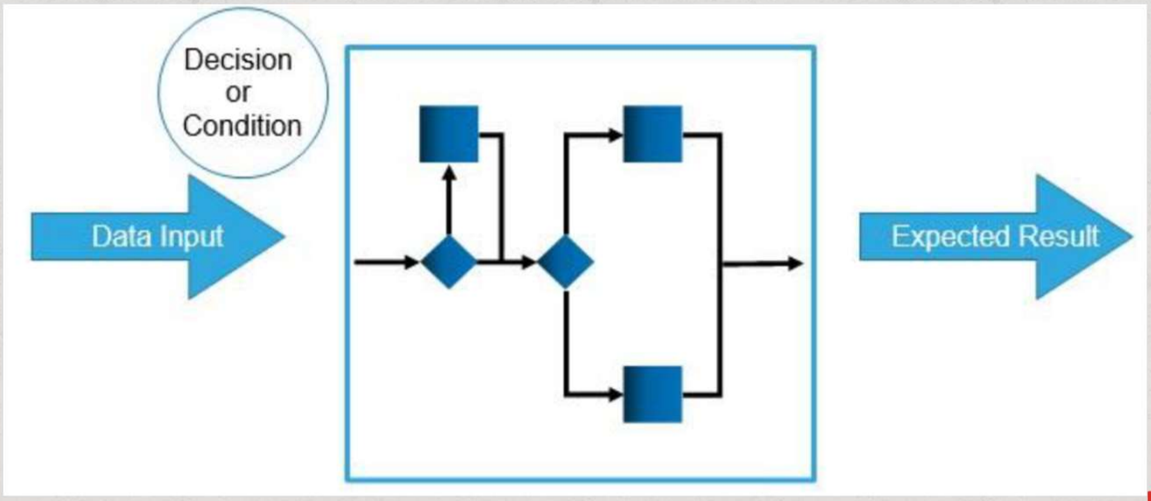
## White-Box Testing

## White-Box Testing

- Also called structural or structure-based techniques
- Based on an analysis of the architecture, detailed design, internal structure, or the code of the test object

KNOW ING

## White-Box Testing



KNOW ING

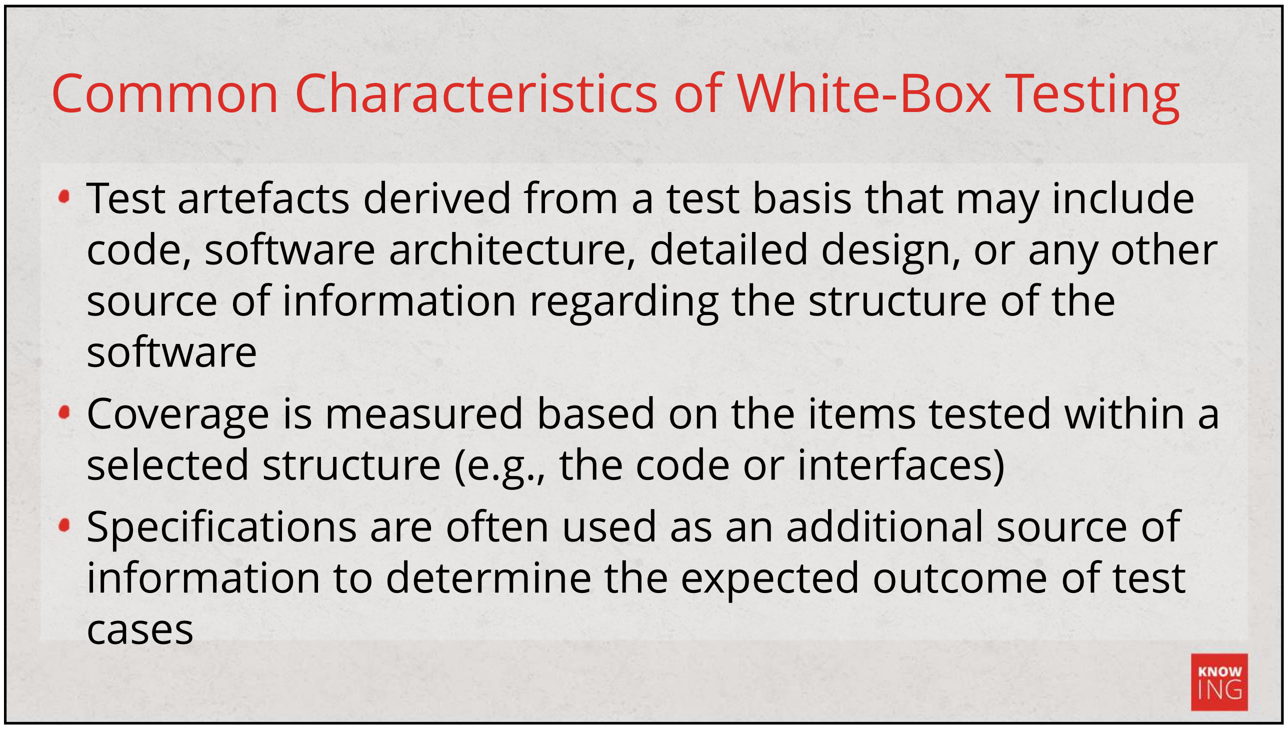
5

## White-Box Testing

- Based on an identified internal structure of the test object
- White-box test techniques can be used at all test levels
- The two code related white-box test techniques below are most commonly used at the component levels:
  – Statement testing
  – Decision testing

KNOW ING

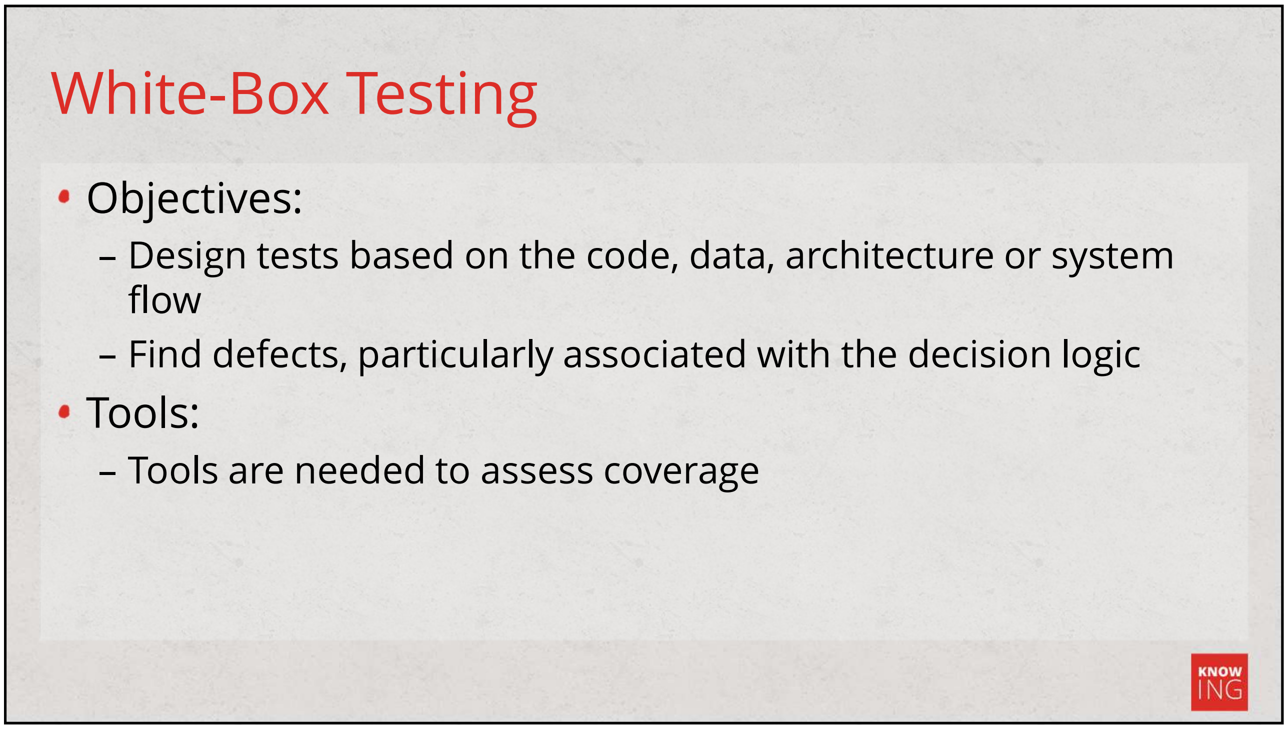
6



## Common Characteristics of White-Box Testing

- Test artefacts derived from a test basis that may include code, software architecture, detailed design, or any other source of information regarding the structure of the software
- Coverage is measured based on the items tested within a selected structure (e.g., the code or interfaces)
- Specifications are often used as an additional source of information to determine the expected outcome of test cases

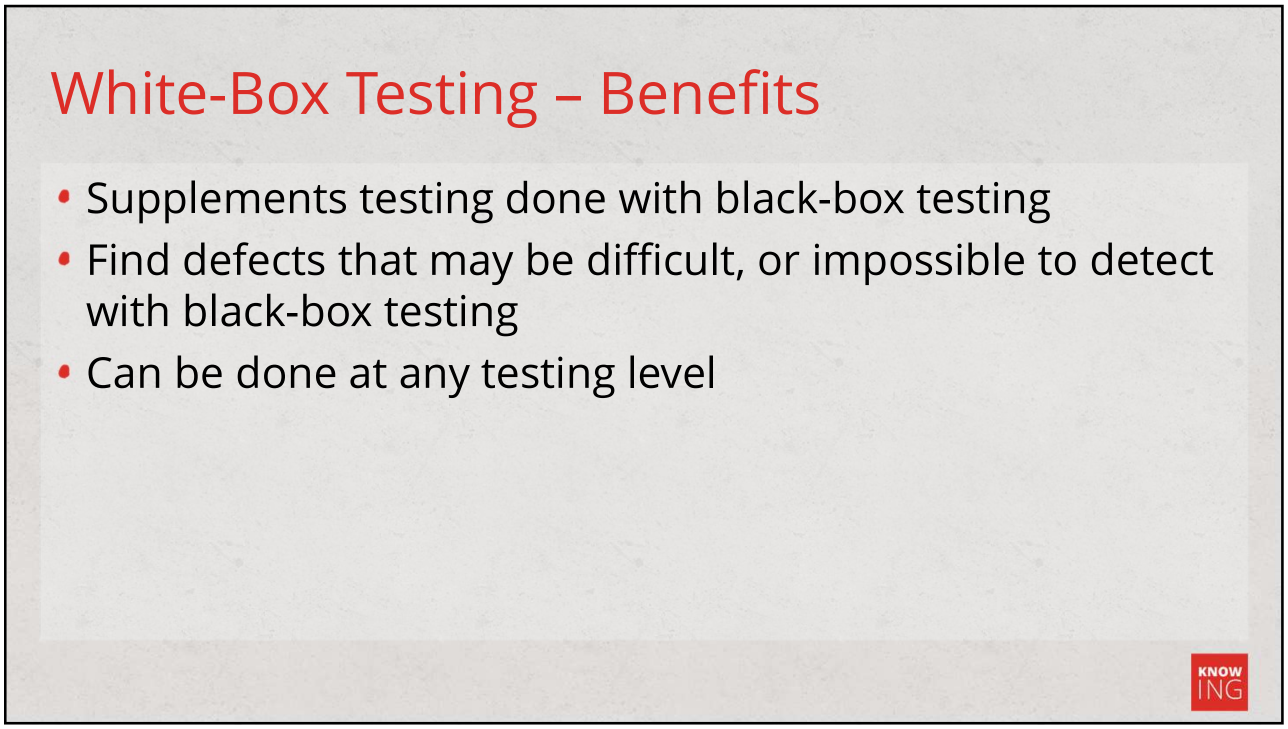
7



## White-Box Testing

- Objectives:
  - Design tests based on the code, data, architecture or system flow
  - Find defects, particularly associated with the decision logic
- Tools:
  - Tools are needed to assess coverage

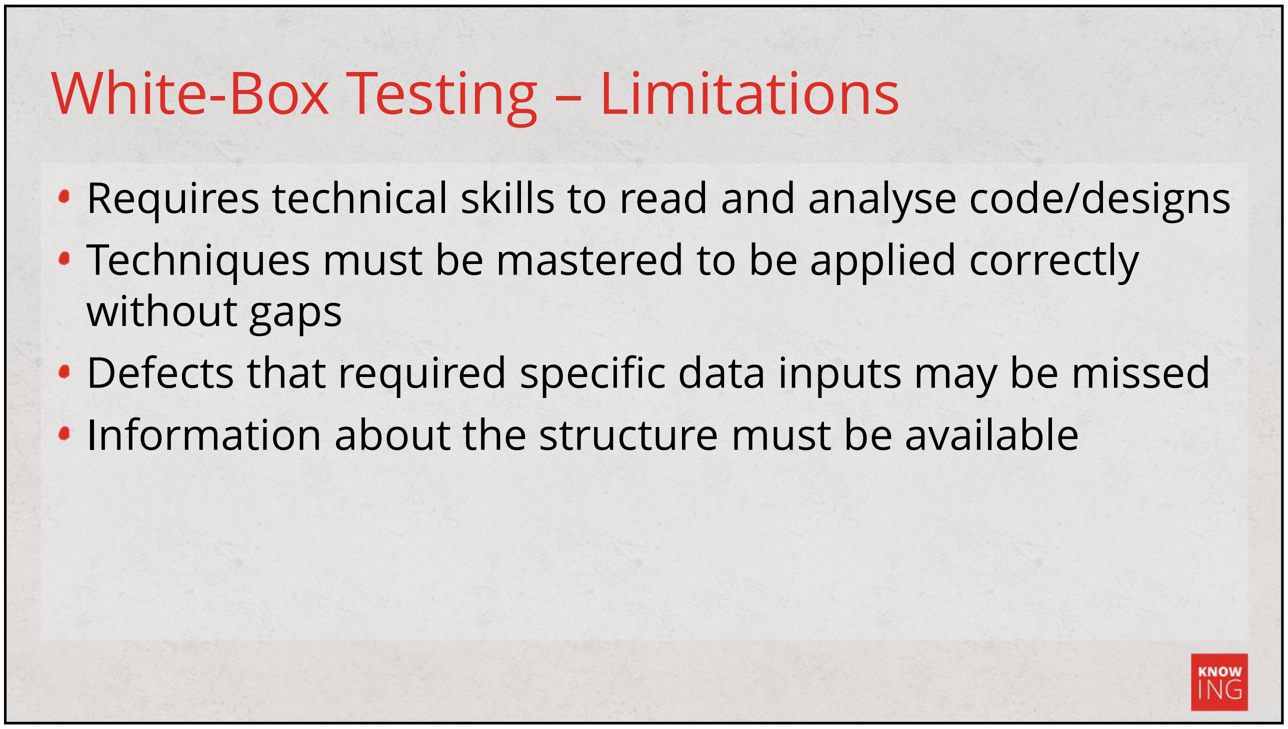
8



## White-Box Testing – Benefits

- Supplements testing done with black-box testing
- Find defects that may be difficult, or impossible to detect with black-box testing
- Can be done at any testing level

9

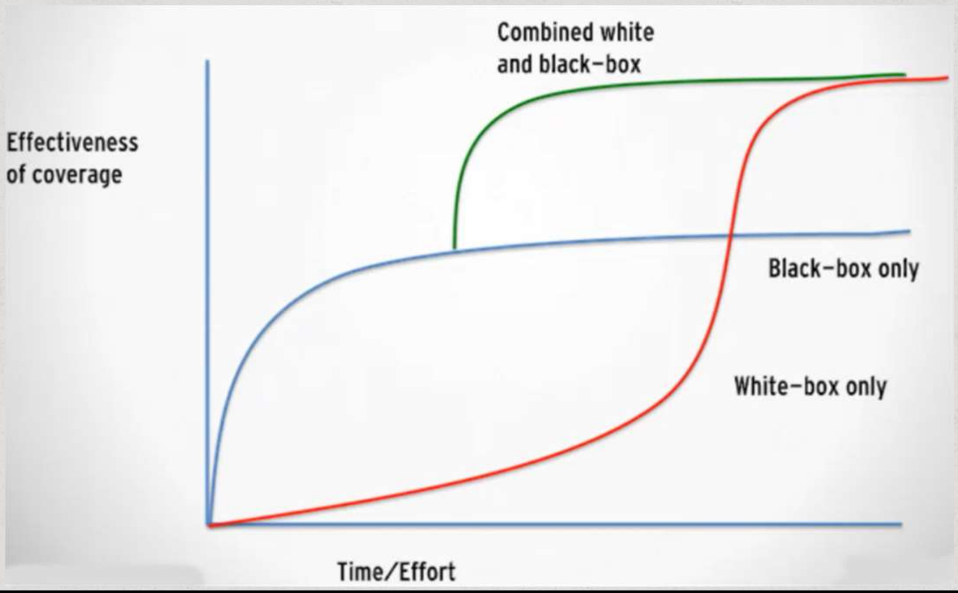


## White-Box Testing – Limitations

- Requires technical skills to read and analyse code/designs
- Techniques must be mastered to be applied correctly without gaps
- Defects that required specific data inputs may be missed
- Information about the structure must be available

10

## White-Box & Black-Box



KNOW ING

11

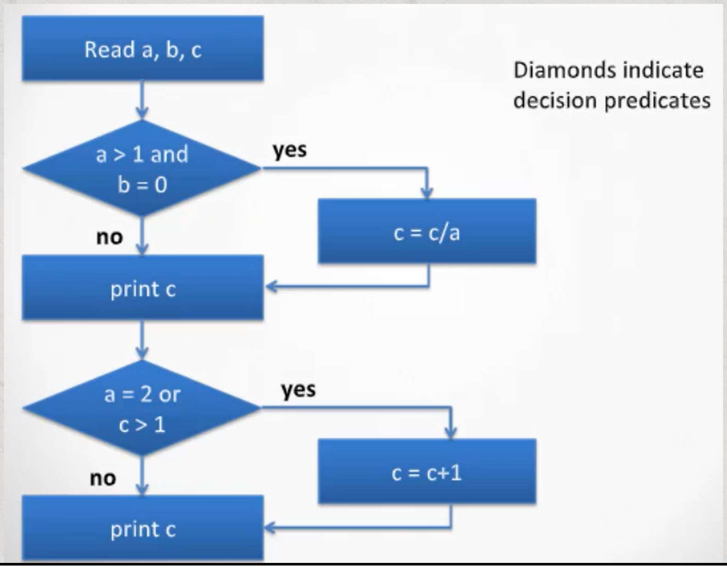
## White-Box Testing – Applicability

- Testing procedures such as backup, maintenance, etc.
- Testing control scripts such as batch processing scripts
- Testing UML or other design structures
- The **code**

KNOW ING

12

## White-Box Testing – Diagram



13

## White-Box Techniques

CRICOS 00111D
TOID 3059

14

## Statement Testing & Coverage

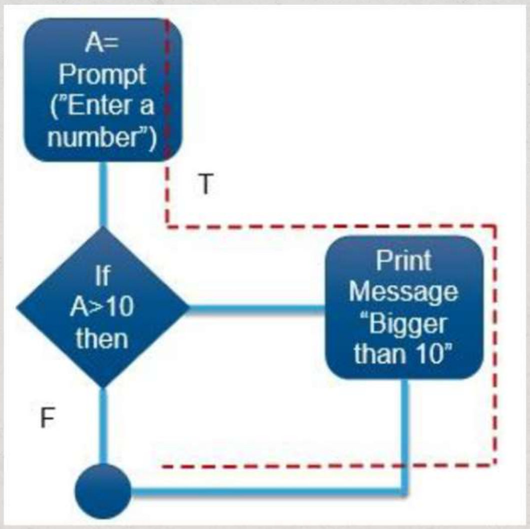
- Statement testing exercises the executable statements in the code
- Coverage is measured as the number of statements executed by the tests divided by the total number of executable statements in the test object, normally expressed as a percentage

15

## Statement Testing

**Program Segment**

```
A= Prompt ("Enter a
number")
If A > 10 then
    Print Message
    'Bigger then 10'
End If
```



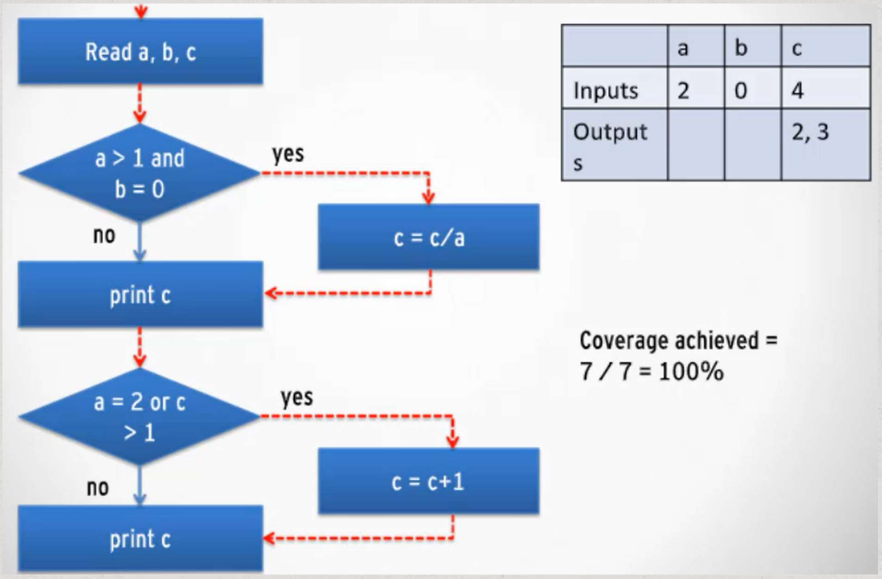
16

## Statement Testing

- Tests are designed to exercise executable statements (not comments)
- Applicability: Minimum approach, non-critical code
- Benefits:
  - Useful to determine coverage achieved in unit tests
  - Some would argue it's irresponsible to not do this as a minimum
- Limitations:
  - Not a substitute for higher levels of coverage
  - Requires having the code to design the tests

17

## Statement Testing



| | a | b | c |
|---|---|---|---|
| Inputs | 2 | 0 | 4 |
| Outputs | | | 2, 3 |

Coverage achieved = 7 / 7 = 100%

18

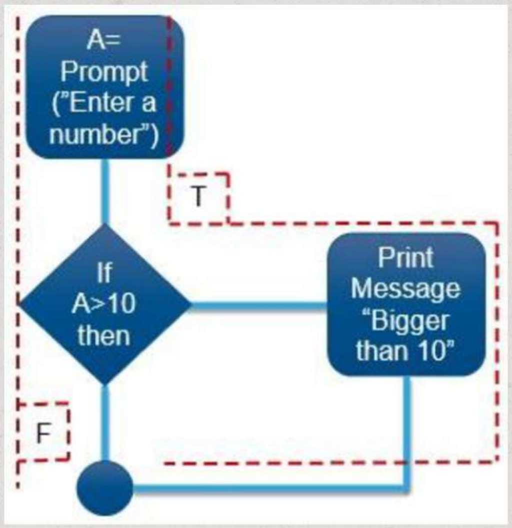
## Decision Testing & Coverage

- Decision testing exercises the decisions in the code and tests the code that is executed based on the decision outcomes
- To do this, the test cases follow the control flows that occur from a decision point
- Coverage is measured as the number of decision outcomes executed by the tests divided by the total number of decision outcomes in the test object, normally expressed as a percentage

## Decision Testing

**Program Segment**

```
A= Prompt ("Enter a
number")
If A > 10 then
      Print Message
      'Bigger then 10'
End If
```

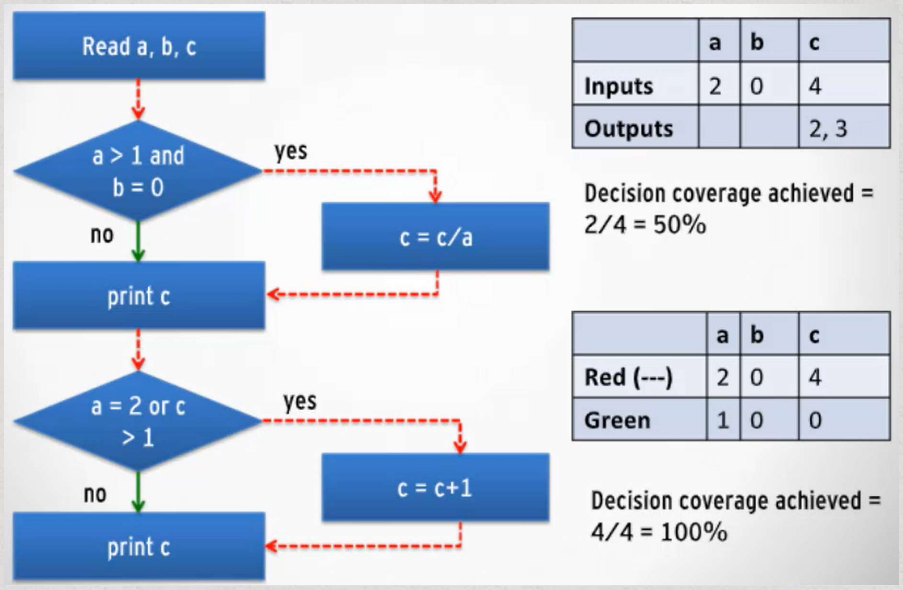


## Decision (Branch) Testing

- Tests are designed to exercise all possible decision outcomes. Decisions are found in if-endif, if-else-endif, case, loop constructs
- Applicability:
  - Stronger than statement testing
  - Verifies that both paths through the decisions are taken
- Benefits:
  - Catches code that might be missed with positive path testing
  - Verifies decision logic (what happens if you don't do the loop?)
- Limitations:
  - Better than statement, but not sufficient for safety-critical applications
  - Requires having the code to design the tests

21

## Decision Testing



| | a | b | c |
|---|---|---|---|
| **Inputs** | 2 | 0 | 4 |
| **Outputs** | | | 2, 3 |

Decision coverage achieved = 2/4 = 50%

| | a | b | c |
|---|---|---|---|
| **Red (---)** | 2 | 0 | 4 |
| **Green** | 1 | 0 | 0 |

Decision coverage achieved = 4/4 = 100%

22

## Value of Statement & Decision Testing

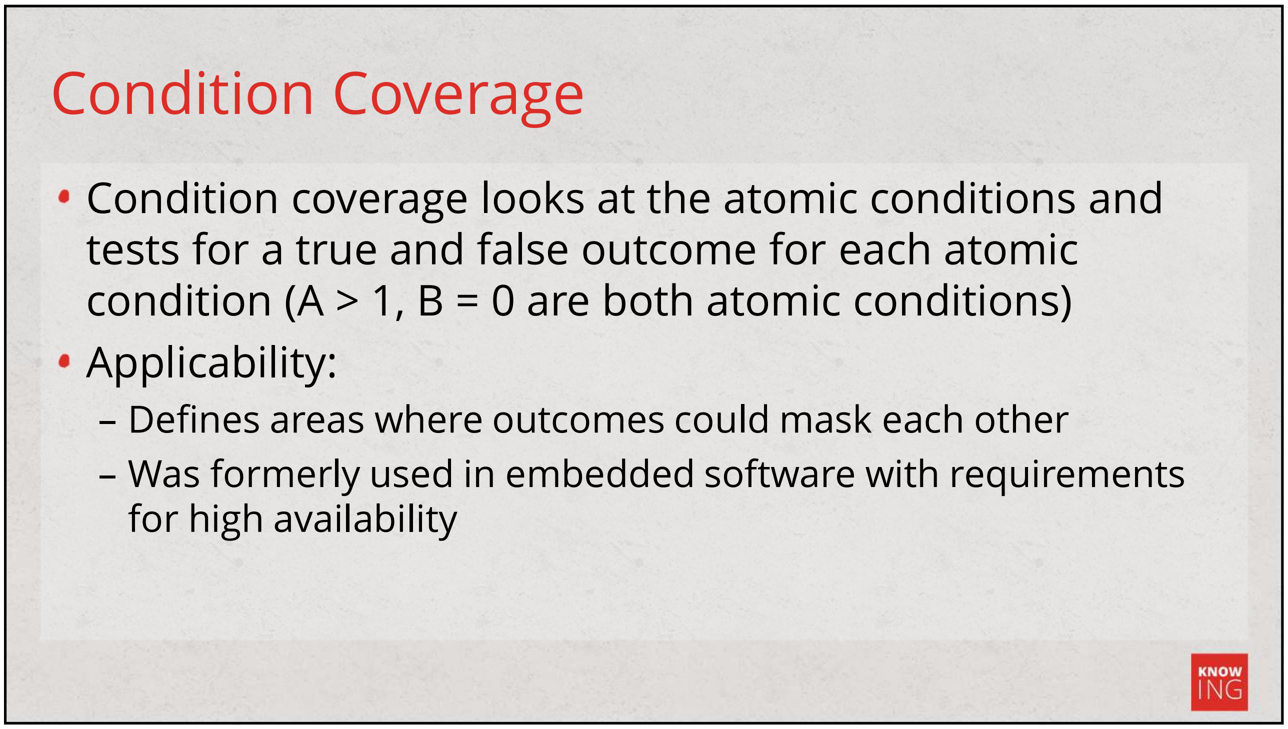
- Statement testing may provide less coverage than decision testing
- Statement coverage helps to find defects in code that was not exercised by other tests
- Decision coverage helps to find defects in code where other tests have not taken both true and false outcomes
- 100% decision coverage will guarantee 100% statement coverage; but not vice versa

23

## Other “Advanced” White-Box Techniques

- Condition coverage
- Decision condition coverage
- Multiple condition coverage
- MC/DC
- Path coverage

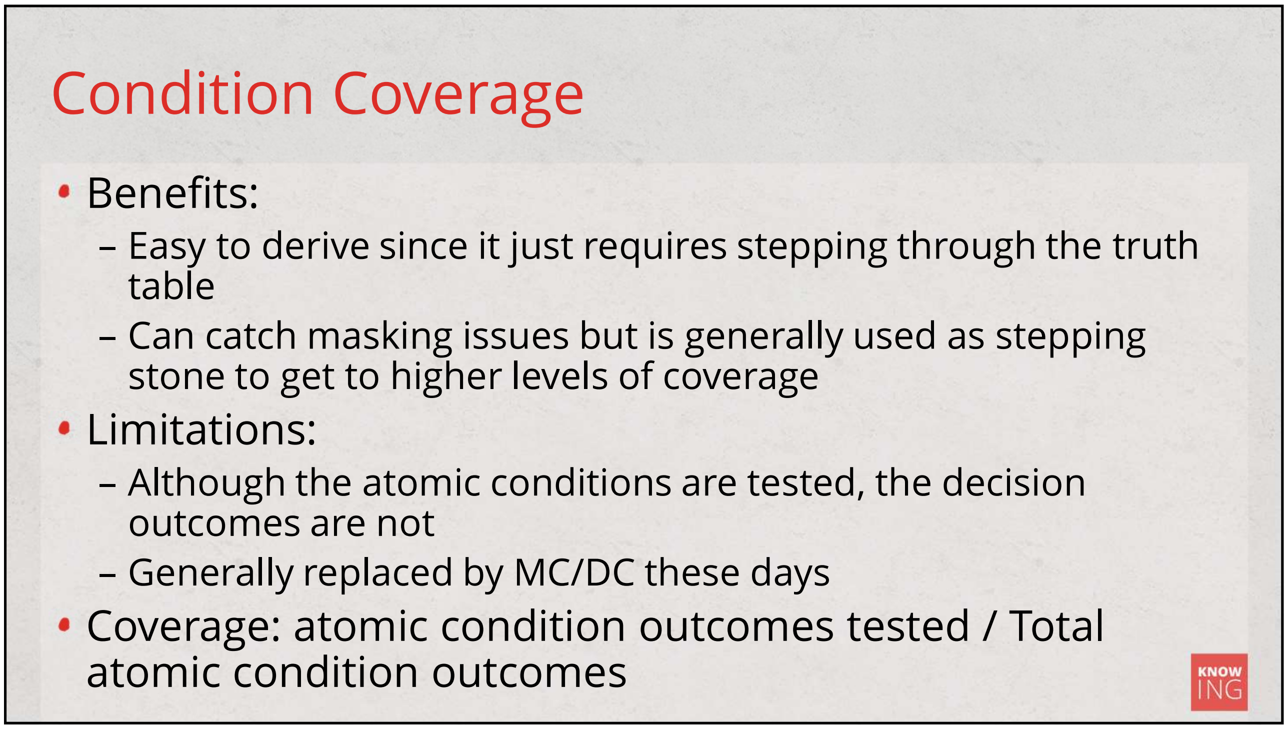
24



## Condition Coverage

- Condition coverage looks at the atomic conditions and tests for a true and false outcome for each atomic condition (A > 1, B = 0 are both atomic conditions)
- Applicability:
  - Defines areas where outcomes could mask each other
  - Was formerly used in embedded software with requirements for high availability

25

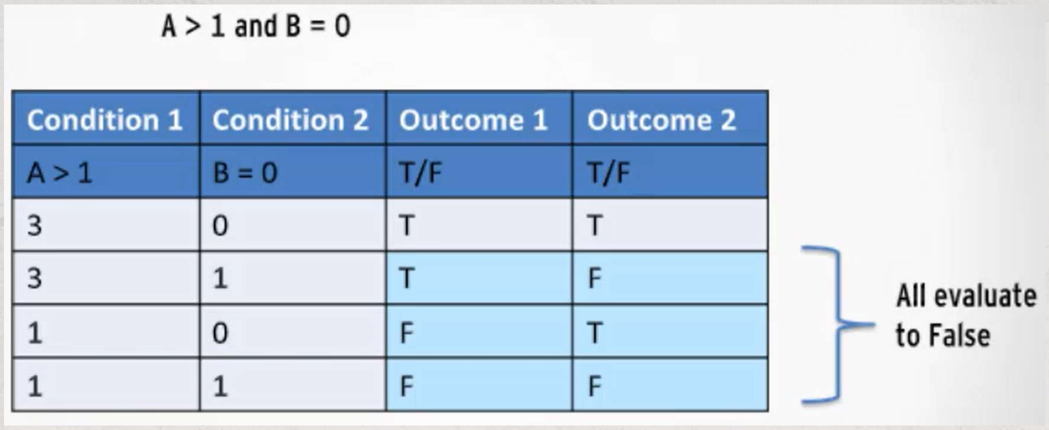


## Condition Coverage

- Benefits:
  - Easy to derive since it just requires stepping through the truth table
  - Can catch masking issues but is generally used as stepping stone to get to higher levels of coverage
- Limitations:
  - Although the atomic conditions are tested, the decision outcomes are not
  - Generally replaced by MC/DC these days
- Coverage: atomic condition outcomes tested / Total atomic condition outcomes

26

## Condition Coverage – Truth Table



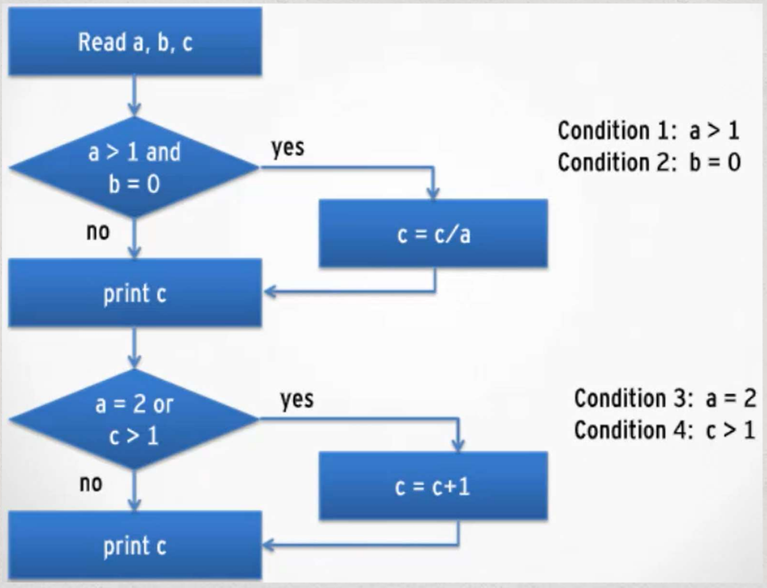
A > 1 and B = 0

| Condition 1 | Condition 2 | Outcome 1 | Outcome 2 |
|---|---|---|---|
| A > 1 | B = 0 | T/F | T/F |
| 3 | 0 | T | T |
| 3 | 1 | T | F |
| 1 | 0 | F | T |
| 1 | 1 | F | F |

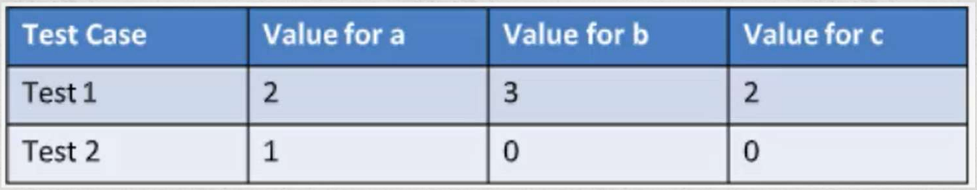
All evaluate to False

- Each atomic condition has to be evaluated to a true and a false result, but the decision outcome doesn't matter

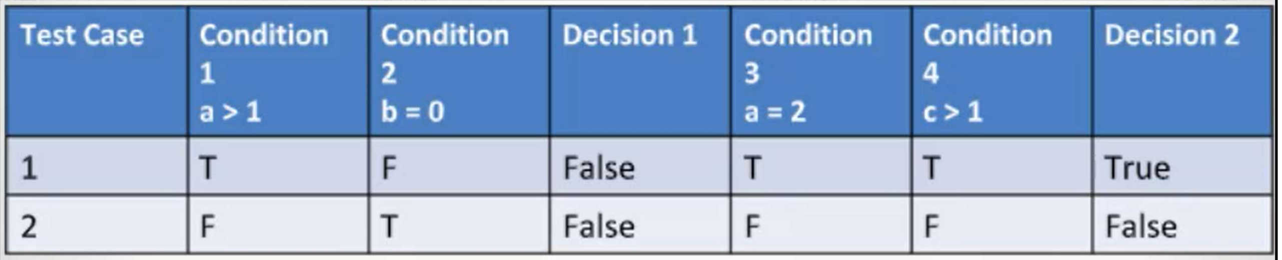
## Condition Coverage



## Condition Coverage



| Test Case | Value for a | Value for b | Value for c |
|---|---|---|---|
| Test 1 | 2 | 3 | 2 |
| Test 2 | 1 | 0 | 0 |



| Test Case | Condition 1 a > 1 | Condition 2 b = 0 | Decision 1 | Condition 3 a = 2 | Condition 4 c > 1 | Decision 2 |
|---|---|---|---|---|---|---|
| 1 | T | F | False | T | T | True |
| 2 | F | T | False | F | F | False |

Condition coverage √
Decision coverage ?

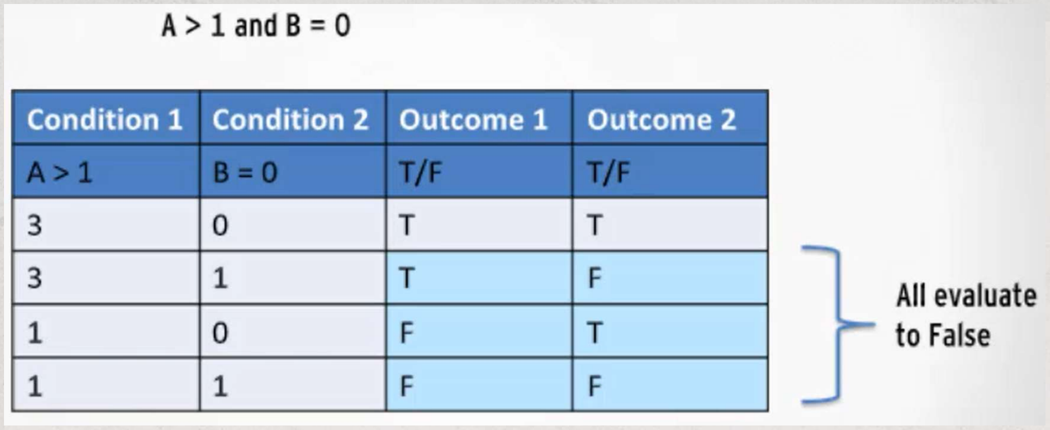
## Decision Condition Coverage

- Verifies T/F for the atomic conditions and T/F for the decision outcome
- Applicability: Used for code that is important, but not critical
- Benefits:
  - Catches issues that could be overlooked by simpler methods, particularly regarding the way that decisions are reached
  - Can be used efficiently if carefully designed

## Decision Condition Coverage

- Limitations:
  - Requires more test cases than basic decision coverage
  - Requires careful analysis and time
- Coverage: atomic conditions + outcomes tested / total atomic conditions + outcomes

## Decision Condition Coverage – Truth Table

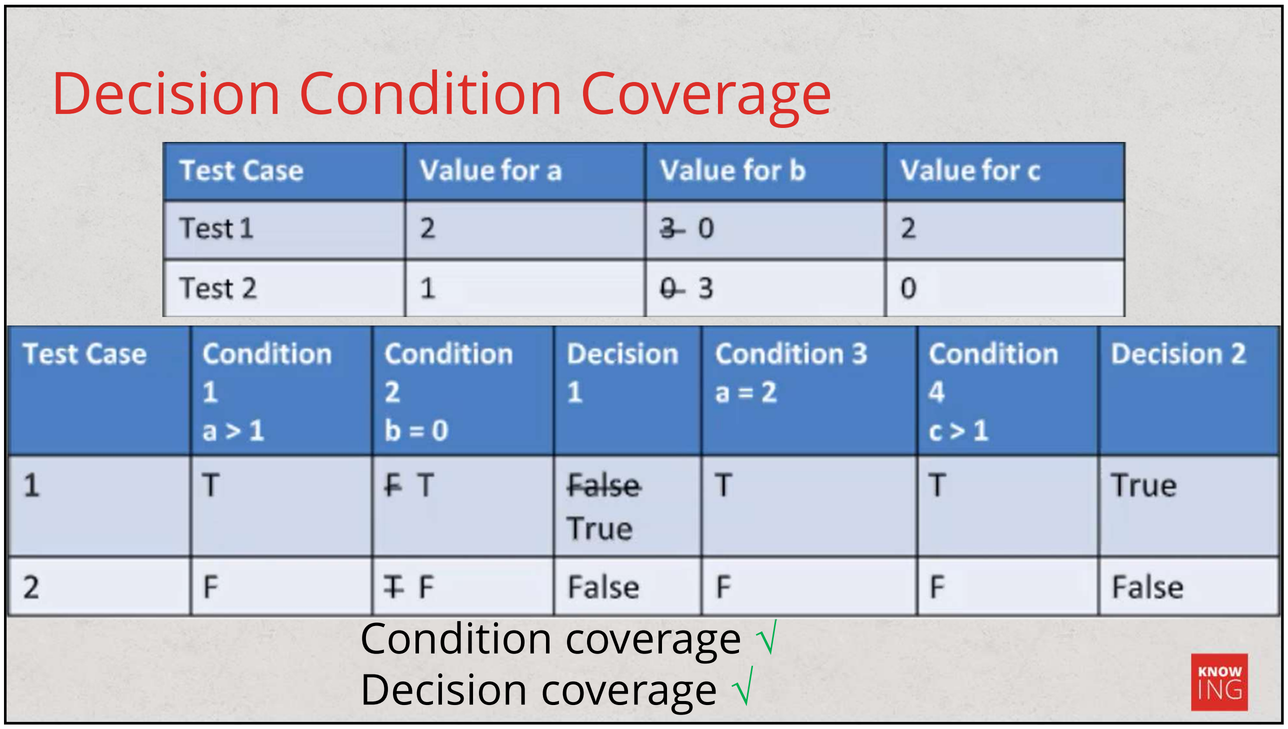


A > 1 and B = 0

| Condition 1 | Condition 2 | Outcome 1 | Outcome 2 |
|---|---|---|---|
| A > 1 | B = 0 | T/F | T/F |
| 3 | 0 | T | T |
| 3 | 1 | T | F |
| 1 | 0 | F | T |
| 1 | 1 | F | F |

All evaluate to False

- Each atomic condition has to be evaluated to a true and a false result, and each decision outcome must be evaluated to a true and a false



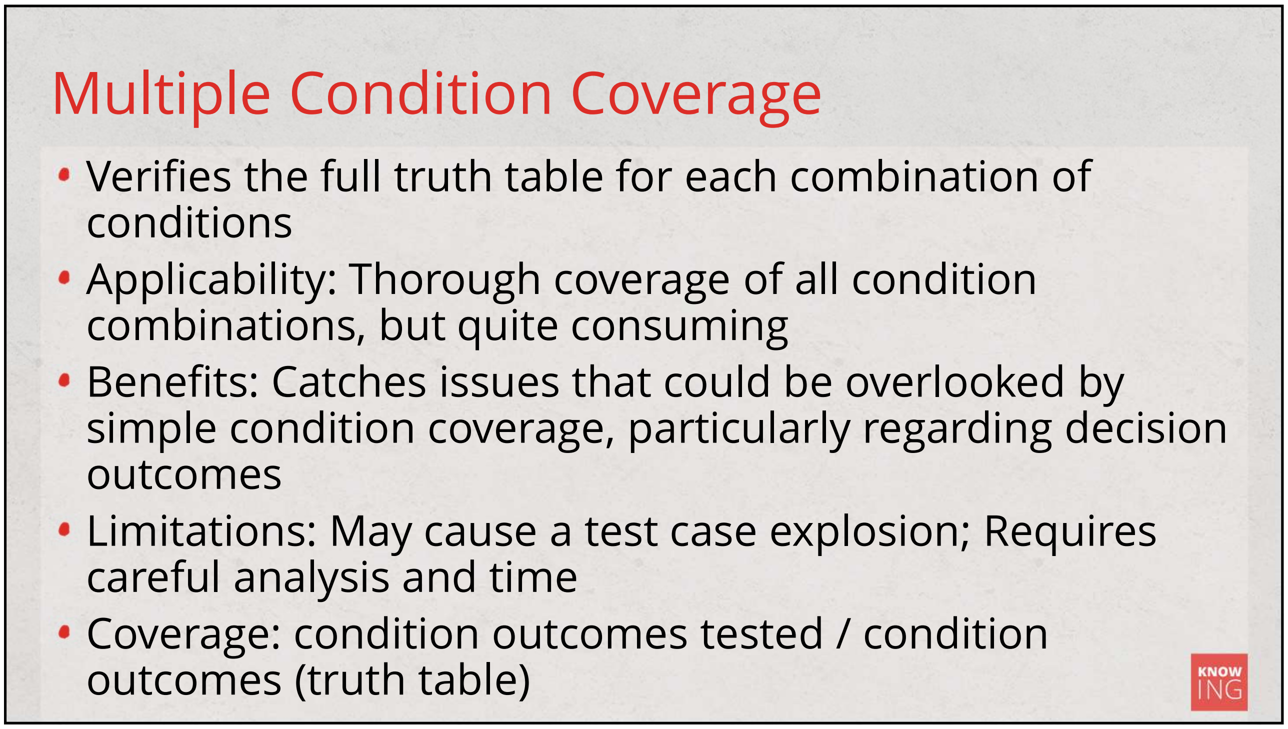
## Decision Condition Coverage

| Test Case | Value for a | Value for b | Value for c |
|---|---|---|---|
| Test 1 | 2 | ~~3~~ 0 | 2 |
| Test 2 | 1 | ~~0~~ 3 | 0 |

| Test Case | Condition 1 a > 1 | Condition 2 b = 0 | Decision 1 | Condition 3 a = 2 | Condition 4 c > 1 | Decision 2 |
|---|---|---|---|---|---|---|
| 1 | T | ~~F~~ T | ~~False~~ True | T | T | True |
| 2 | F | ~~T~~ F | False | F | F | False |

Condition coverage √
Decision coverage √

33

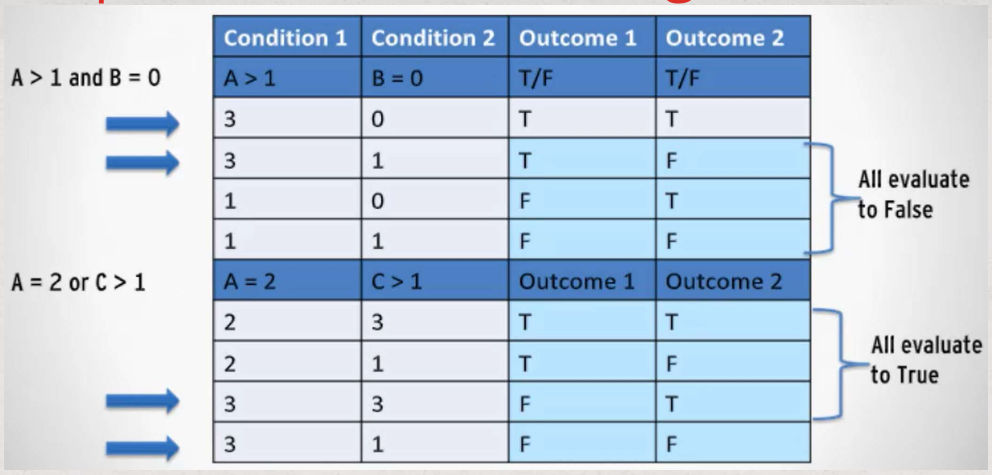


## Multiple Condition Coverage

- Verifies the full truth table for each combination of conditions
- Applicability: Thorough coverage of all condition combinations, but quite consuming
- Benefits: Catches issues that could be overlooked by simple condition coverage, particularly regarding decision outcomes
- Limitations: May cause a test case explosion; Requires careful analysis and time
- Coverage: condition outcomes tested / condition outcomes (truth table)

34

## Multiple Condition Coverage – Truth Table


A > 1 and B = 0
A = 2 or C > 1
All evaluate to False
All evaluate to True

| Condition 1 | Condition 2 | Outcome 1 | Outcome 2 |
|---|---|---|---|
| A > 1 | B = 0 | T/F | T/F |
| 3 | 0 | T | T |
| 3 | 1 | T | F |
| 1 | 0 | F | T |
| 1 | 1 | F | F |
| A = 2 | C > 1 | Outcome 1 | Outcome 2 |
| 2 | 3 | T | T |
| 2 | 1 | T | F |
| 3 | 3 | F | T |
| 3 | 1 | F | F |

- Full multiple condition coverage must have at least $2^n$ tests where n is the number of conditions
- Need 4 cases for each decision
- Realistically though, the arrowed cases could be combined

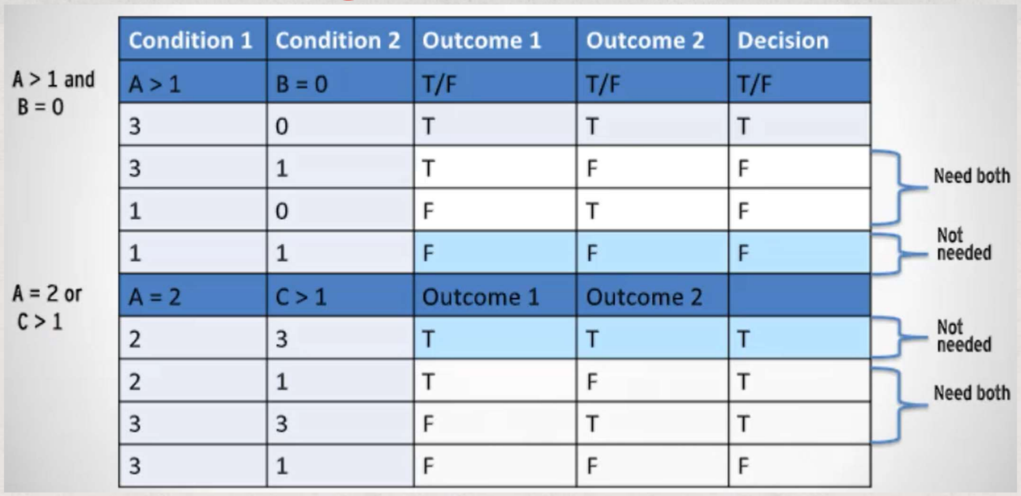
KNOW ING

## Multiple Condition/Decision Coverage

- MC/DC considers only condition combinations where each conditions has an impact on the decision outcome
- Applicability: Provides thorough coverage without excessive tests; Good for safety critical applications
- Benefits: Reduces the number of tests from multiple condition while still covering all conditions and decisions
- Limitations: Requires careful analysis and time; Coupling and short-circuiting can complicate testing
- Coverage: Conditions & outcomes / total conditions & outcomes

KNOW ING

## MC/DC Coverage – Truth Table


A > 1 and B = 0
A = 2 or C > 1
Need both
Not needed
Not needed
Need both

| Condition 1 | Condition 2 | Outcome 1 | Outcome 2 | Decision |
|---|---|---|---|---|
| A > 1 | B = 0 | T/F | T/F | T/F |
| 3 | 0 | T | T | T |
| 3 | 1 | T | F | F |
| 1 | 0 | F | T | F |
| 1 | 1 | F | F | F |
| A = 2 | C > 1 | Outcome 1 | Outcome 2 | |
| 2 | 3 | T | T | T |
| 2 | 1 | T | F | T |
| 3 | 3 | F | T | T |
| 3 | 1 | F | F | F |

- Full MD/DC coverage usually requires n + 1 tests, where n is the number of conditions
- In both these cases, 3 tests are needed

## Coupling

- If a term occurs more than once in an expression, it may be impossible to vary the value to change the decision

if (A and B) or (A and C) then …

- In this case, A “couples” the conditions in such a way that if A changes, both conditions are affected

## Short-Circuiting

- Some programming languages take “short cuts”

if (A or B) then …

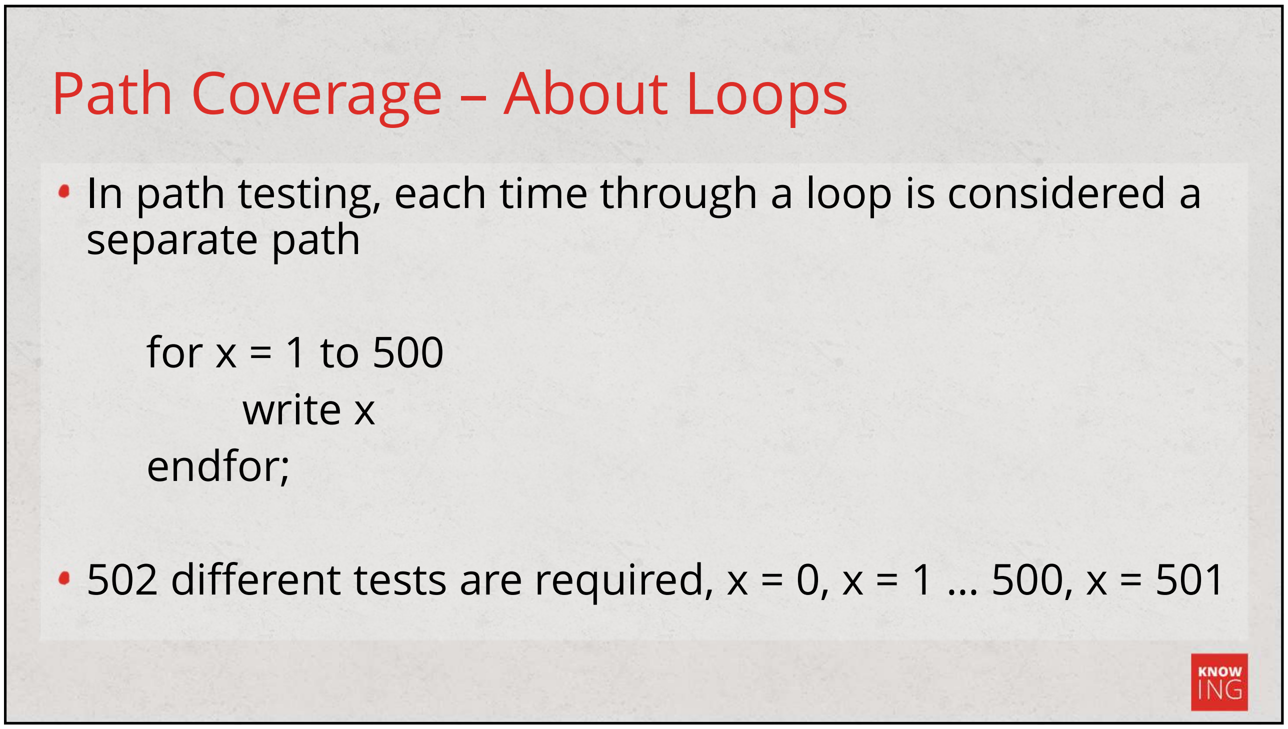
- In this case, B’s value doesn’t matter if A is false because A’s values will determine the outcome.

39

## Path Coverage

- Ideally tests every unique path through the code
- Applicability: Partial path testing is often used for safety critical software; Good for attaining high levels of coverage
- Benefits: May find issues missed in decision testing
- Limitations: Tools are usually required to calculate the paths in complex modules
- Coverage: If all paths, disregarding loops, are covered, statement and branch coverage is achieve and better coverage is provided

40



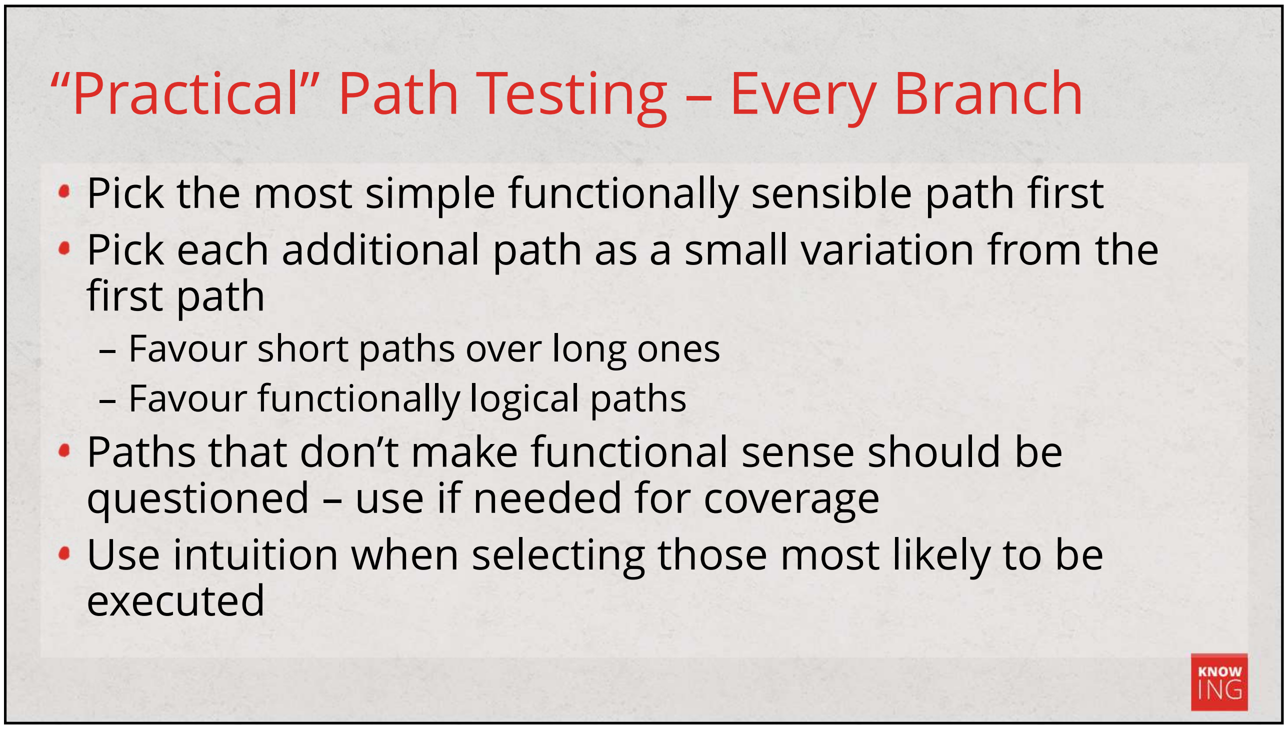
## Path Coverage – About Loops

- In path testing, each time through a loop is considered a separate path

```
for x = 1 to 500
        write x
endfor;
```

- 502 different tests are required, x = 0, x = 1 … 500, x = 501

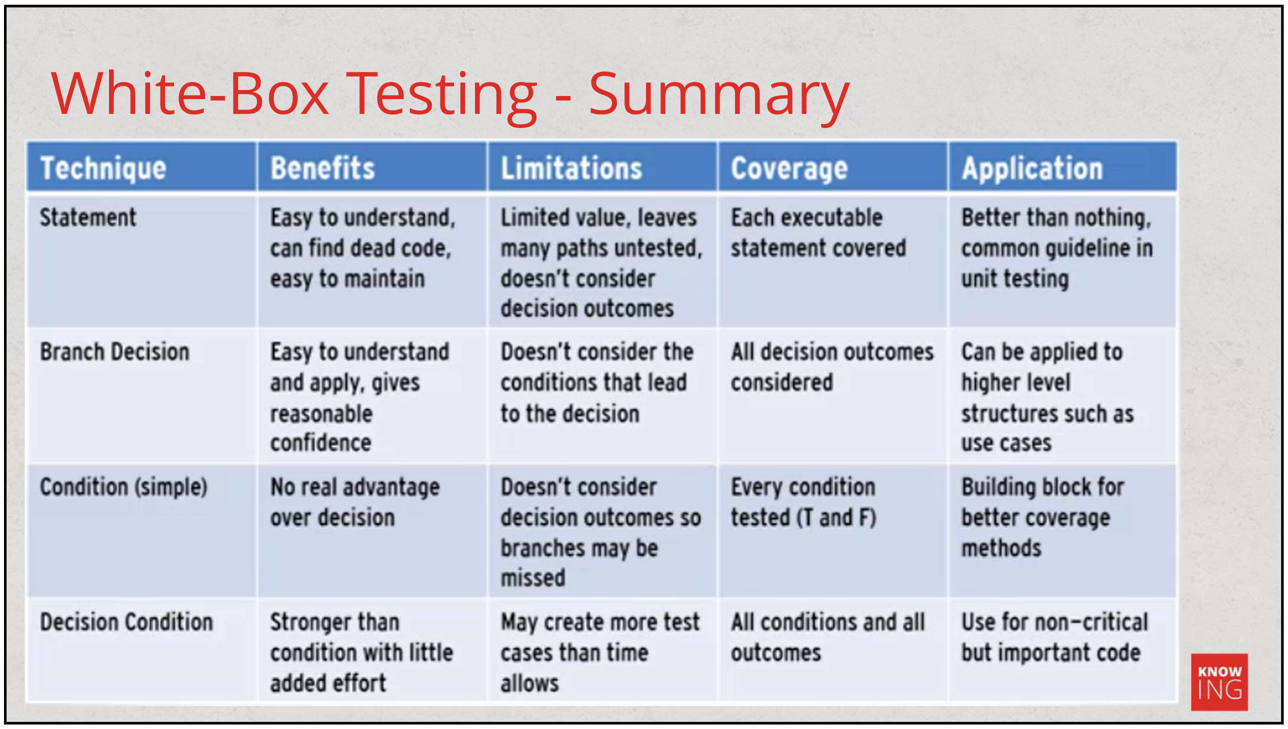
41



## "Practical" Path Testing – Every Branch

- Pick the most simple functionally sensible path first
- Pick each additional path as a small variation from the first path
  - Favour short paths over long ones
  - Favour functionally logical paths
- Paths that don't make functional sense should be questioned – use if needed for coverage
- Use intuition when selecting those most likely to be executed

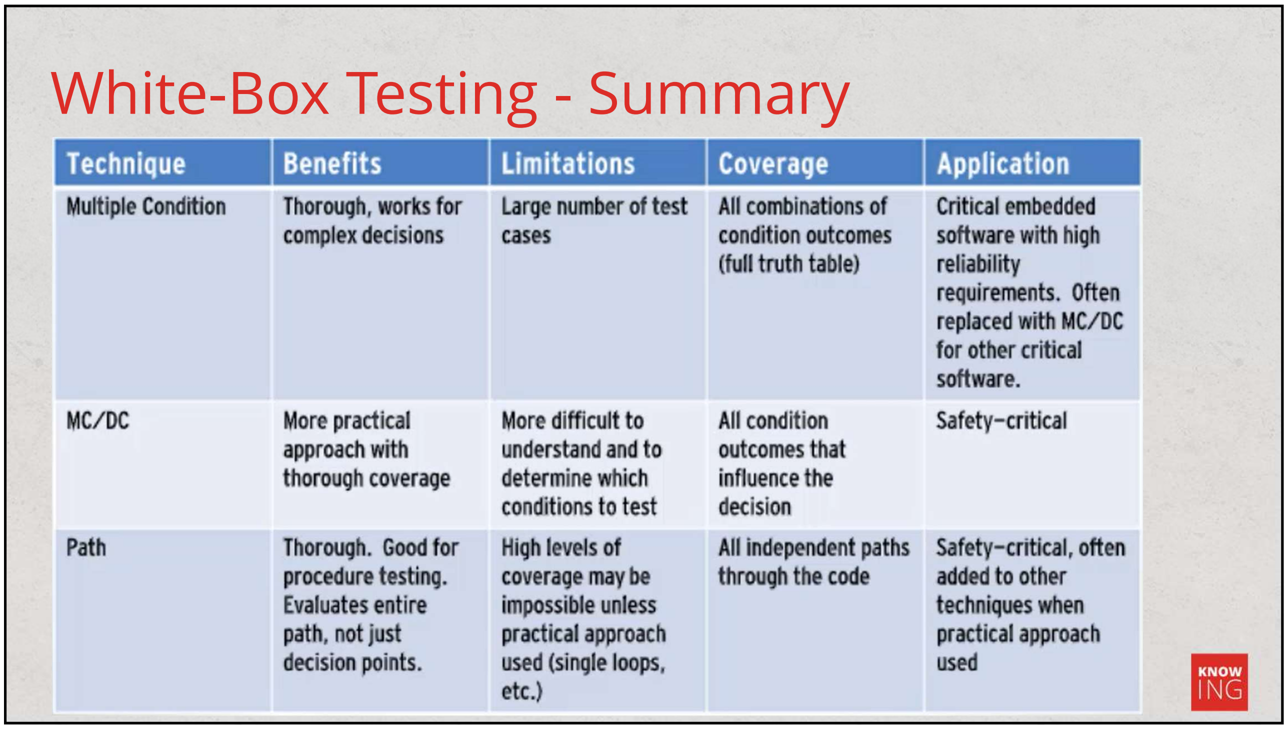
42



## White-Box Testing - Summary

| Technique | Benefits | Limitations | Coverage | Application |
|---|---|---|---|---|
| Statement | Easy to understand, can find dead code, easy to maintain | Limited value, leaves many paths untested, doesn't consider decision outcomes | Each executable statement covered | Better than nothing, common guideline in unit testing |
| Branch Decision | Easy to understand and apply, gives reasonable confidence | Doesn't consider the conditions that lead to the decision | All decision outcomes considered | Can be applied to higher level structures such as use cases |
| Condition (simple) | No real advantage over decision | Doesn't consider decision outcomes so branches may be missed | Every condition tested (T and F) | Building block for better coverage methods |
| Decision Condition | Stronger than condition with little added effort | May create more test cases than time allows | All conditions and all outcomes | Use for non-critical but important code |

KNOW ING

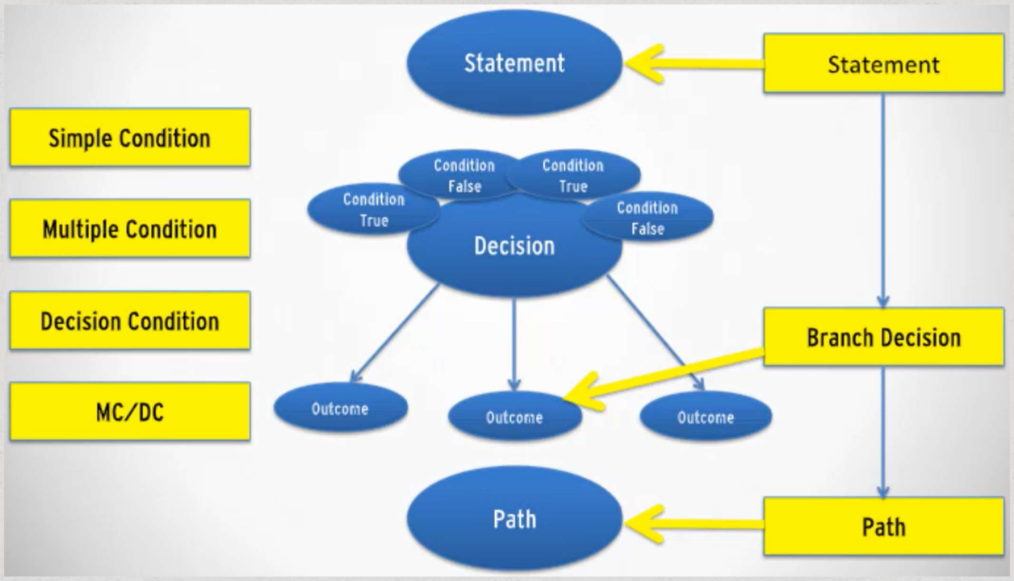


## White-Box Testing - Summary

| Technique | Benefits | Limitations | Coverage | Application |
|---|---|---|---|---|
| Multiple Condition | Thorough, works for complex decisions | Large number of test cases | All combinations of condition outcomes (full truth table) | Critical embedded software with high reliability requirements. Often replaced with MC/DC for other critical software. |
| MC/DC | More practical approach with thorough coverage | More difficult to understand and to determine which conditions to test | All condition outcomes that influence the decision | Safety-critical |
| Path | Thorough. Good for procedure testing. Evaluates entire path, not just decision points. | High levels of coverage may be impossible unless practical approach used (single loops, etc.) | All independent paths through the code | Safety-critical, often added to other techniques when practical approach used |

KNOW ING

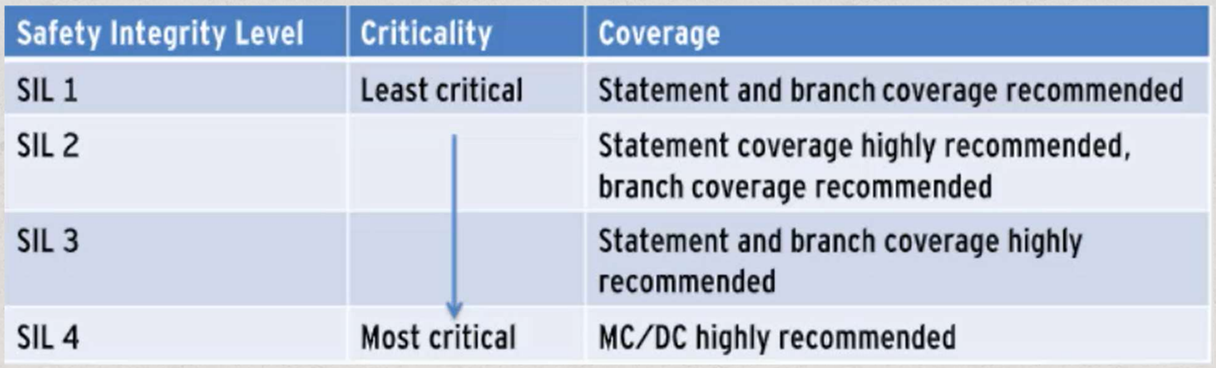
## White-Box Testing – Summary



## DO-178B (ED_12B) Requirements

| | | | |
|---|---|---|---|
| A | Catastrophic | Affects critical function needed to fly or land plane | MC/DC required |
| B | Hazardous | Large negative impact on safety or performance | Decision required, MC/DC optional |
| C | Major | Significant failure, less serious than A or B | Statement required |
| D | Minor | Failure noticeable, less impact than C | No specific requirements |
| E | No effect | Failure has no impact on safety | No specific requirement |

## IEC-61508



| Safety Integrity Level | Criticality | Coverage |
|---|---|---|
| SIL 1 | Least critical | Statement and branch coverage recommended |
| SIL 2 | | Statement coverage highly recommended, branch coverage recommended |
| SIL 3 | | Statement and branch coverage highly recommended |
| SIL 4 | Most critical | MC/DC highly recommended |

KNOW ING

47

## Experience-Based Testing

48

## Experience-Based Testing

- Experience of developers, testers and users to design, implement, and execute tests
- These techniques are often combined with black-box and white-box test techniques

KNOW ING

## Experience-Based Testing

- Formal Definition:

"Procedure to derive and/or select test cases based on the tester's experience, knowledge and intuition."

KNOW ING

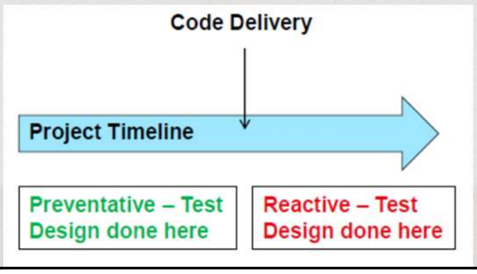
## Experience-Based Testing

- Utilise the tester's skill and intuition as well as their experience with similar applications and technologies
- Suitable for finding defects
- May be inappropriate in systems requiring detailed test documentation
- Less suitable for producing reusable test procedures
- Not appropriate for achieving specific test coverage levels.

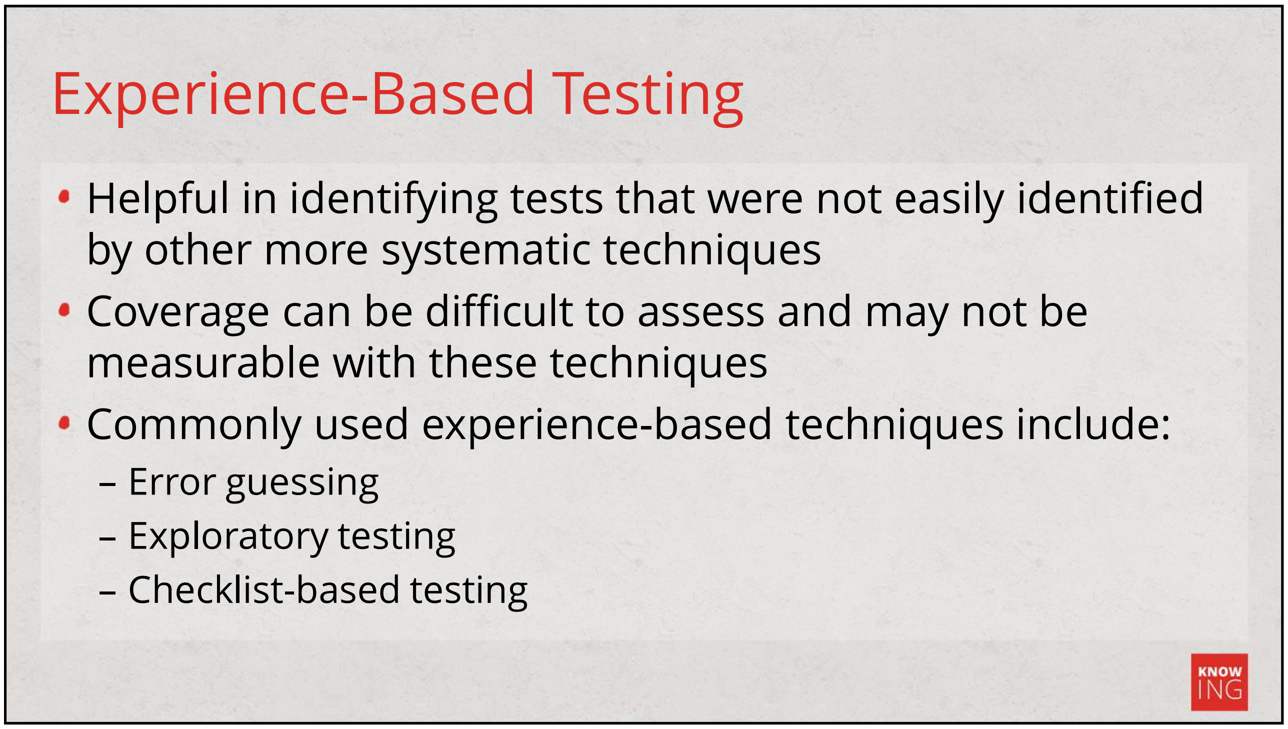
51

## Experience-Based Testing

- Experience-based testing reflects a dynamic and heuristic approach where testing is reactive, not preventative
- Test execution and evaluation are concurrent tasks
- For some experienced-based tests, a more structured approach can be taken so less dynamic


Code Delivery
Project Timeline
Preventative – Test Design done here
Reactive – Test Design done here

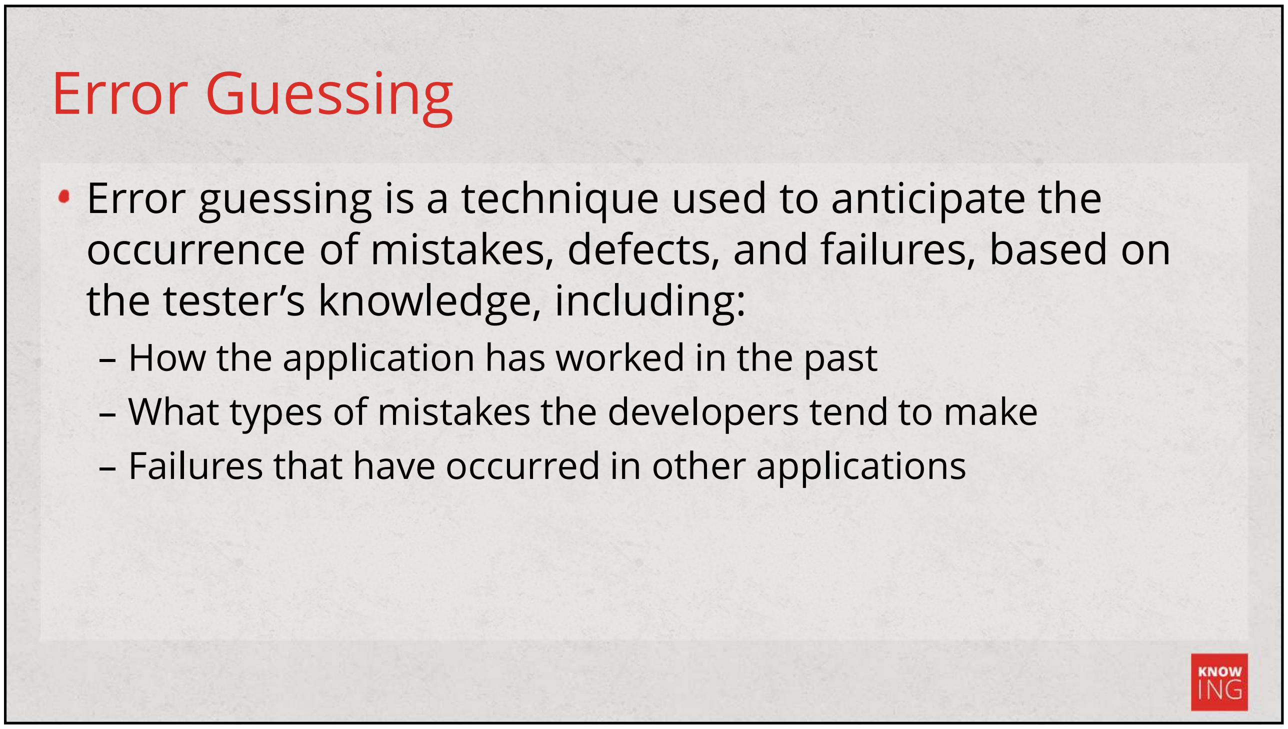
52



## Experience-Based Testing

- Helpful in identifying tests that were not easily identified by other more systematic techniques
- Coverage can be difficult to assess and may not be measurable with these techniques
- Commonly used experience-based techniques include:
  - Error guessing
  - Exploratory testing
  - Checklist-based testing

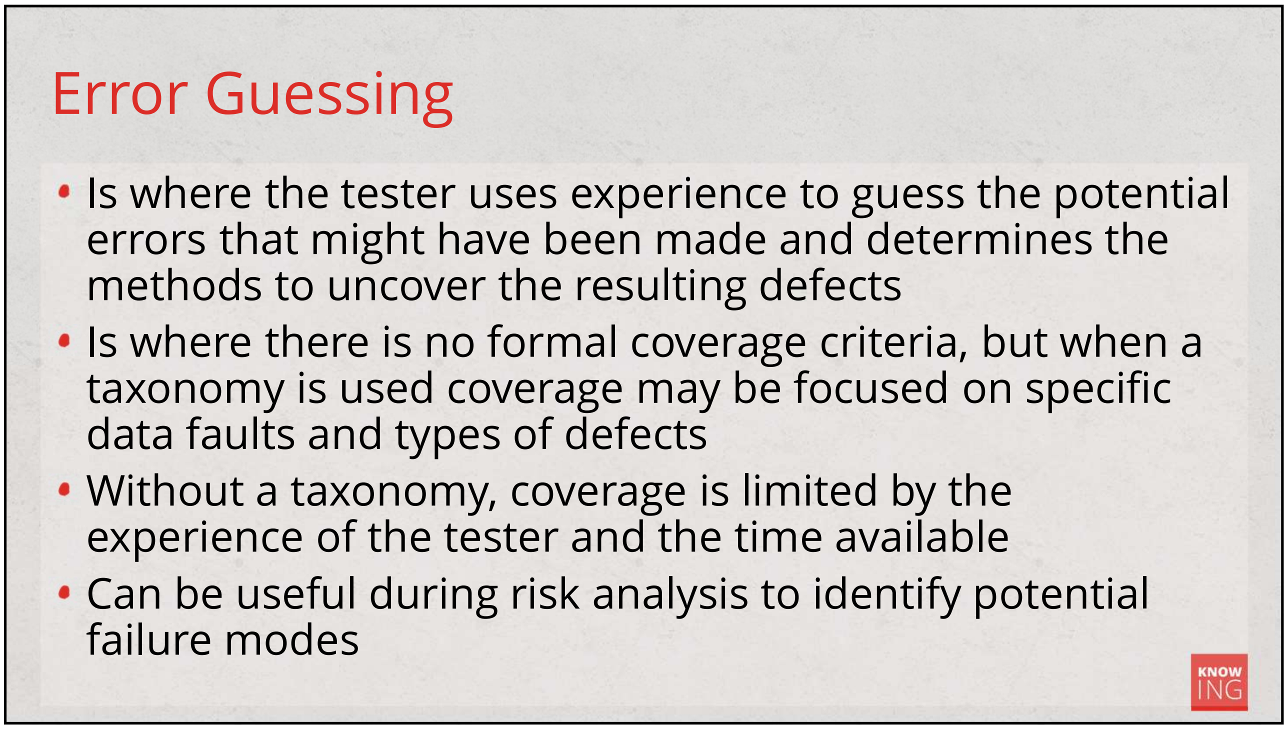
53



## Error Guessing

- Error guessing is a technique used to anticipate the occurrence of mistakes, defects, and failures, based on the tester’s knowledge, including:
  - How the application has worked in the past
  - What types of mistakes the developers tend to make
  - Failures that have occurred in other applications

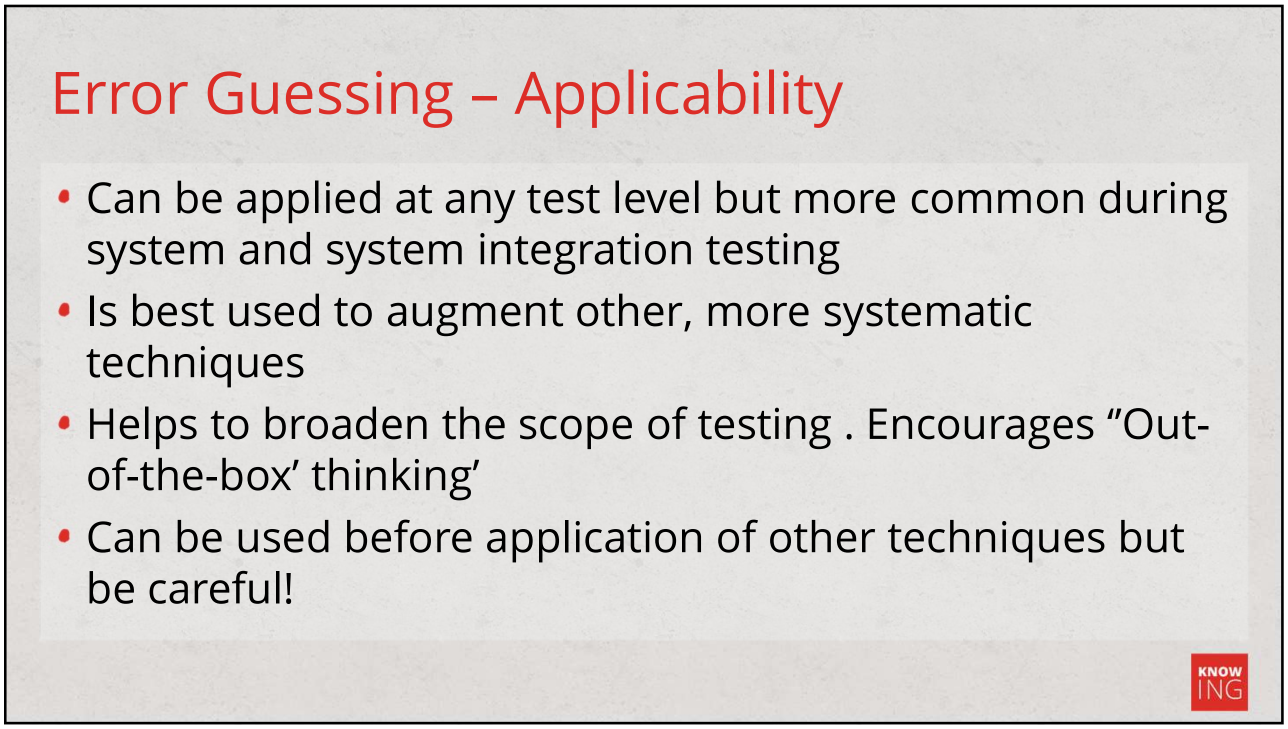
54



## Error Guessing

- Is where the tester uses experience to guess the potential errors that might have been made and determines the methods to uncover the resulting defects
- Is where there is no formal coverage criteria, but when a taxonomy is used coverage may be focused on specific data faults and types of defects
- Without a taxonomy, coverage is limited by the experience of the tester and the time available
- Can be useful during risk analysis to identify potential failure modes

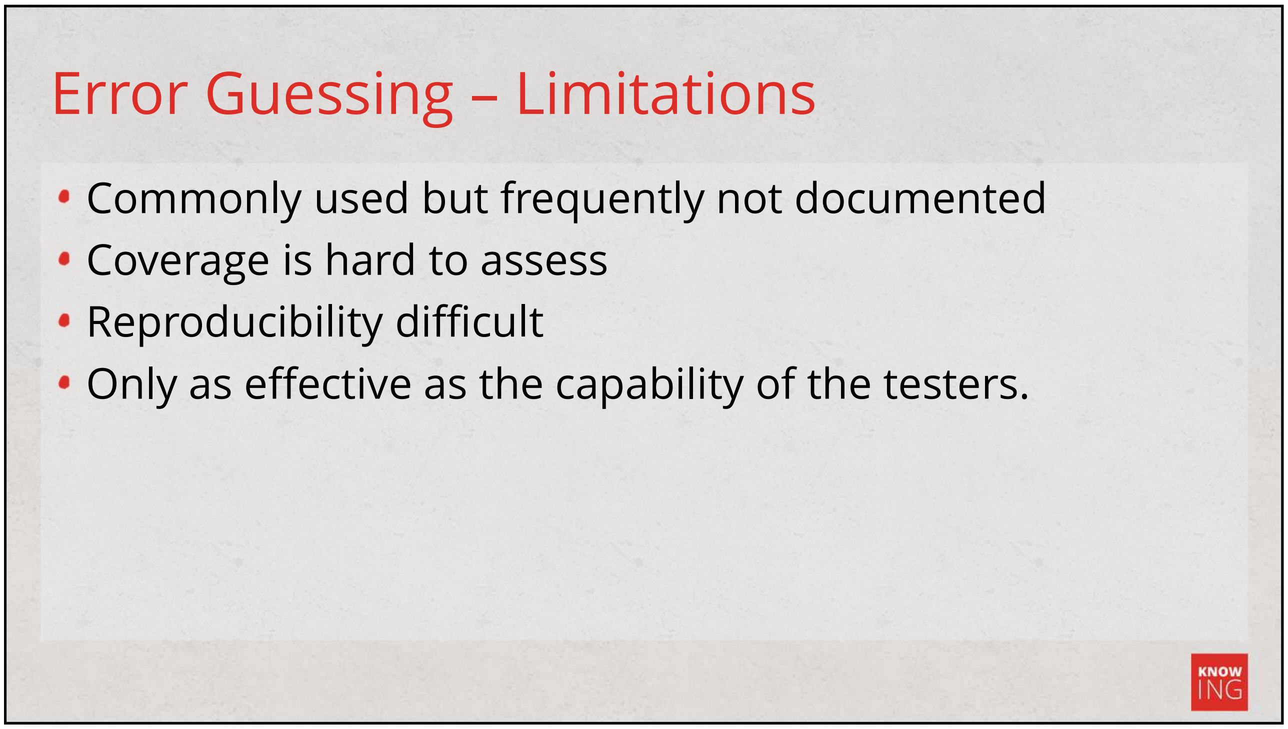
55



## Error Guessing – Applicability

- Can be applied at any test level but more common during system and system integration testing
- Is best used to augment other, more systematic techniques
- Helps to broaden the scope of testing . Encourages ”Out-of-the-box’ thinking’
- Can be used before application of other techniques but be careful!

56

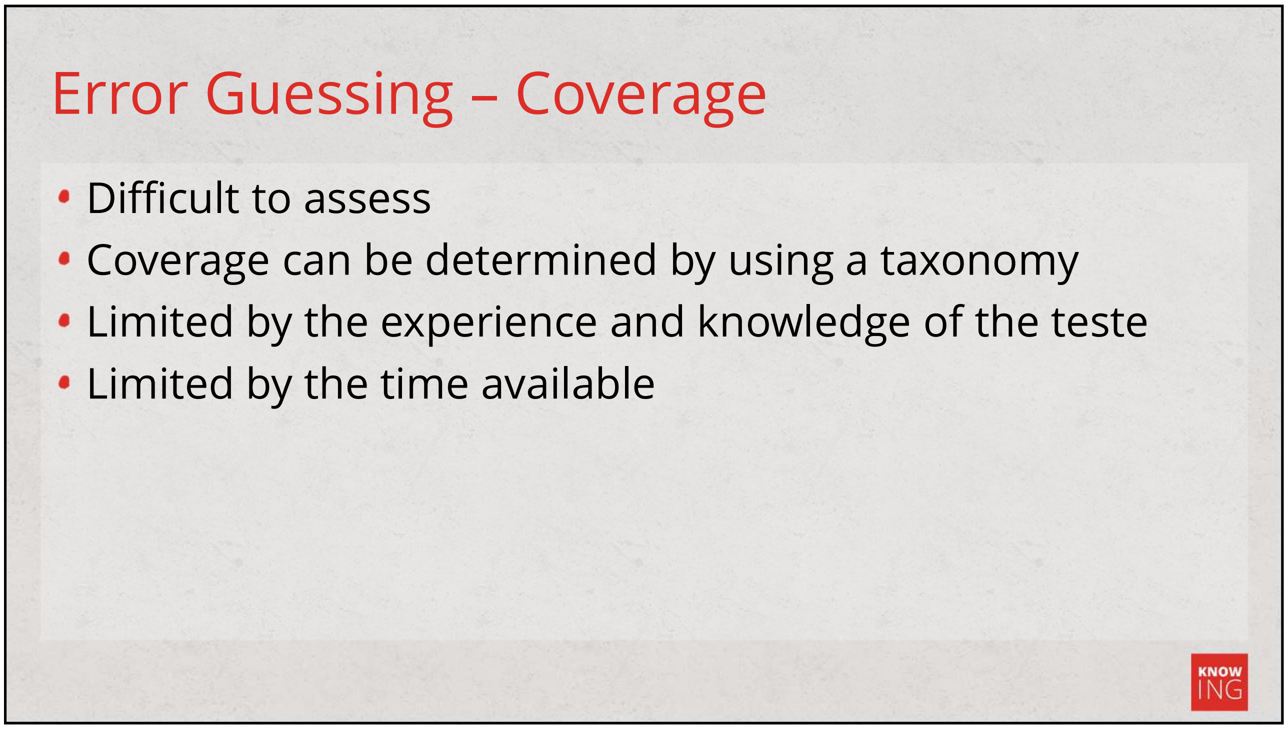


## Error Guessing – Limitations

- Commonly used but frequently not documented
- Coverage is hard to assess
- Reproducibility difficult
- Only as effective as the capability of the testers.

KNOW ING

57

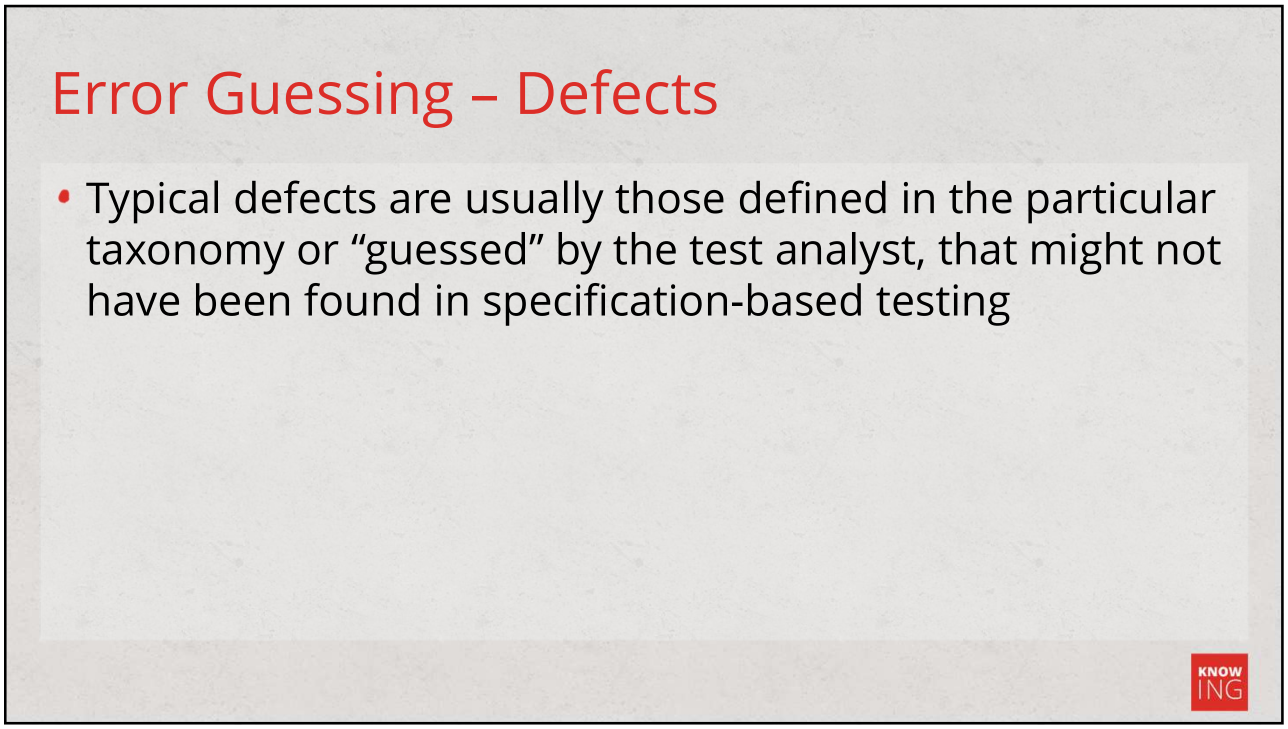


## Error Guessing – Coverage

- Difficult to assess
- Coverage can be determined by using a taxonomy
- Limited by the experience and knowledge of the teste
- Limited by the time available

KNOW ING

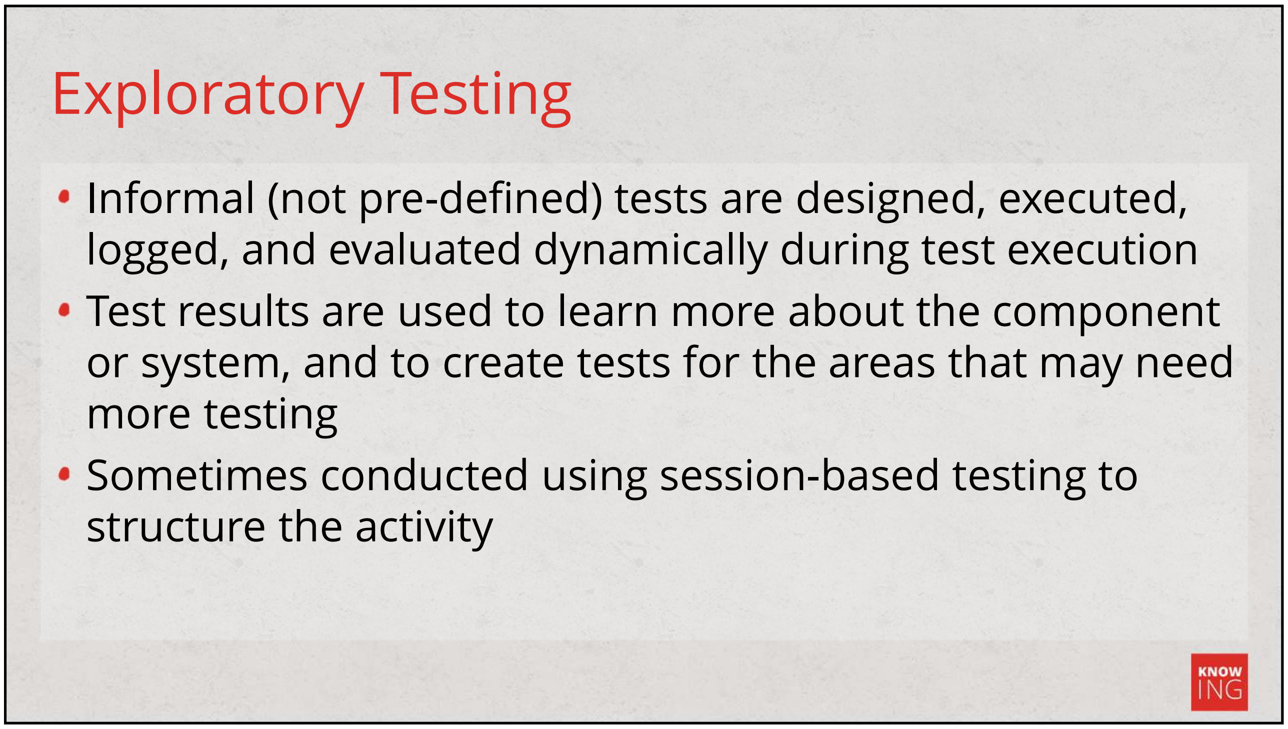
58



## Error Guessing – Defects

- Typical defects are usually those defined in the particular taxonomy or “guessed” by the test analyst, that might not have been found in specification-based testing

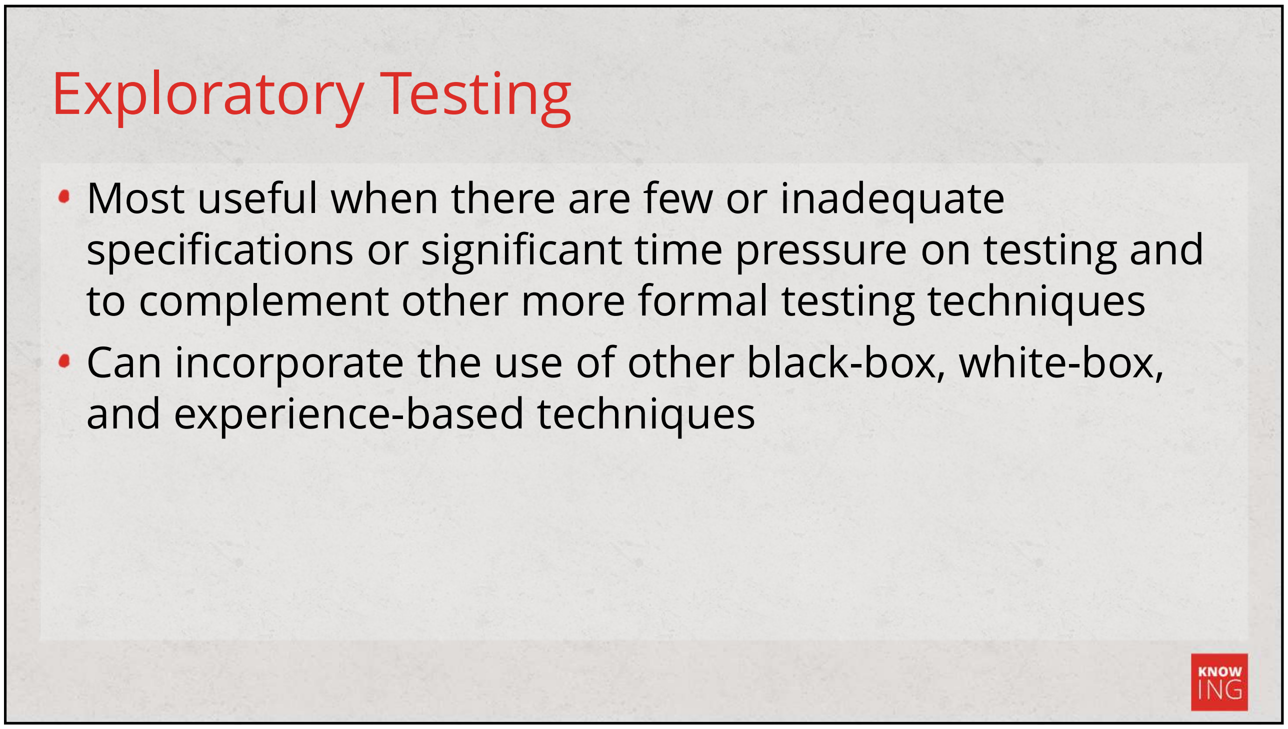
59



## Exploratory Testing

- Informal (not pre-defined) tests are designed, executed, logged, and evaluated dynamically during test execution
- Test results are used to learn more about the component or system, and to create tests for the areas that may need more testing
- Sometimes conducted using session-based testing to structure the activity

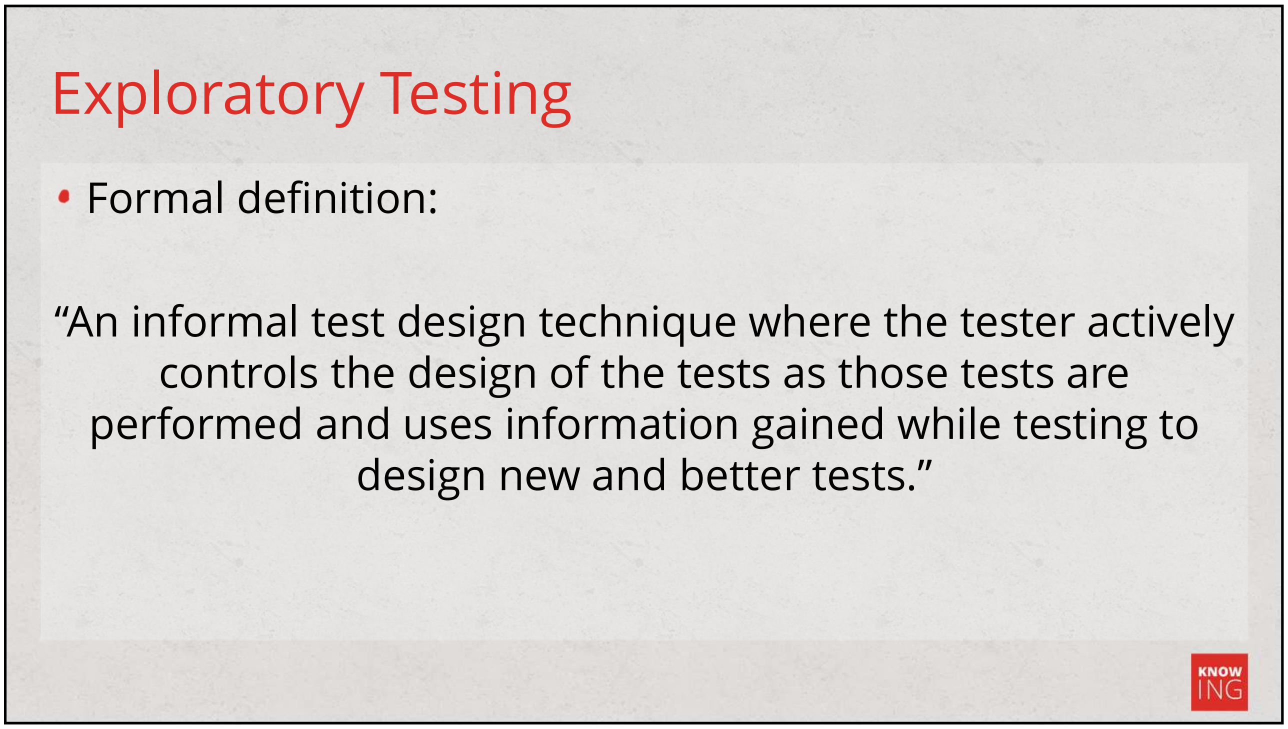
60



## Exploratory Testing

- Most useful when there are few or inadequate specifications or significant time pressure on testing and to complement other more formal testing techniques
- Can incorporate the use of other black-box, white-box, and experience-based techniques

61



## Exploratory Testing

- Formal definition:

"An informal test design technique where the tester actively controls the design of the tests as those tests are performed and uses information gained while testing to design new and better tests."

62

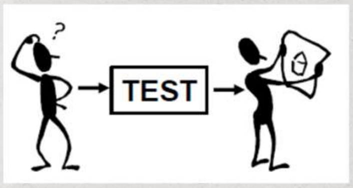
## Exploratory Testing

- The tester simultaneously:
  - Learns about the product and its defects
  - Plans the testing work to be done
  - Designs and executes the tests
  - Reports the results;
- There is no formal coverage criteria, but charters can be used to specify tasks, objectives and deliverables:
  - The tester dynamically adjusts these goals during execution.

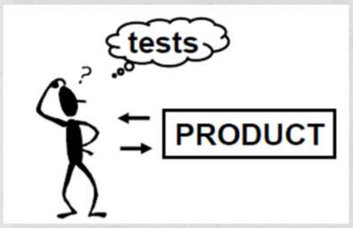
63

## Exploratory Testing vs. Traditional Testing

- Based on “traditional” techniques, tests are first designed and then recorded. They are to be executed at some later time (by a different tester).



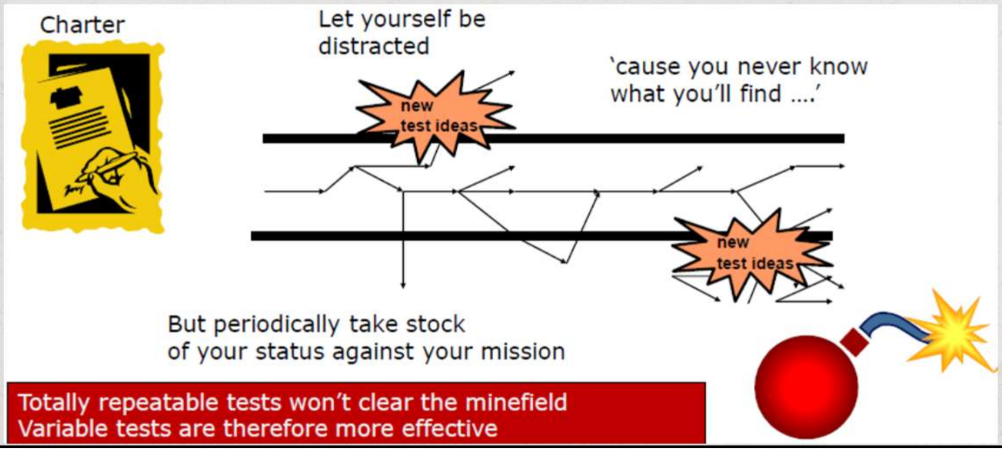
- In exploratory testing, tests are designed and executed at the same time, and they sometimes are not recorded.



64

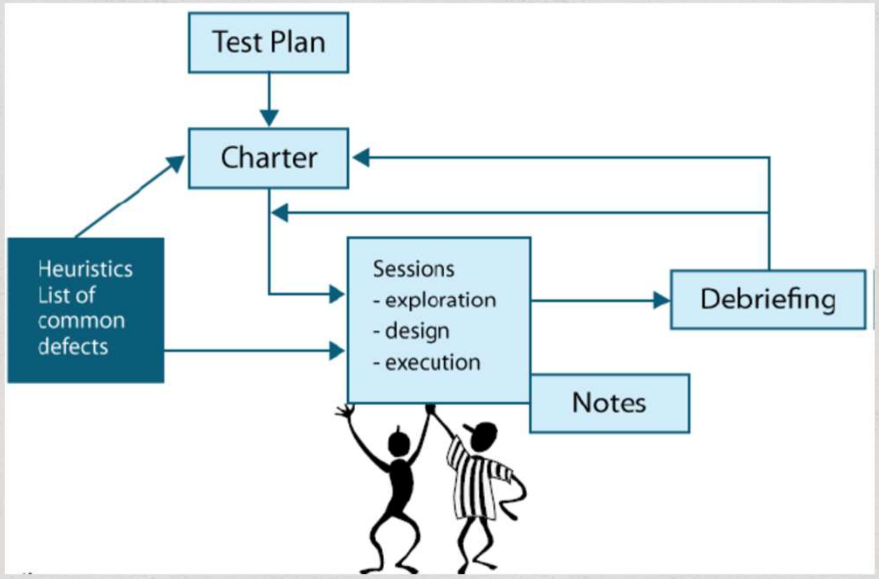
## Exploratory Testing

- A common goal of exploration is to *probe* for *weak* areas of the program


Charter
Let yourself be distracted
'cause you never know what you'll find ....'
new test ideas
new test ideas
But periodically take stock of your status against your mission

Totally repeatable tests won't clear the minefield
Variable tests are therefore more effective

## Exploratory Testing – Process

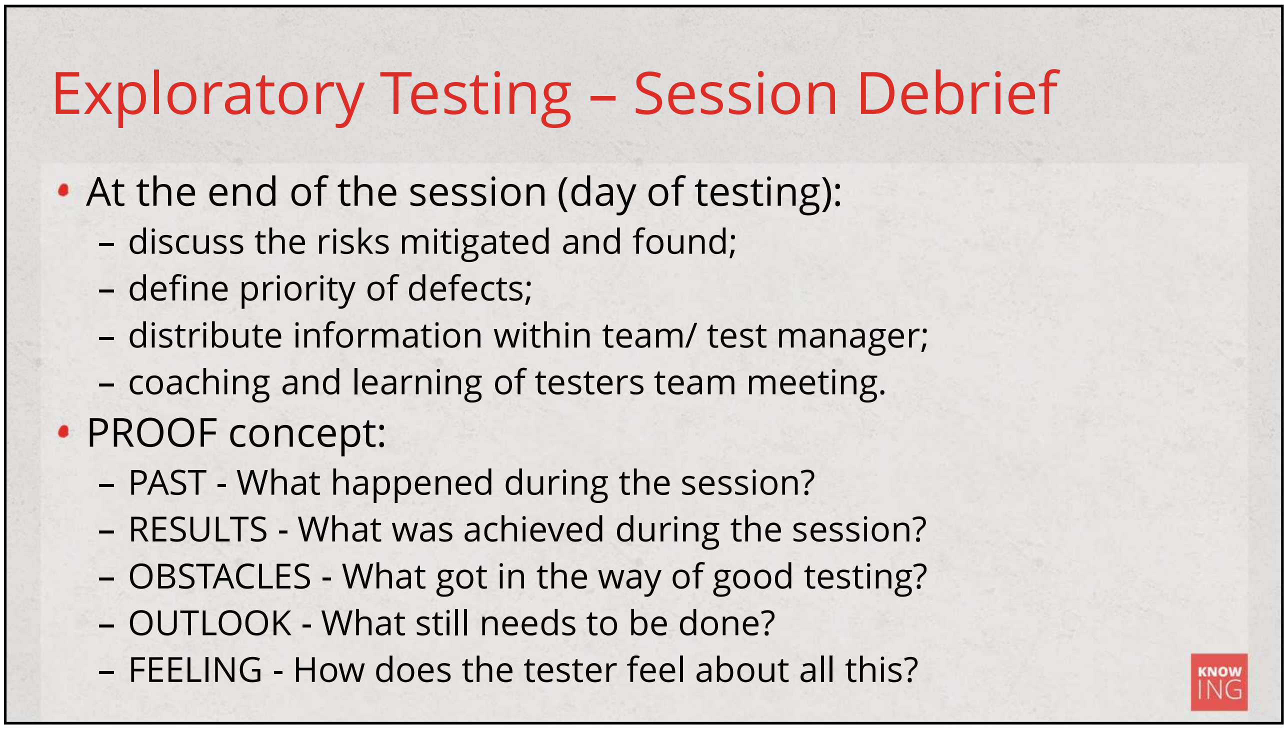


## Exploratory Testing – Test Sessions

- Session-based test management (SBTM) is a concept for managing exploratory testing.
- A test session can be separated into three stages:
  1. Session Setup: Setting up the test environment and improving the understanding of the product.
  2. Test Design and Execution: Scanning the test object and looking for problems.
  3. Defect Investigation & Reporting: Begins when the tester finds something that looks to be a failure.

## Exploratory Testing – Session Notes & Sheets

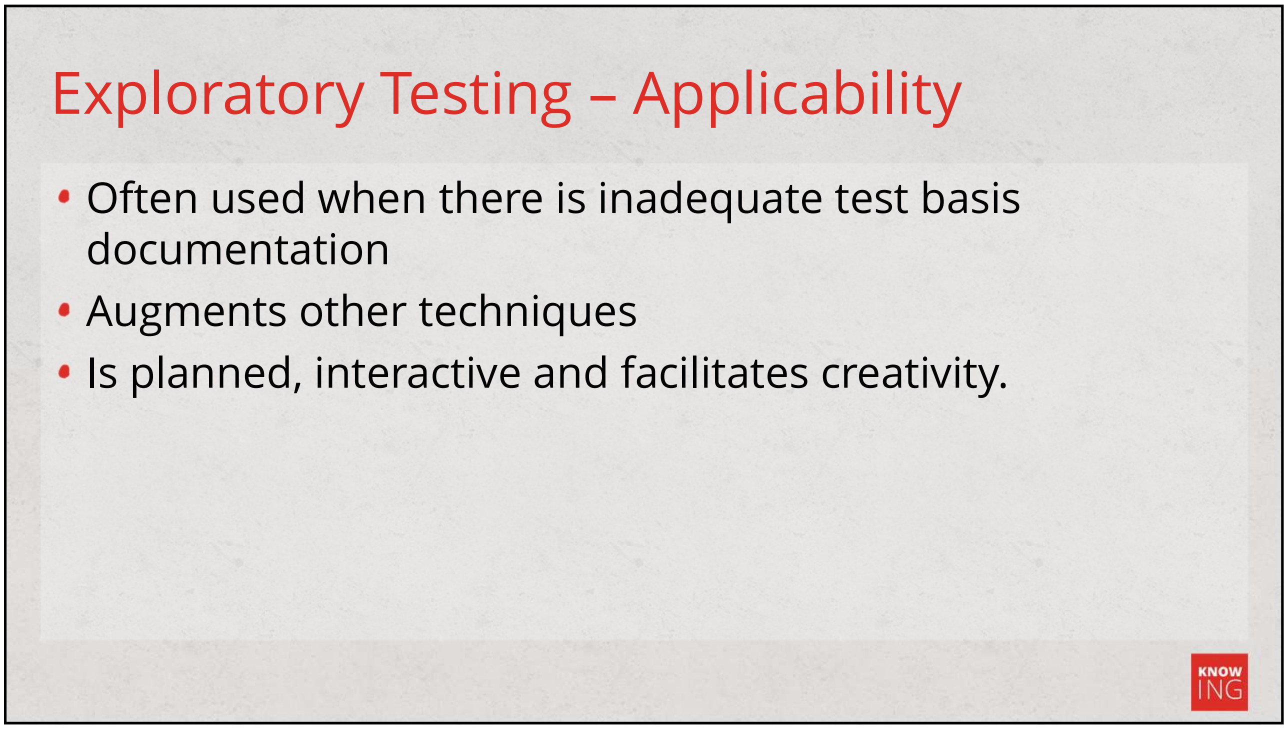
- The SBTM session sheet consists of the following:
  – Session charter;
  – Tester name(s);
  – Date and time started;
  – Task breakdown (sessions);
  – Data files;
  – Test notes;
  – Issues;
  – Defect.
- At the end of each session Debrief meeting!



## Exploratory Testing – Session Debrief

- At the end of the session (day of testing):
  - discuss the risks mitigated and found;
  - define priority of defects;
  - distribute information within team/ test manager;
  - coaching and learning of testers team meeting.
- PROOF concept:
  - PAST - What happened during the session?
  - RESULTS - What was achieved during the session?
  - OBSTACLES - What got in the way of good testing?
  - OUTLOOK - What still needs to be done?
  - FEELING - How does the tester feel about all this?

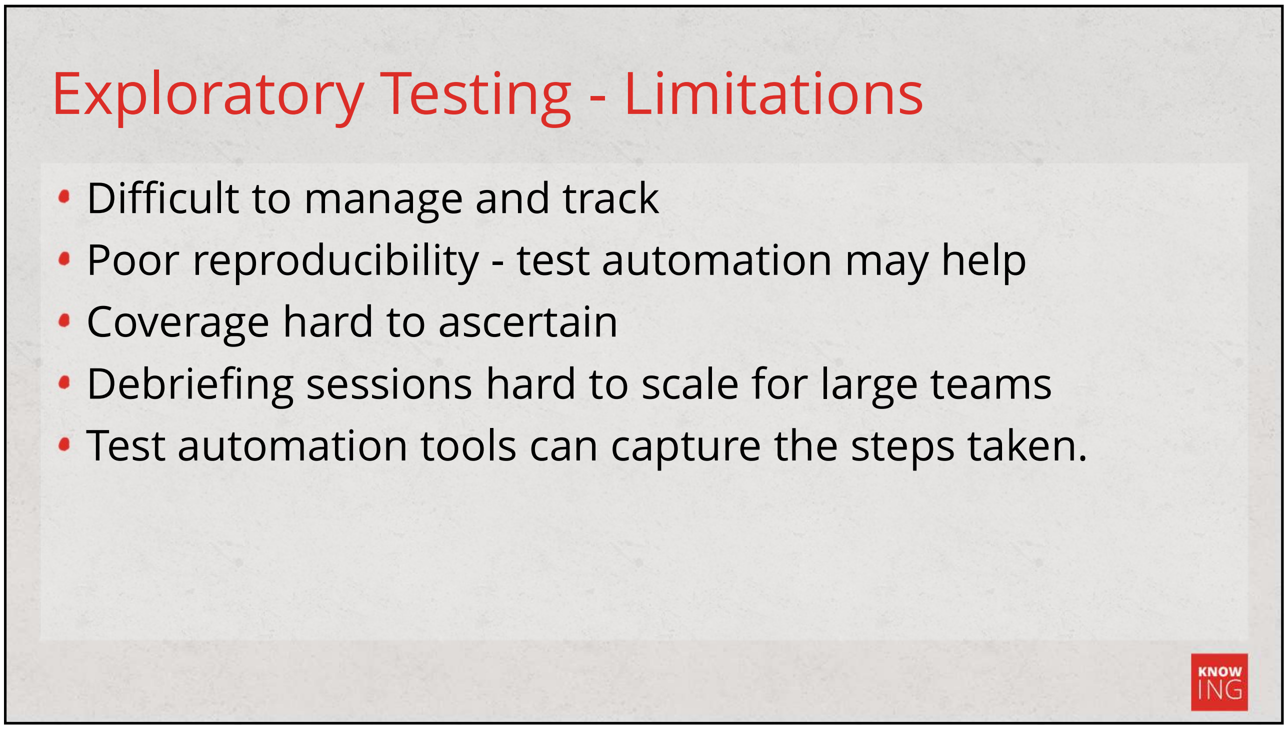
69



## Exploratory Testing – Applicability

- Often used when there is inadequate test basis documentation
- Augments other techniques
- Is planned, interactive and facilitates creativity.

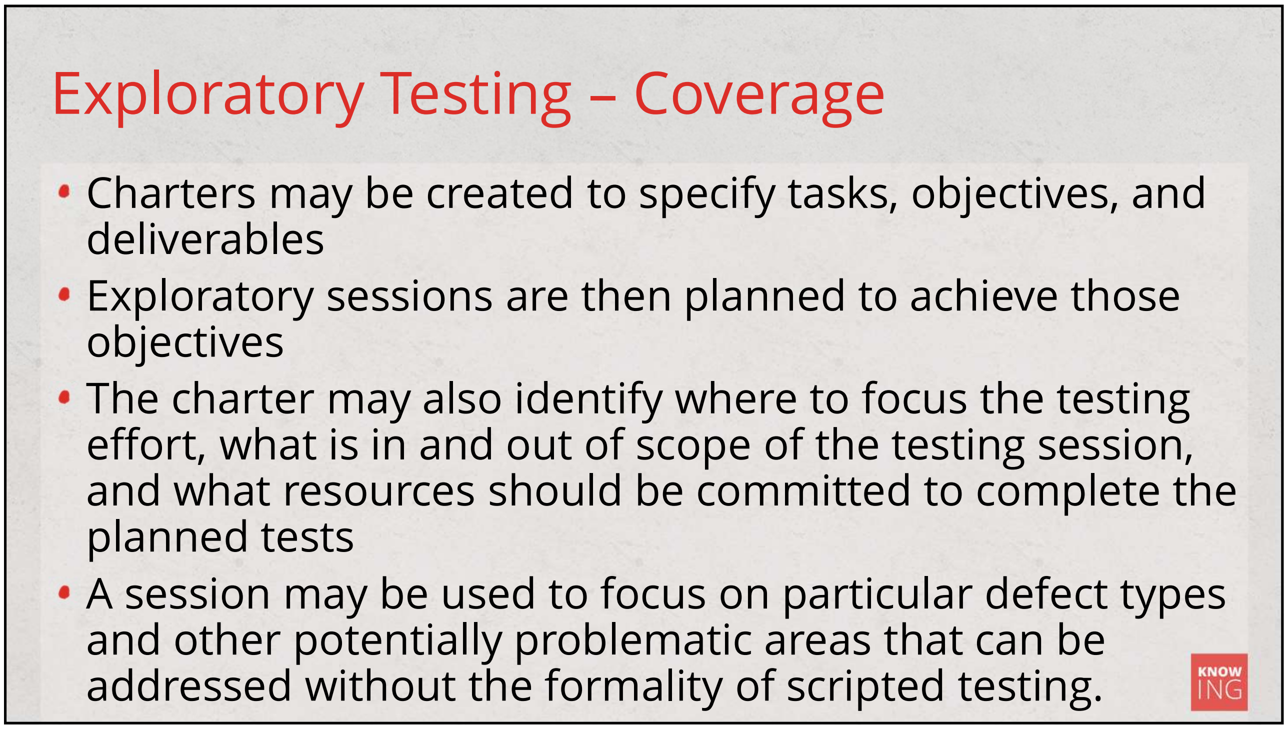
70



## Exploratory Testing - Limitations

- Difficult to manage and track
- Poor reproducibility - test automation may help
- Coverage hard to ascertain
- Debriefing sessions hard to scale for large teams
- Test automation tools can capture the steps taken.

71



## Exploratory Testing – Coverage

- Charters may be created to specify tasks, objectives, and deliverables
- Exploratory sessions are then planned to achieve those objectives
- The charter may also identify where to focus the testing effort, what is in and out of scope of the testing session, and what resources should be committed to complete the planned tests
- A session may be used to focus on particular defect types and other potentially problematic areas that can be addressed without the formality of scripted testing.

72

## Exploratory Testing – Defects

- Missed during functional testing
- Falling between functional boundaries
- Performance and security related .

73

## Exploratory Testing

- Exploratory testing is simultaneous learning, test design, and test execution
- It is:
  - An interactive test process where the information gained while testing to design new and better tests
  - Different from error-guessing and such as there is a "formal" process defined
  - Testers have the skills to listen, read, think and report rigorously and effectively
  - Exploratory Testing = Exploratory Test Management

74

## Checklist-Based Testing

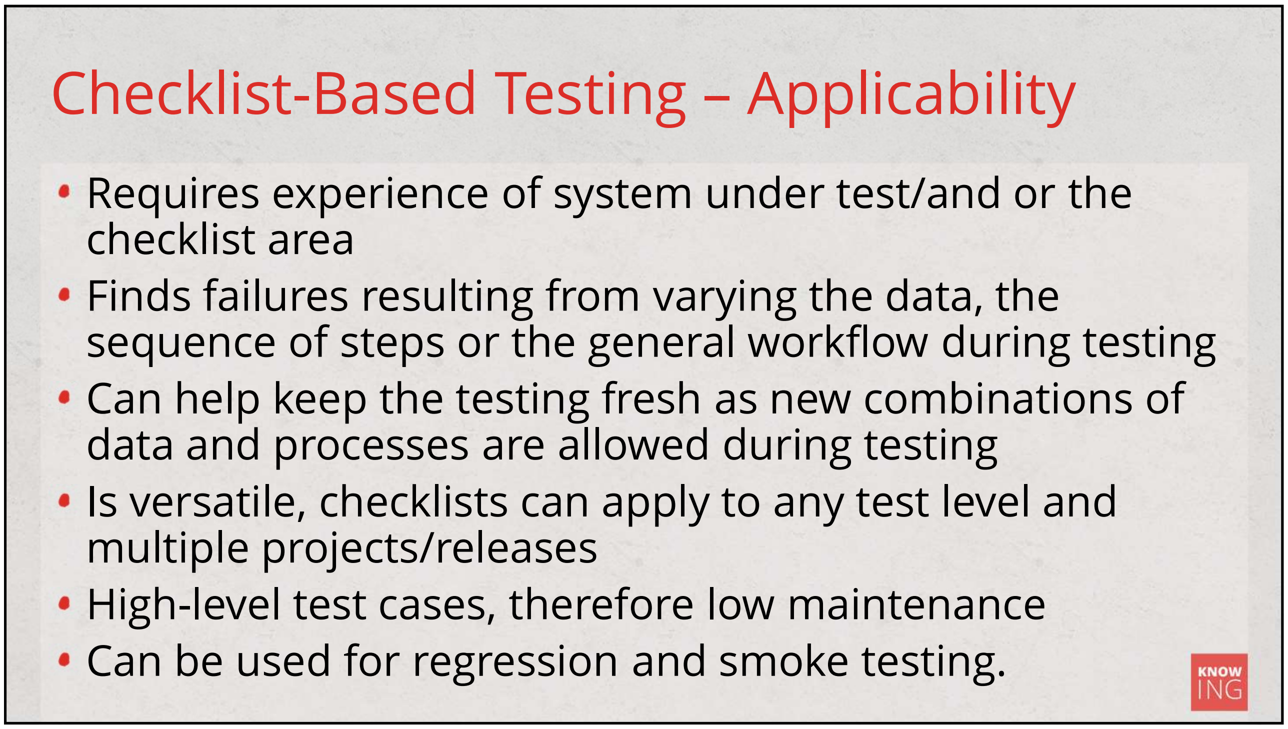
- Testers design, implement, and execute tests to cover test conditions found in a checklist
- Checklists can be created to support various test types, including functional and non-functional testing
- In the absence of detailed test cases, checklist-based testing can provide guidelines and a degree of consistency
- As these are high-level lists, some variability in the actual testing is likely to occur, resulting in potentially greater coverage but less repeatability

75

## Checklist-Based Testing

- Is where the experienced tester uses a high-level list of items to be noted, checked of remembered
- Alternately where a set of rules or criteria are created against which product has to be verified
- The checklists are built based on standards, experience and other considerations
- There is no formal coverage criteria, but testing may be focused to ensure that each item on the checklist has been covered.

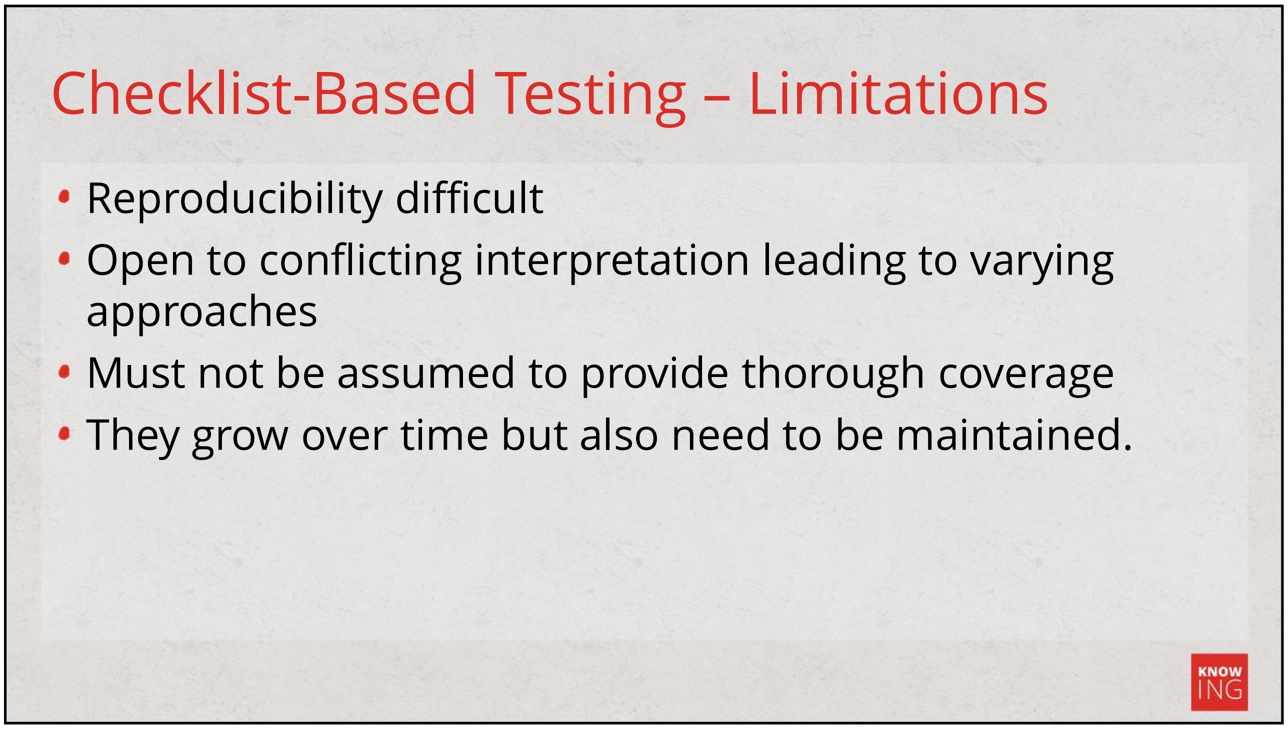
76



## Checklist-Based Testing – Applicability

- Requires experience of system under test/and or the checklist area
- Finds failures resulting from varying the data, the sequence of steps or the general workflow during testing
- Can help keep the testing fresh as new combinations of data and processes are allowed during testing
- Is versatile, checklists can apply to any test level and multiple projects/releases
- High-level test cases, therefore low maintenance
- Can be used for regression and smoke testing.

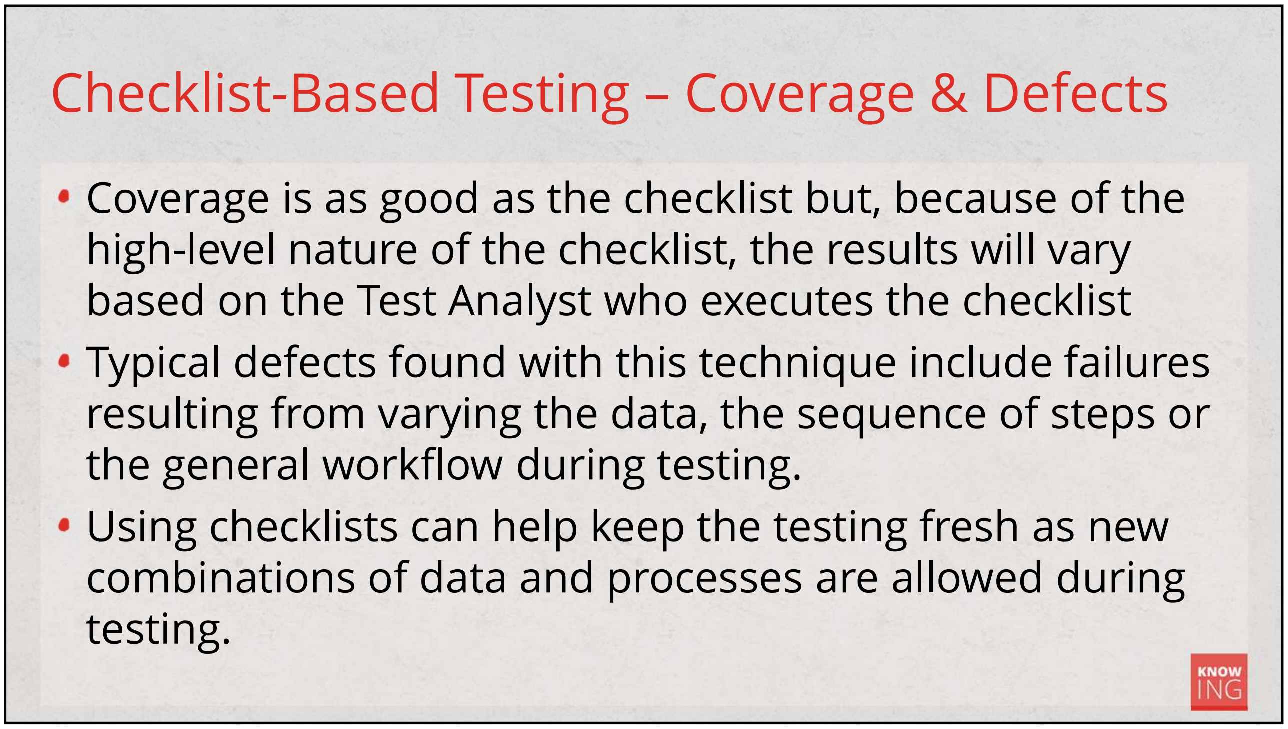
77



## Checklist-Based Testing – Limitations

- Reproducibility difficult
- Open to conflicting interpretation leading to varying approaches
- Must not be assumed to provide thorough coverage
- They grow over time but also need to be maintained.

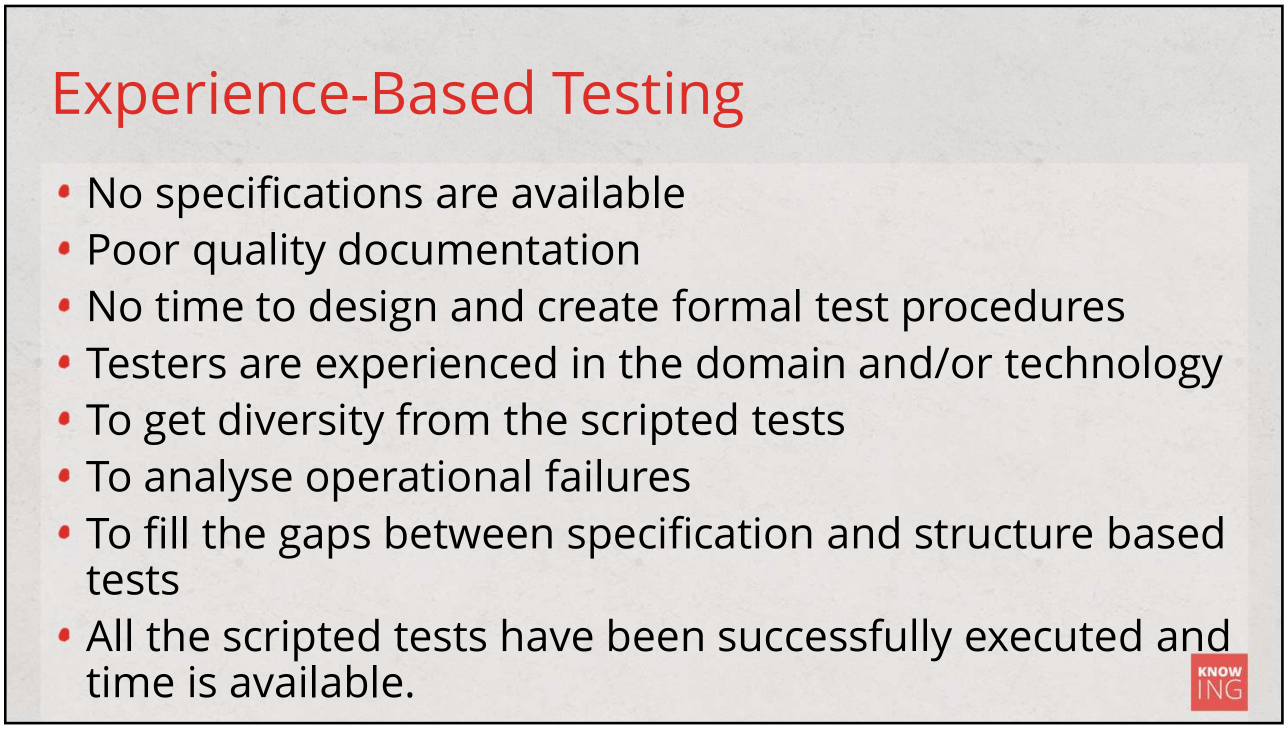
78



## Checklist-Based Testing – Coverage & Defects

- Coverage is as good as the checklist but, because of the high-level nature of the checklist, the results will vary based on the Test Analyst who executes the checklist
- Typical defects found with this technique include failures resulting from varying the data, the sequence of steps or the general workflow during testing.
- Using checklists can help keep the testing fresh as new combinations of data and processes are allowed during testing.

79



## Experience-Based Testing

- No specifications are available
- Poor quality documentation
- No time to design and create formal test procedures
- Testers are experienced in the domain and/or technology
- To get diversity from the scripted tests
- To analyse operational failures
- To fill the gaps between specification and structure based tests
- All the scripted tests have been successfully executed and time is available.

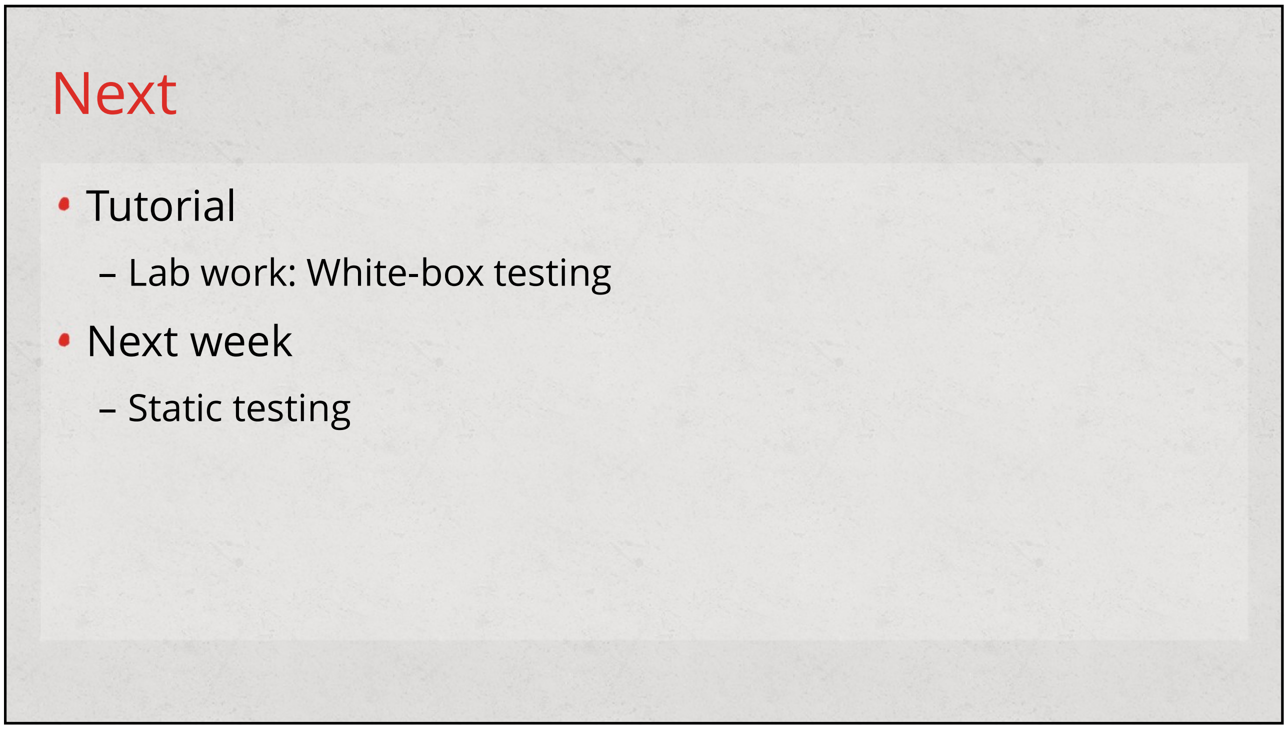
80

## Summary

- Statement, decision
- Condition, MC/DC
- Experience-based testing

## References

- Sommerville, I. *Software Engineering* (Chapter 8)
- Bruegge, B. & Dutoit A.H. *Object-Oriented Software Engineering Using UML, Patterns, and Java* (Chapter 11)
- International Software Testing Qualifications Board. *ISTQB Foundation Level (Core) Syllabus*



## Next

- Tutorial
  - Lab work: White-box testing
- Next week
  - Static testing