Software Testing

Dr. RAJIB MALL

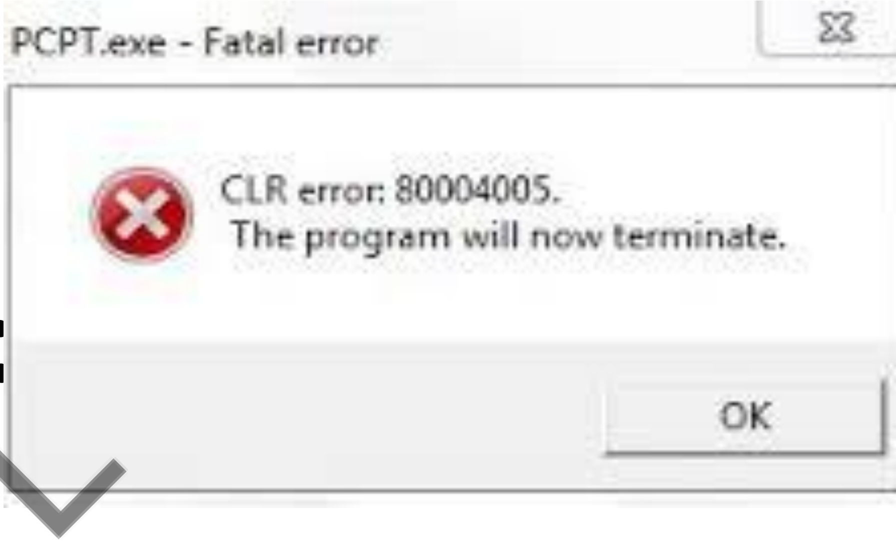
Professor

Department Of Computer Science & Engineering

IIT Kharagpur.

Faults and Failures

- A program may fail during testing:



PCPT.exe - Fatal error

CLR error: 80004005.
The program will now terminate.

  - ▪ A manifestation of a fault (also called defect or bug).
  - ▪ **Mere presence of a fault may not lead to a failure.**

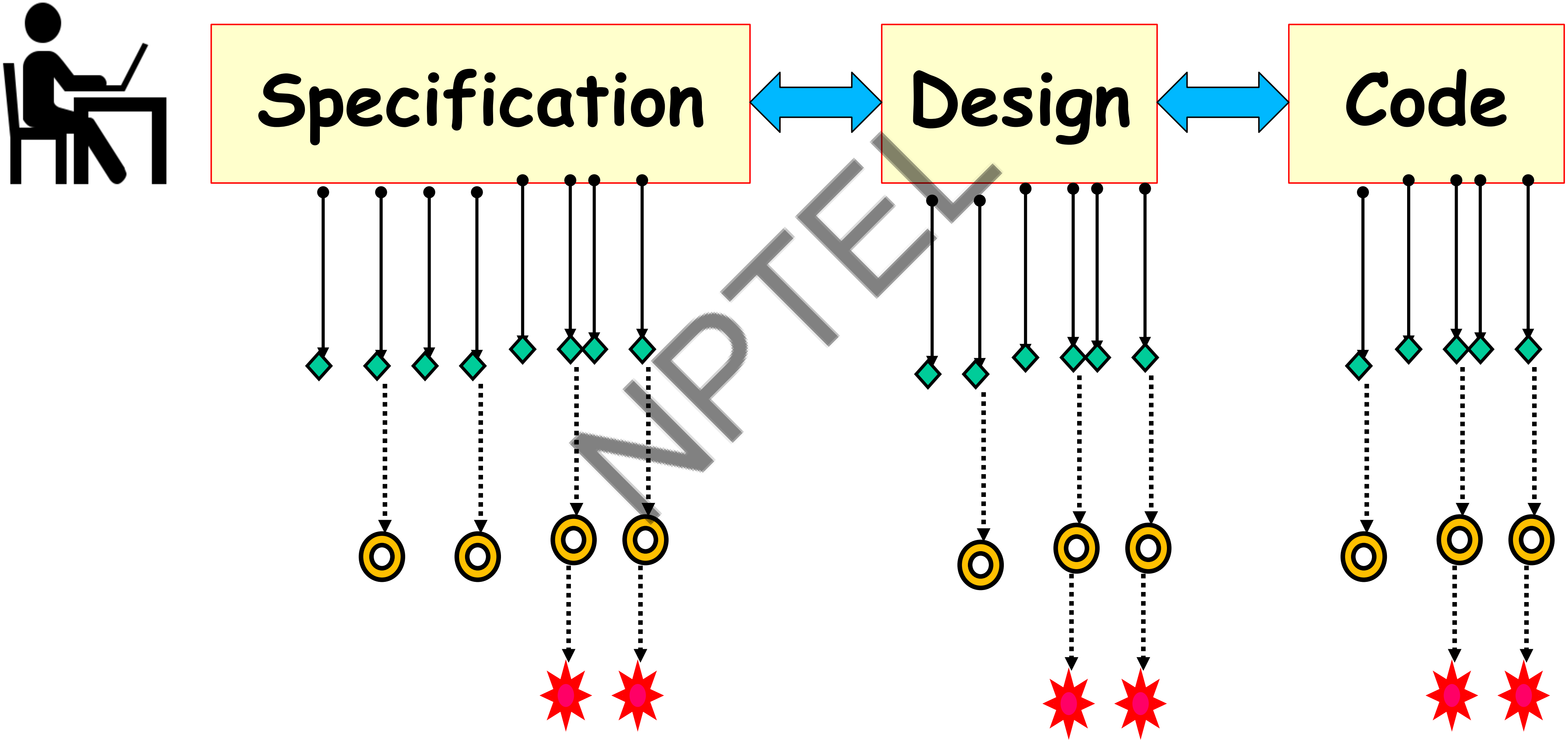
Errors, Faults, Failures

- Programming is human effort-intensive:
  - Therefore, inherently error prone.
- IEEE std 1044, 1993 defined errors and faults as synonyms :
- **IEEE Revision of std 1044 in 2010 introduced finer distinctions:**
  - For more expressive communications distinguished between Errors and Faults

◆ Error or mistake

◎ Fault, defect, or bug

✸ Failure



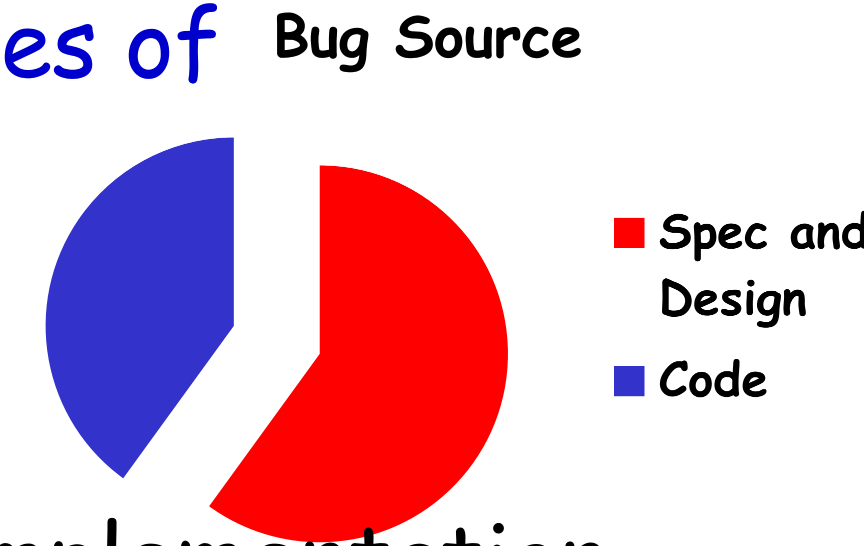
Specification

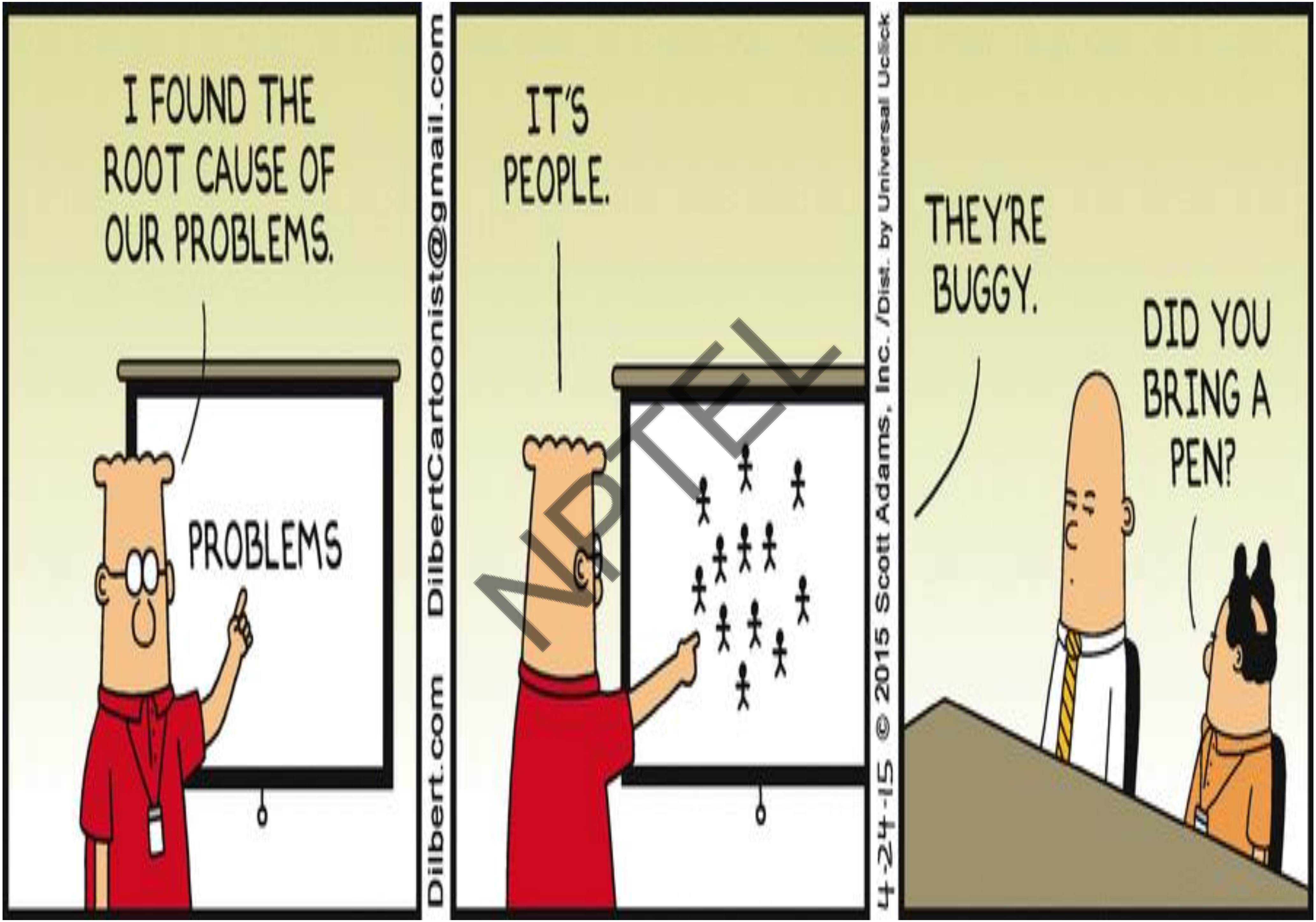
Design

Code

Error Tit-Bits

- Even experienced programmers make many errors:
  - Avg. 50 bugs per 1000 lines of source code
- Extensively tested software contains:
  - About 1 bug per 1000 lines of source code.
- Bug distribution:
  - 60% spec/design, 40% implementation.




I FOUND THE ROOT CAUSE OF OUR PROBLEMS.
PROBLEMS
Dilbert.com DilbertCartoonist@gmail.com
IT'S PEOPLE.
4-24-15 © 2015 Scott Adams, Inc. /Dist. by Universal Uclick
THEY'RE BUGGY.
DID YOU BRING A PEN?

How are Bugs Reduced?

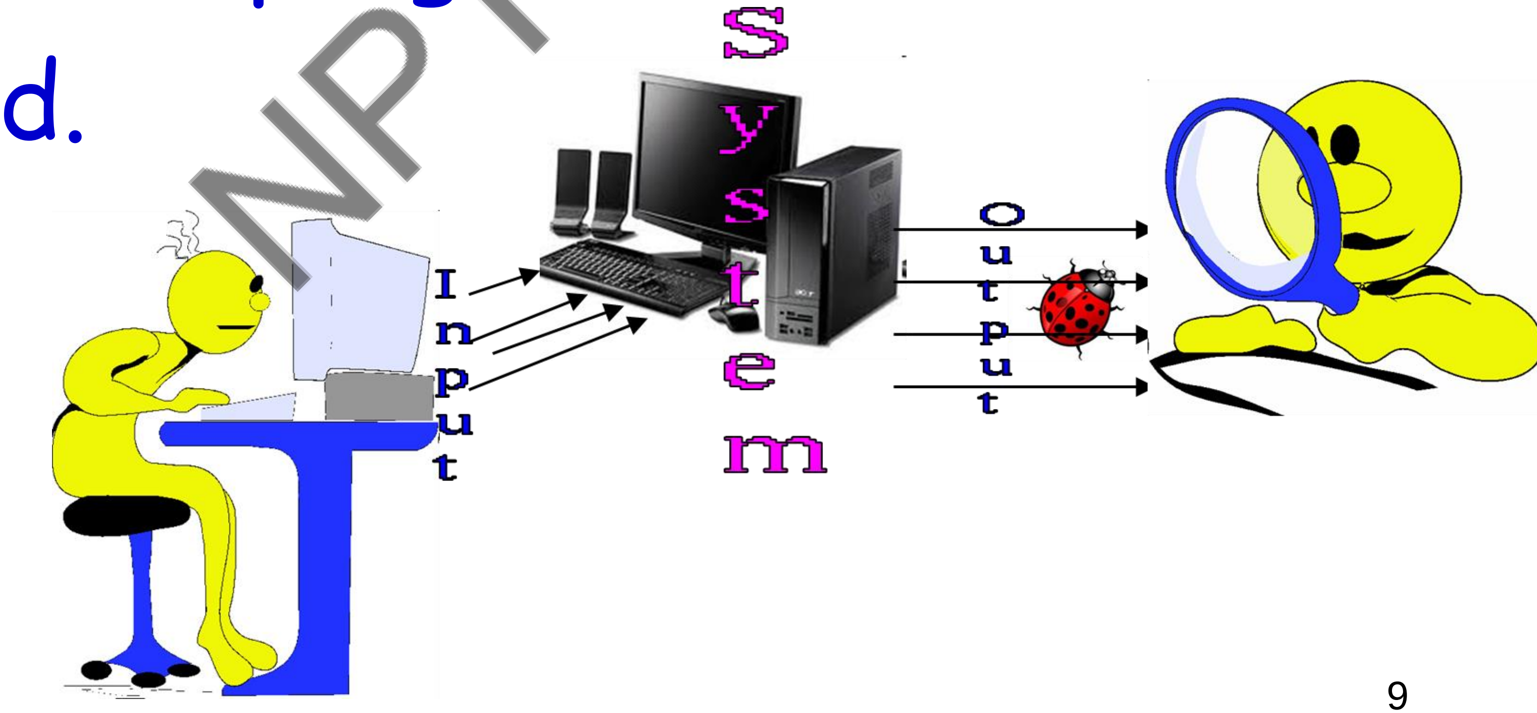
- Review
- **Testing**
- Formal specification and verification
- Use of development process

Cost of Not Adequately Testing

- **Can be enormous**
- Example:
- Ariane 5 rocket self-destructed 37 seconds after launch
- **Reason: A software exception bug went undetected…**
  - Conversion from 64-bit floating point to 16-bit signed integer value had caused an exception
    - The floating point number was larger than 32767
    - Efficiency considerations had led to the disabling of the exception handler.
- **Total Cost: over $1 billion**

How to Test?

- Input test data to the program.
- Observe the output:
  - Check if the program behaved as expected.



Examine Test Result…

- If the program does not behave as expected:
  - Note the conditions under which it failed (Test report).
  - Later debug and correct.

Testing Facts

- Consumes the largest effort among all development activities:
  - Largest manpower among all roles
  - Implies more job opportunities
- About 50% development effort
  - But 10% of development time?
  - How?

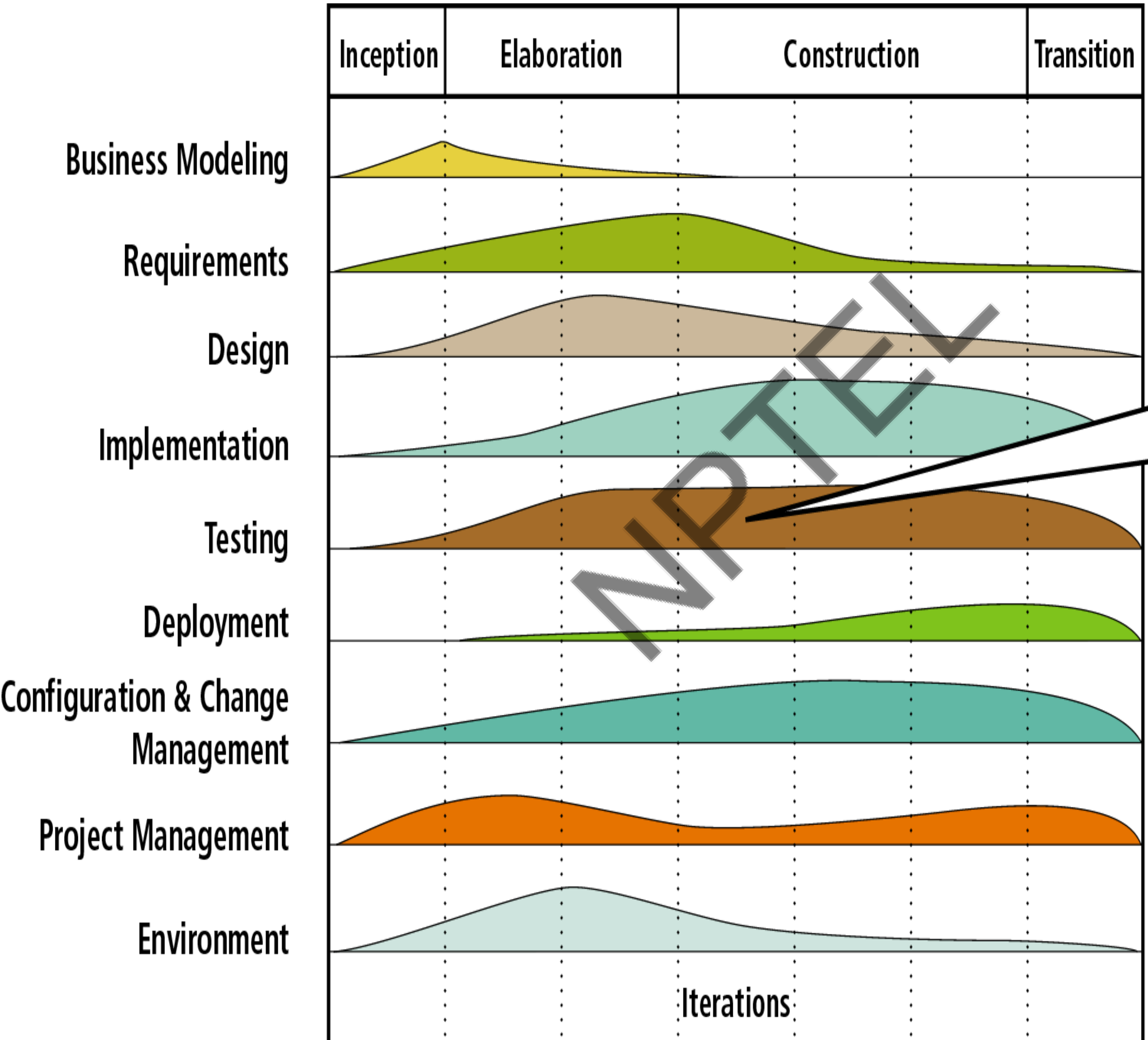
Testing Facts

- Testing is getting more complex and sophisticated every year.
  - Larger and more complex programs
  - Newer programming paradigms
  - Newer testing techniques
  - Test automation

Testing Perception

- Testing is often viewed as not very challenging --- less preferred by novices, but:
  - Over the years testing has taken a center stage in all types of software development.
  - "**Monkey testing is passe**" --- Large number of innovations have taken place in testing area --- requiring tester to have good knowledge of test techniques.
  - Challenges of test automation

Testing Activities Now Spread Over Entire Life Cycle



Define and conduct unit testing

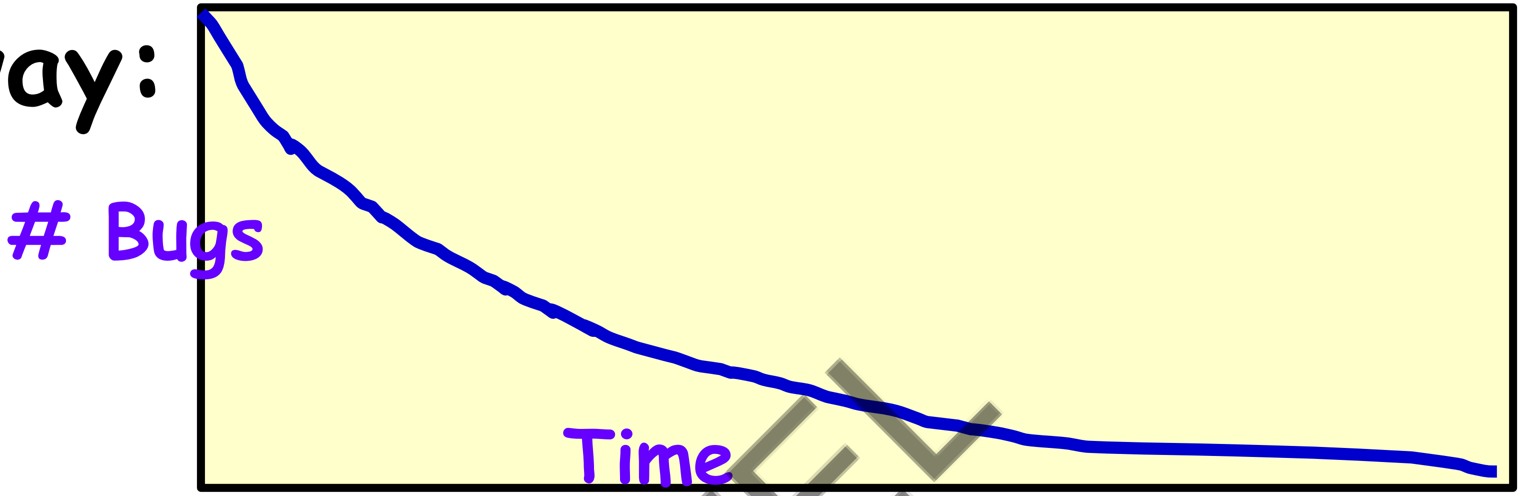
Define and conduct integration testing

Define and conduct usability testing

Define and conduct user acceptance testing

Test How Long?

One way:



- Another way:
  - Seed bugs… run test cases
  - See if all (or most) are getting detected

Verification versus Validation

- Verification is the process of determining:
  - Whether output of one phase of development conforms to its previous phase.
- Validation is the process of determining:
  - Whether a fully developed system conforms to its SRS document..

Verification versus Validation

- Verification is concerned with phase containment of errors:
  - Whereas, the aim of validation is that the final product is error free.

Verification and Validation Techniques

- Review
- Simulation
- Unit testing
- Integration testing

- System testing

Verification	Validation
Are you building it right?	Have you built the right thing?
Checks whether an artifact conforms to its previous artifact.	Checks the final product against the specification.
Done by developers.	Done by Testers.
Static and dynamic activities: reviews, unit testing.	Dynamic activities: Execute software and check against requirements.

Testing Levels

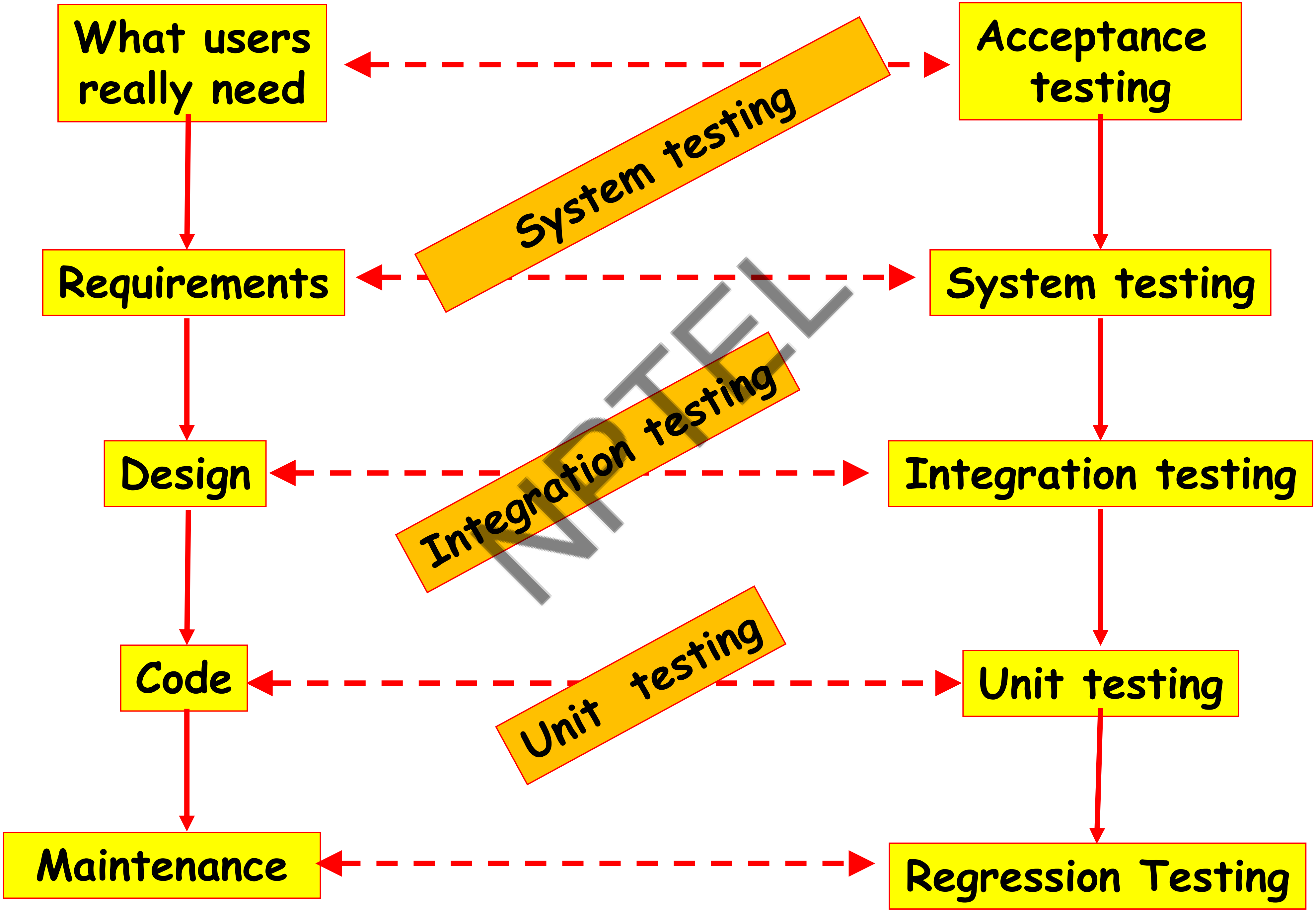
4 Testing Levels

- Software tested at 4 levels:
  - Unit testing
  - Integration testing
  - System testing
  - Regression testing

Test Levels

- Unit testing
  - Test each module (unit, or component) independently
  - Mostly done by developers of the modules
- Integration and system testing
  - Test the system as a whole
  - Often done by separate testing or QA team
- Acceptance testing
  - Validation of system functions by the customer

Levels of Testing



What users really need

Requirements

Design

Code

Maintenance

Acceptance testing

System testing

Integration testing

Unit testing

Regression Testing

System testing

Integration testing

Unit testing

Overview of Activities During System and Integration Testing

- Test Suite Design
- Run test cases
- Check results to detect failures.
- Prepare failure list

Tester

- Debug to locate errors
- Correct errors.

Developer

Quiz 1

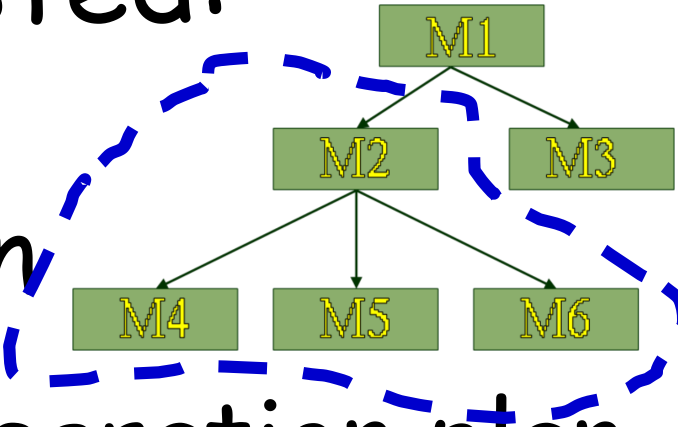
- As testing proceeds more and more bugs are discovered.
  - How to know when to stop testing?
- Give examples of the types of bugs detected during:
  - Unit testing?
  - Integration testing?
  - System testing?

Unit testing

- During unit testing, functions (or modules) are tested in isolation:
  - What if all modules were to be tested together (i.e. system testing)?
    - It would become difficult to determine which module has the error.

Integration Testing

- After modules of a system have been coded and unit tested:
  - Modules are integrated in steps according to an integration plan
  - The partially integrated system is tested at each integration step.



Integration and System Testing

- **Integration test evaluates a group of functions or classes:**
  - Identifies interface compatibility, unexpected parameter values or state interactions, and run-time exceptions
  - **System test tests working of the entire system**
- **Smoke test**:
  - System test performed daily or several times a week after every build.

Types of System Testing

- Based on types test:
  - Functionality test
  - Performance test
- Based on who performs testing:
  - Alpha
  - Beta
  - Acceptance test

Performance test

- Determines whether a system or subsystem meets its non-functional requirements:
  - **Response times**
  - **Throughput**
  - **Usability**
  - **Stress**
  - **Recovery**
  - **Configuration, etc.**

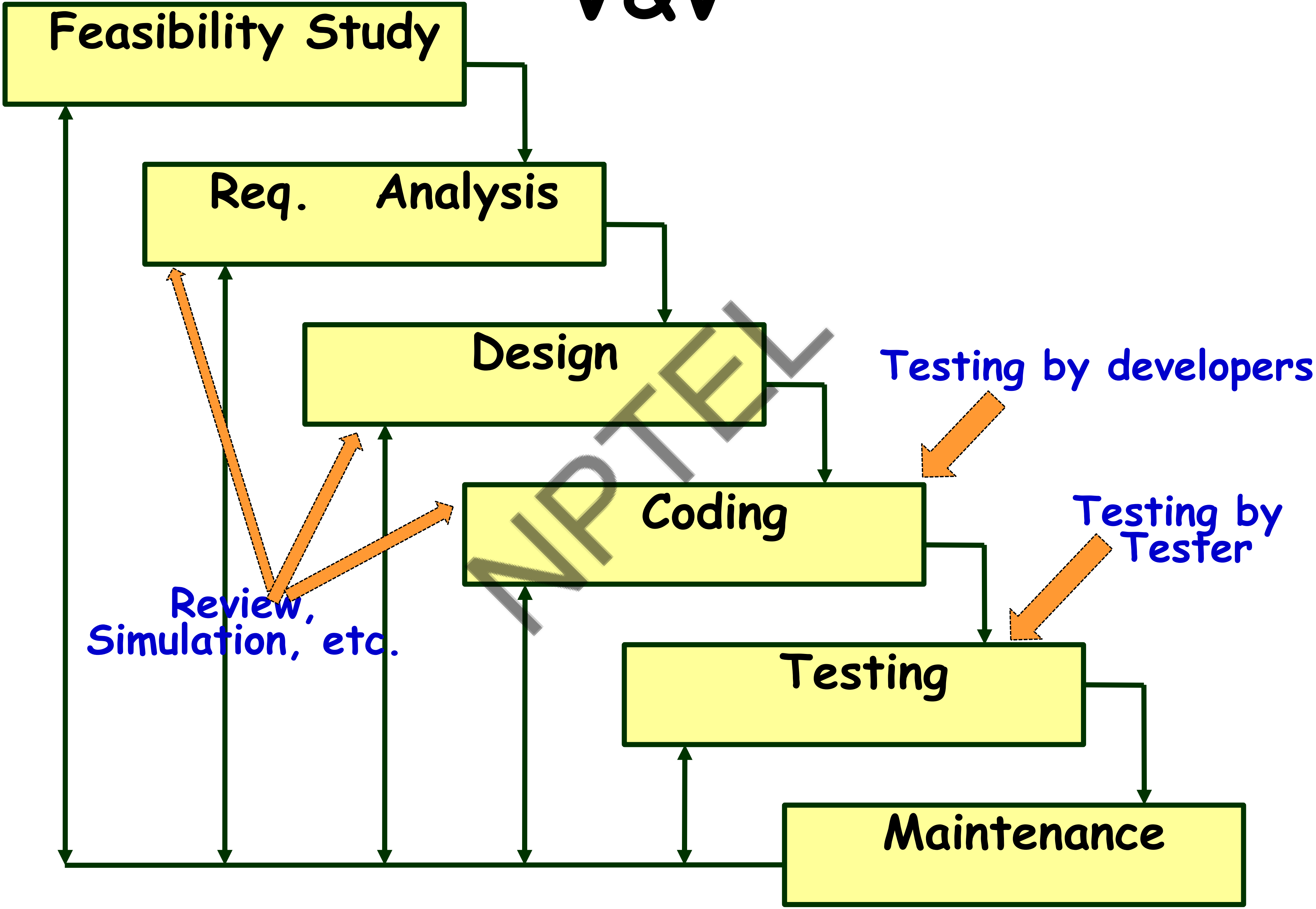
User Acceptance Testing

- User determines whether the system fulfills his requirements
  - **Accepts or rejects delivered system based on the test results.**

Who Tests Software?

- **Programmers:**
  - Unit testing
  - Test their own or other's programmer's code
- **Users:**
  - Usability and acceptance testing
  - Volunteers are frequently used to test beta versions
- **Test team:**
  - All types of testing except unit and acceptance
  - Develop test plans and strategy

V&V



Feasibility Study

Req. Analysis

Design

Coding

Testing

Maintenance

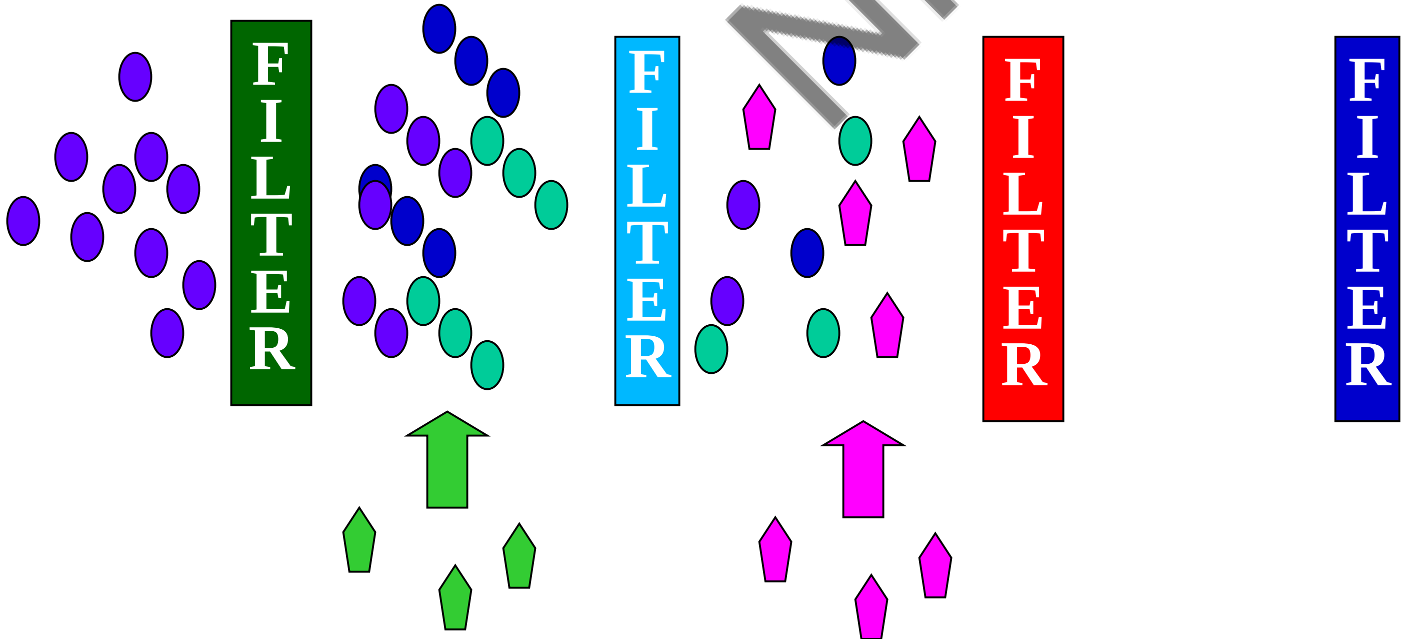
Testing by developers

Testing by Tester

Review, Simulation, etc.

Pesticide Effect

- Errors that escape a fault detection technique:
  - **Can not be detected by further applications of that technique.**



Capers Jones Rule of Thumb

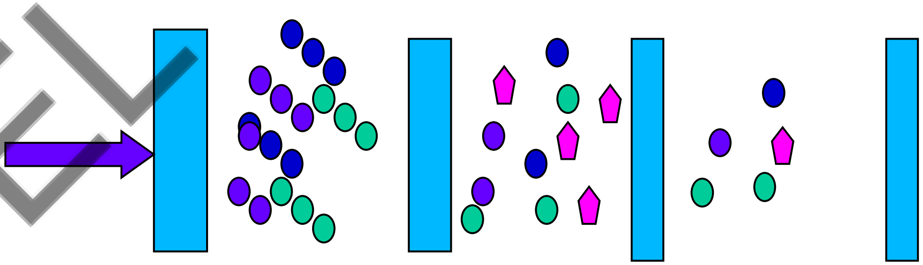
Capers Jones

- Each of software review, inspection, and test step will find 30% of the bugs present.

In IEEE Computer, 1996

Pesticide Effect

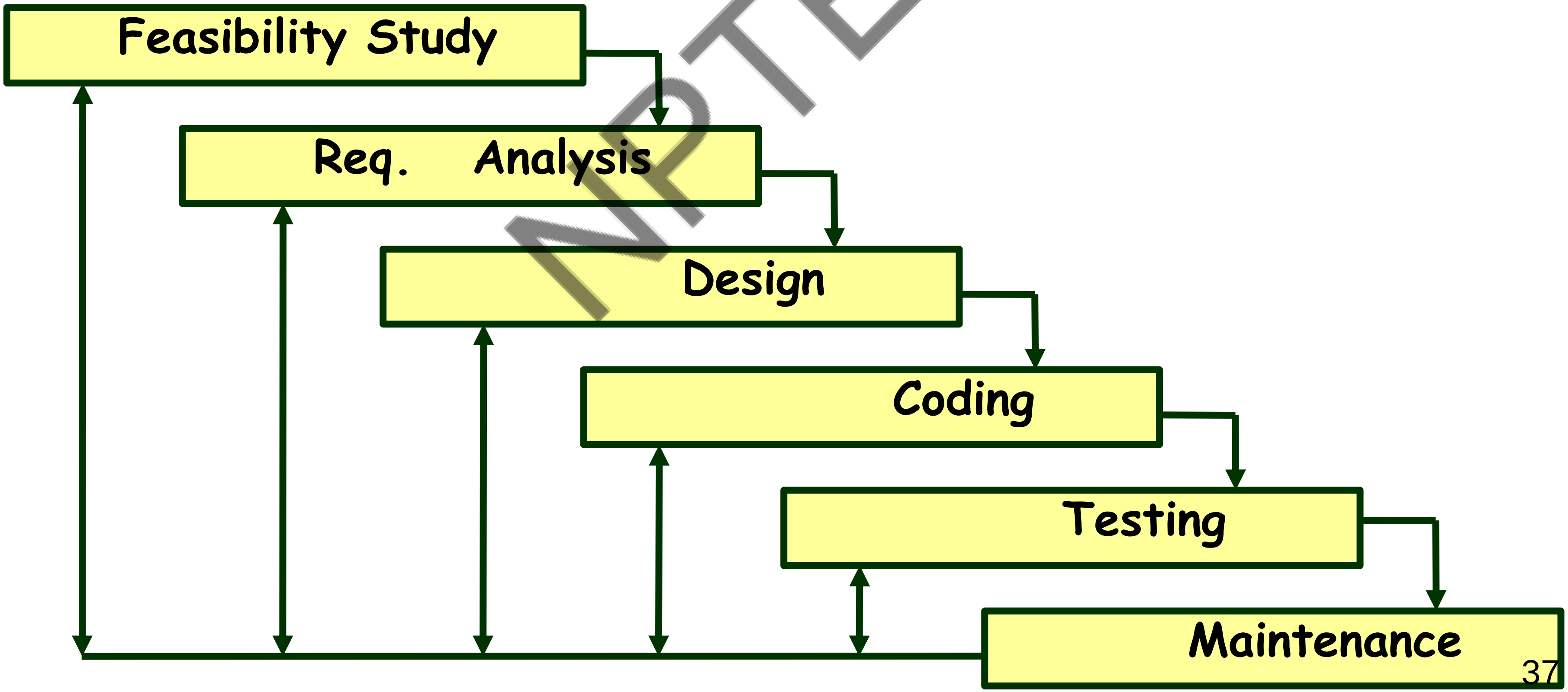
- Assume to start with 1000 bugs
- We use 4 fault detection techniques :



  - Each detects only 70% bugs existing at that time
  - How many bugs would remain at end?
  - **$1000*(0.3)^4$=81 bugs**

Quiz

1. When are verification undertaken in waterfall model?
2. When is testing undertaken in waterfall model?
3. When is validation undertaken in waterfall model?



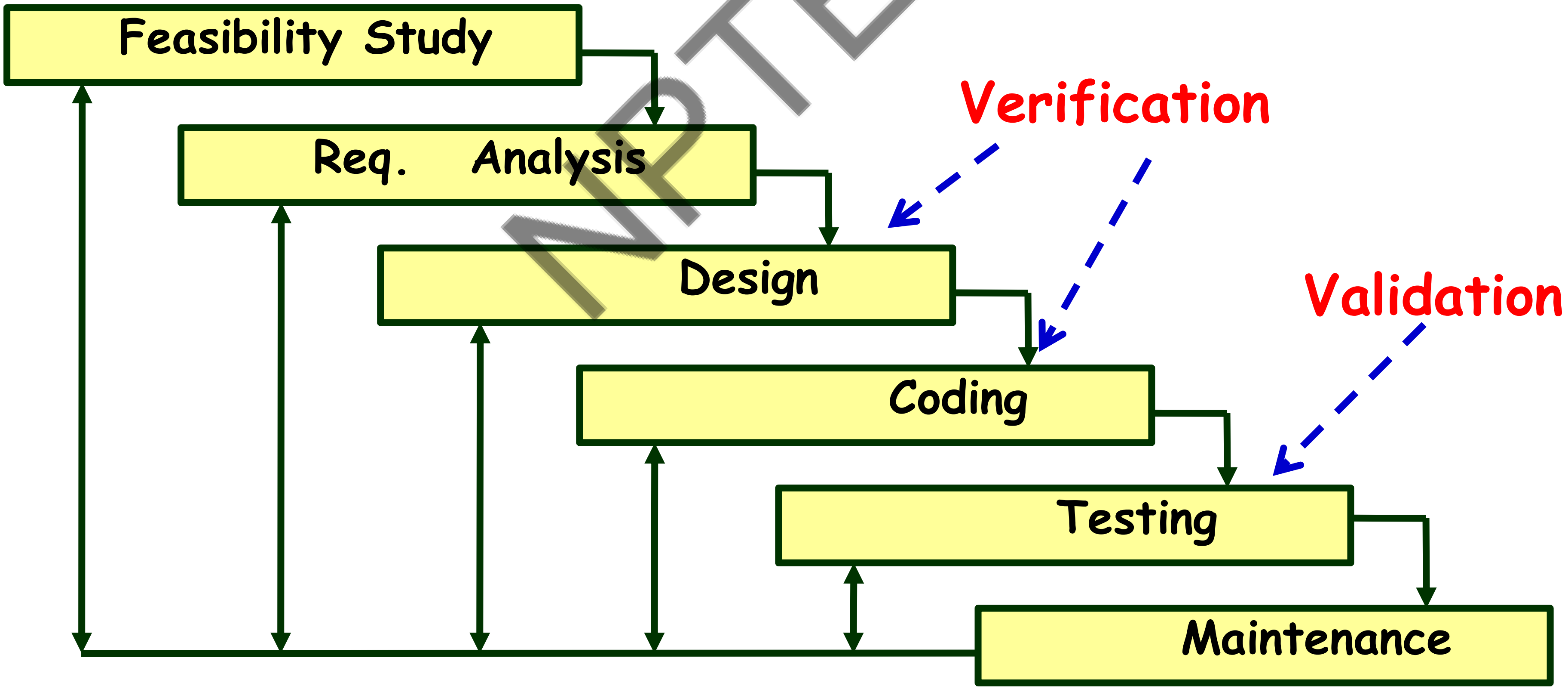
Quiz: Solution

1. When are verification undertaken in waterfall model?

2. When is testing undertaken in waterfall model?

   Ans: Coding phase and Testing phase

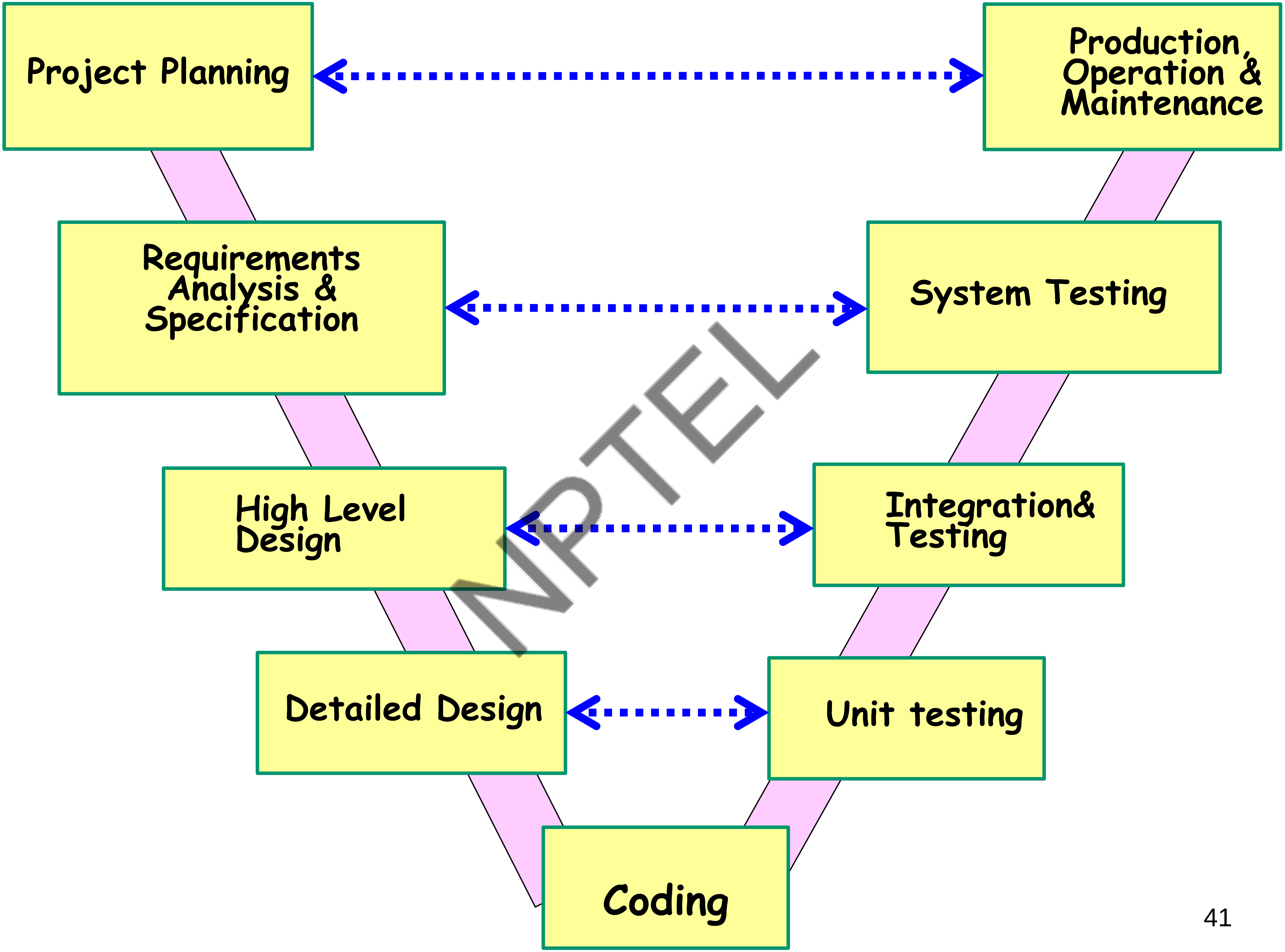
3. When is validation undertaken in waterfall model?



V Life Cycle Model

V Model

- It is a variant of the Waterfall
  - **Emphasizes verification and validation (V&V) activities.**
  - V&V activities are spread over the entire life cycle.
- In every phase of development:
  - **Testing activities are planned in parallel with development.**


Project Planning
Production, Operation & Maintenance
Requirements Analysis & Specification
System Testing
High Level Design
Integration& Testing
Detailed Design
Unit testing
Coding

V Model Steps

- Planning

- Requirements Specification and Analysis

- System and acceptance testing

- Design

- Integration and Testing

V Model: Strengths

- Starting from early stages of software development:
  - **Emphasize planning for verification and validation of the software**
- Each deliverable is made testable
- Intuitive and easy to use

V Model Weaknesses

- Does not support overlapping of phases
- Does not support iterations
- Not easy to handle later changes in requirements
- Does not support any risk handling method

When to Use V Model

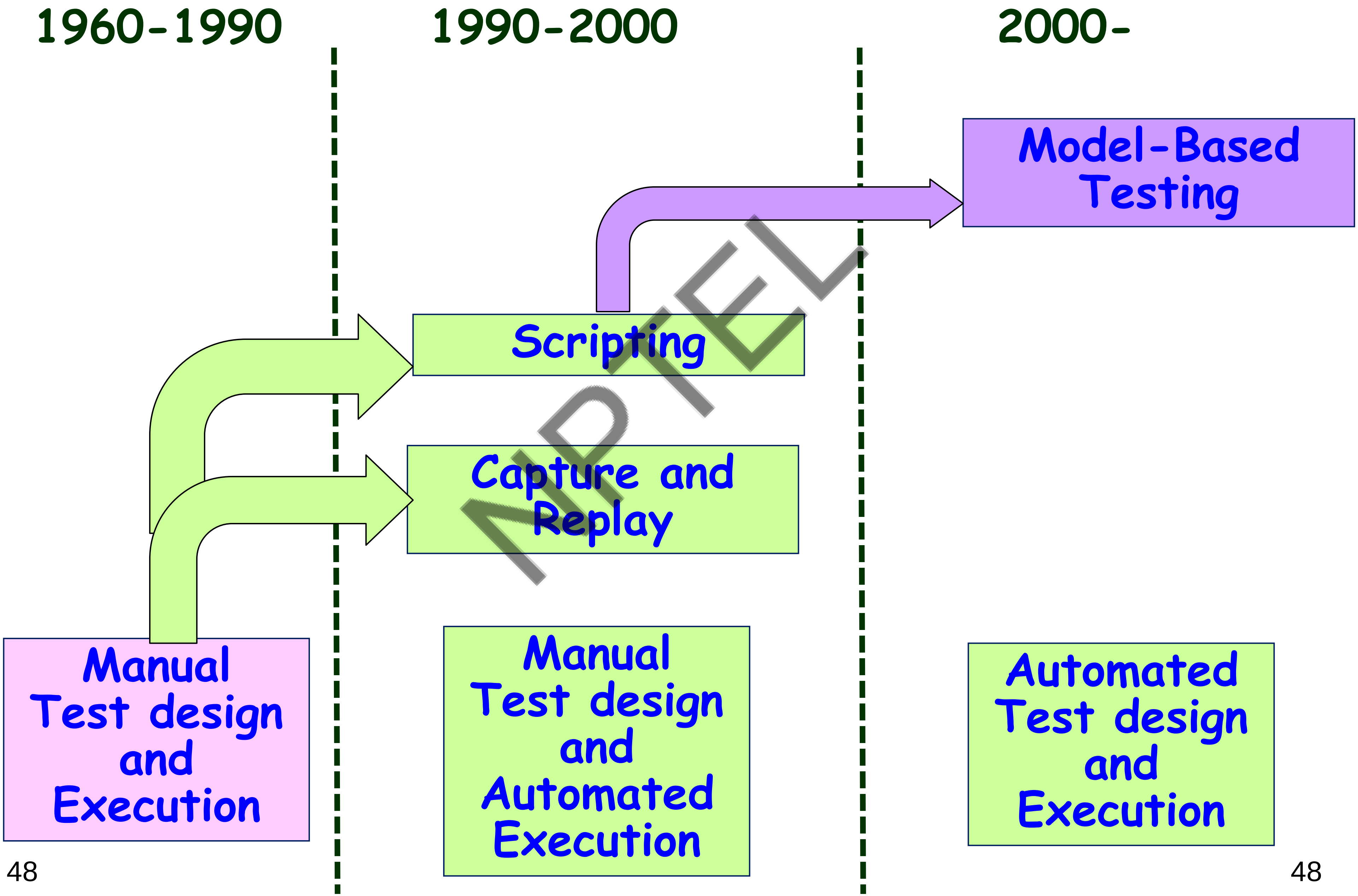
- Natural choice for systems requiring high reliability:
- **Example: embedded control applications:**
  - All requirements are known up-front
  - Solution and technology are known

A Few More Basic Concepts on Testing…

How Many Latent Errors?

- Several independent studies [Jones],[schroeder], etc. conclude:
  - 85% errors get removed at the end of a typical testing process.
  - Why not more?
  - All practical test techniques are basically heuristics… they help to reduce bugs… but do not guarantee complete bug removal…

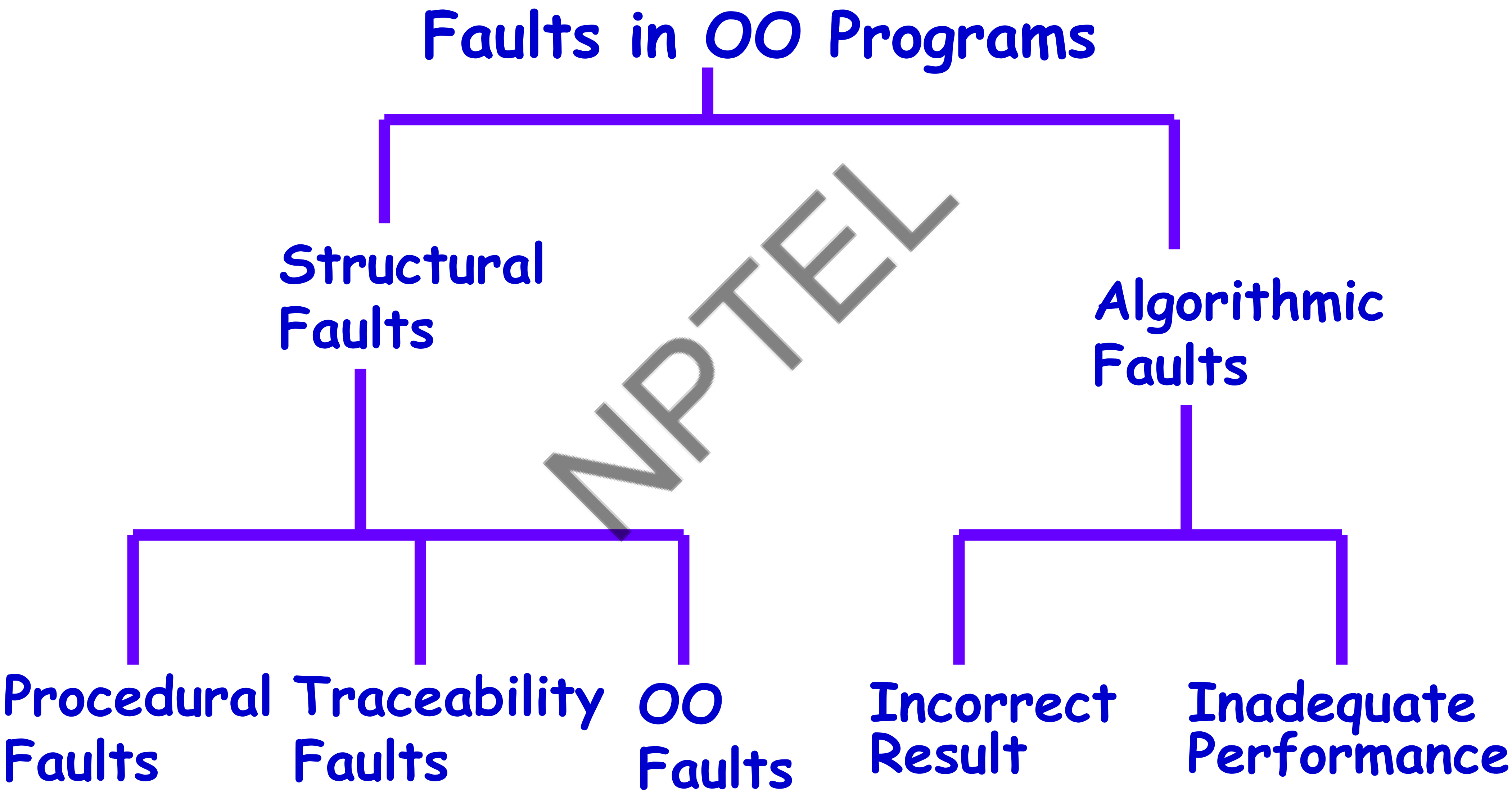
Evolution of Test Automation


1960-1990
1990-2000
2000-
Model-Based Testing
Scripting
Capture and Replay
Manual Test design and Execution
Manual Test design and Automated Execution
Automated Test design and Execution

Fault Model

- Types of faults possible in a program.
- Some types can be ruled out:
  - For example, file related-problems in a program not using files.

Fault Model of an OO Program



Hardware Fault-Model

- Essentially only four types:
  - Stuck-at 0
  - Stuck-at 1
  - Open circuit
  - Short circuit
- Testing is therefore simple:
  - Devise ways to test the presence of each
- Hardware testing is usually fault-based testing.

Test Cases

- Each test case typically tries to establish correct working of some functionality:
  - Executes (covers) some program elements.
  - For certain restricted types of faults, fault-based testing can be used.

Test data versus test cases

- **Test data**:
  - Inputs used to test the system

- **Test cases:**
  - Inputs to test the system,
  - State of the software, and
  - The predicted outputs from the inputs

Test Cases and Test Suites

- A test case is a triplet [I,S,O]
  - I is the data to be input to the system,
  - S is the state of the system at which the data will be input,
  - O is the expected output of the system.

What are Negative Test Cases?

- **Purpose:**
    - Helps to ensure that the application gracefully handles invalid and unexpected user inputs and the application does not crash.
- **Example:**
    - If user types letter in a numeric field, it should not crash and display the message: **"incorrect data type, please enter a number..."**

Test Cases and Test Suites

- Test a software using a set of carefully designed test cases:
  - The set of all test cases is called the **test suite.**

Test Execution Example: Return Book

- Test case [I,S,O]

1. **Set the program in the required state:** Book record created, member record created, Book issued

2. **Give the defined input:** Select renew book option and request renew for a further 2 wk period.

3. **Observe the output:**

   - Compare it to the expected output.

Sample: Recording of Test Case & Results

Test Case number

Test Case author

Test purpose

Pre-condition:

Test inputs:

Expected outputs (if any):

Post-condition:

Test Execution history:

- Test execution date
- Test execution person
- Test execution result (s) : Pass/Fail
  - If failed : Failure information
  - : fix status

Test Team- Human Resources

- **Test Planning:** Experienced people
- **Test scenario and test case design:** Experienced and test qualified people
- **Test execution:** semi-experienced to inexperienced
- **Test result analysis:** experienced people
- **Test tool support:** experienced people
- May include external people:
  - **Users**
  - **Industry experts**

Why Design of Test Cases?

- Exhaustive testing of any non-trivial system is impractical:
  - Input data domain is extremely large.
- Design an optimal test suite, meaning:
  - Of reasonable size, and
  - Uncovers as many errors as possible.

Design of Test Cases

- If test cases are selected randomly:
  - Many test cases would not contribute to the significance of the test suite,
  - Would only detect errors that are already detected by other test cases in the suite.
- Therefore, the number of test cases in a randomly selected test suite:
  - Does not indicate the effectiveness of testing.

Design of Test Cases

- Testing a system using a large number of randomly selected test cases:
  - **Does not mean that most errors in the system will be uncovered.**
- Consider following example:
  - Find the maximum of two integers x and y.

Design of Test Cases

- The code has a simple programming error:

-
  ```
  If (x>y) max = x;
      else max = x; // should be max=y;
  ```
- Test suite {(x=3,y=2);(x=2,y=3)} can detect the bug,
- A larger test suite {(x=3,y=2);(x=4,y=3);(x=5,y=1)} does not detect the bug.

Test Plan

- Before testing activities start, a test plan is developed.
- The test plan documents the following:
  - Features to be tested
  - Features not to be tested
  - Test strategy
  - Test suspension criteriastopping criteria
  - Test effort
  - Test schedule

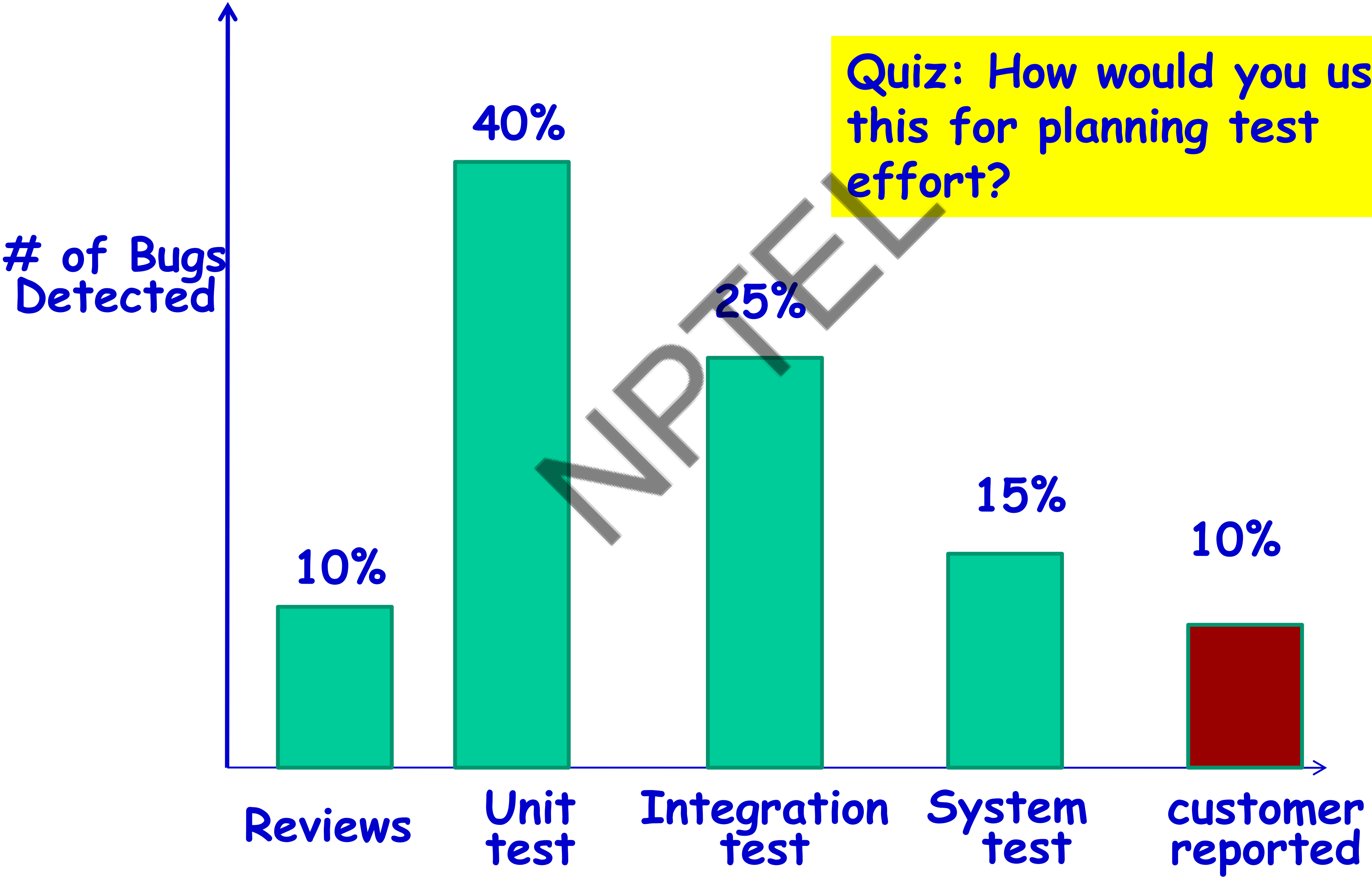
Design of Test Cases

- Systematic approaches are required to design an effective test suite:
  - Each test case in the suite should target different faults.

Testing Strategy

- Test Strategy primarily addresses:
  - **Which types of tests to deploy?**
  - **How much effort to devote to which type of testing?**
    - **Black-box: Usage-based testing** (based on customers' actual usage pattern)
    - **White-box testing** can be guided by black box testing results

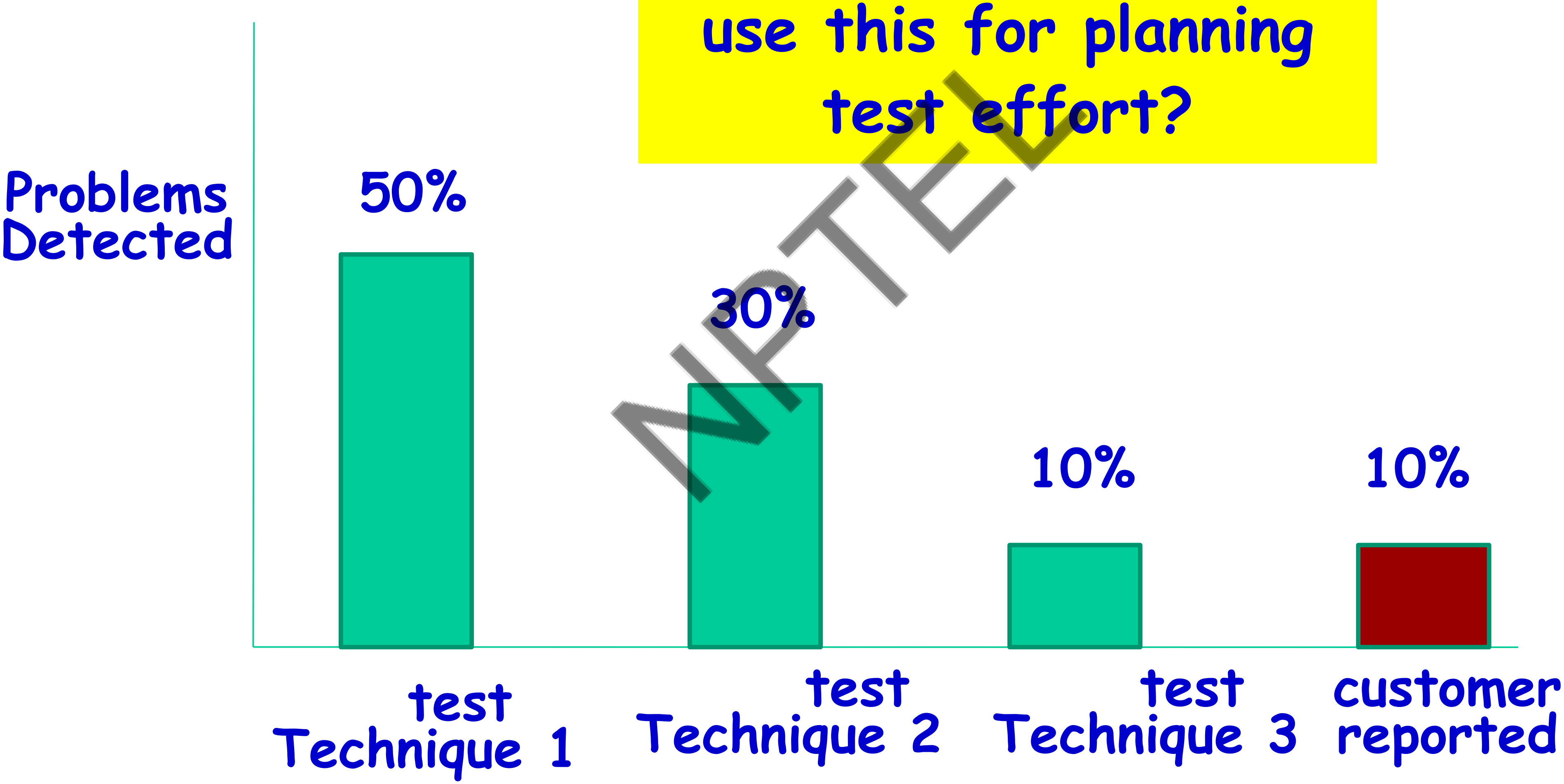
Consider Past Bug Detection Data…



Quiz: How would you use this for planning test effort?

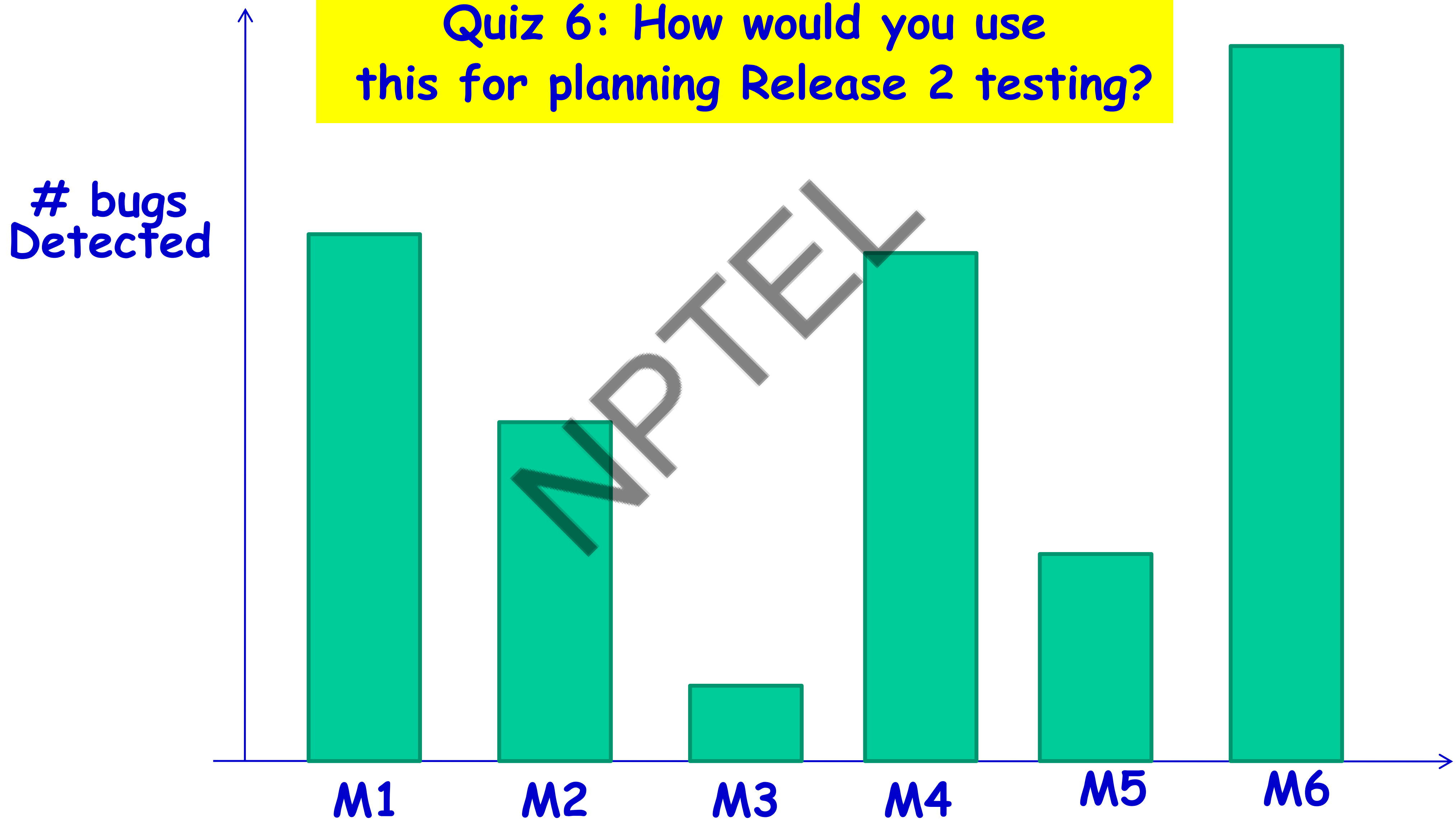
Consider Past Bug Detection Data…

Quiz: How would you use this for planning test effort?



Distribution of Error Prone Modules customer reported bugs for Release 1

Quiz 6: How would you use this for planning Release 2 testing?



Defect clustering: A few modules usually contain most defects…

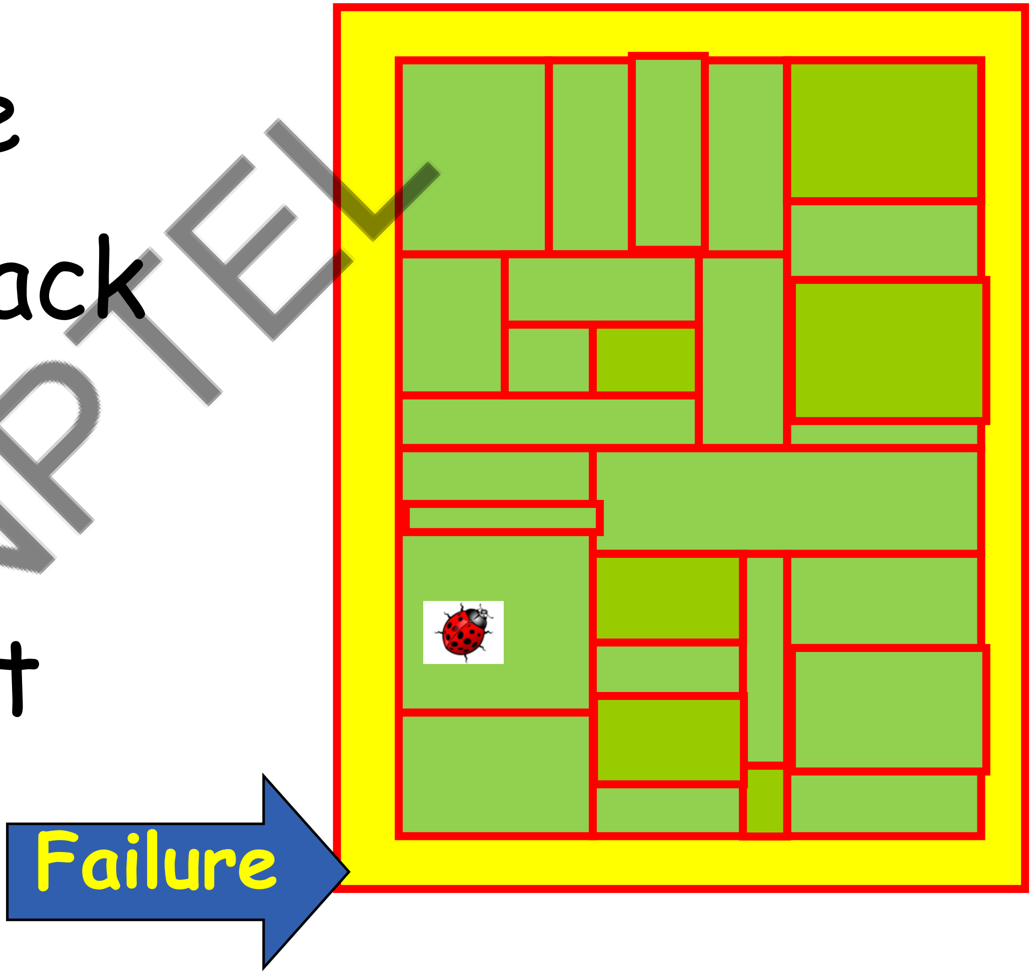
Unit Testing

When and Why of Unit Testing?

- Unit testing carried out:
  - After coding of a unit is complete and it compiles successfully.
- Unit testing reduces debugging effort substantially.

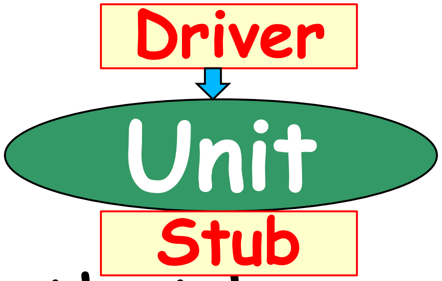
Why unit test?

- Without unit test:
  - Errors become difficult to track down.
  - Debugging cost increases substantially…

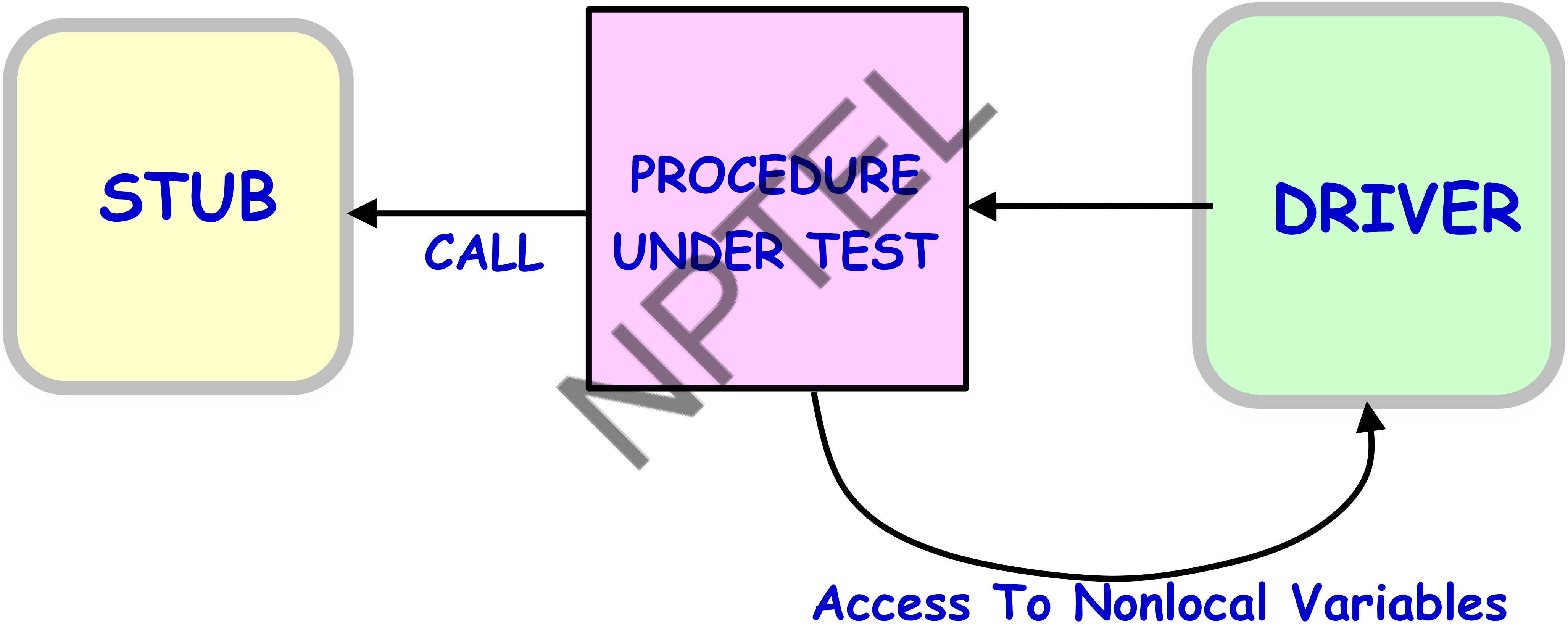


Unit Testing

- **Testing of individual methods, modules, classes, or components in isolation:**
  - Carried out before integrating with other parts of the software being developed.
- Support required for for Unit testing:
  - **Driver**
    - Simulates the behavior of a function that calls and possibly supplies some data to the function being tested.
  - **Stub**
    - Simulates the behavior of a function that has not yet been written.



Unit Testing



Quiz

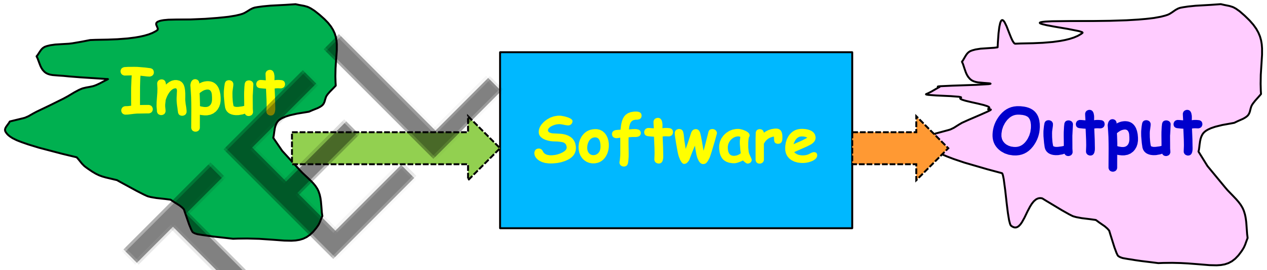
- Unit testing can be considered as which one of the following types of activities?
  - **Verification**
  - **Validation**

Design of Unit Test Cases

- There are essentially three main approaches to design test cases:
  - **Black-box approach**
  - **White-box (or glass-box) approach**
  - **Grey-box approach**

Black-Box Testing

- Test cases are designed using only functional specification of the software:



  - Without any knowledge of the internal structure of the software.

- Black-box testing is also known as functional testing.

What is Hard about BB Testing

- Data domain is large
- A function may take multiple parameters:
  - We need to consider the combinations of the values of the different parameters.

What's So Hard About Testing?

- Consider

```
int check-equal(int x, int y)
```

- Assuming a 64 bit computer
  - Input space = 2^{128}

- Assuming it takes 10secs to key-in an integer pair:
  - It would take about a billion years to enter all possible values!
  - Automatic testing has its own problems!

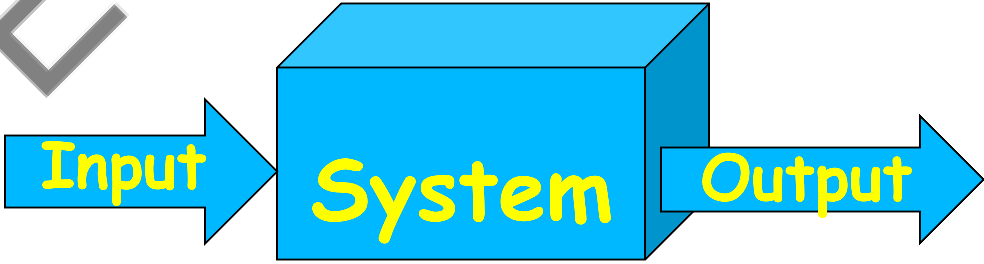
Solution

- Construct model of the data domain:
  - Called Domain based testing
  - Select data based on the domain model

Black-box Testing

Black Box Testing

- Considers the software as a black box:
  - Test data derived from the specification
    - No knowledge of code necessary
- Also known as:
  - Data-driven or
  - Input/output driven testing



- The goal is to achieve the thoroughness of exhaustive input testing:
  - With much less effort!!!!

Black-Box Testing

- Scenario coverage
- Equivalence class partitioning
- Special value (risk-based) testing
  - Boundary value testing
  - Cause-effect (Decision Table) testing
  - Combinatorial testing
  - Orthogonal array testing

Black-Box Testing

- Scenario coverage
- Equivalence class partitioning
- Special value (risk-based) testing
  - Boundary value testing
  - Cause-effect (Decision Table) testing
  - Combinatorial testing
  - Orthogonal array testing

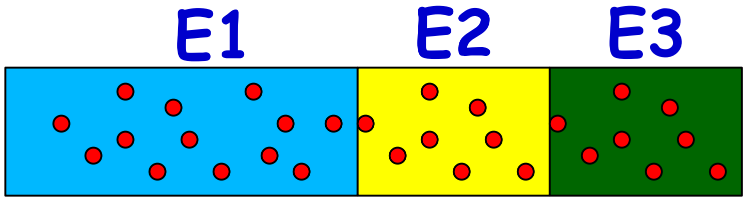
Equivalence Class Testing

Equivalence Class Partitioning

- The input values to a program:
  - Partitioned into equivalence classes.
- Partitioning is done such that:
  - **Program behaves in similar ways to every input value belonging to an equivalence class.**
  - **At the least there should be as many equivalence classes as scenarios.**

Why Define Equivalence Classes?

- Premise:



  - Testing code with any one representative value from a equivalence class:
  - As good as testing using any other values from the equivalence class.

Equivalence Class Partitioning

- How do you identify equivalence classes?
  - Identify scenarios
  - Examine the input data.
  - Examine output
- Few guidelines for determining the equivalence classes can be given…

Guidelines to Identify Equivalence Classes

- If an input condition specifies a range, one valid and two invalid equivalence class are defined.

- If an input condition specifies a member of a set, one valid and one invalid equivalence classes are defined.

- If an input condition is Boolean, one valid and one invalid classes are defined.

- Example:

- Area code: range --- value defined between 10000 and 90000

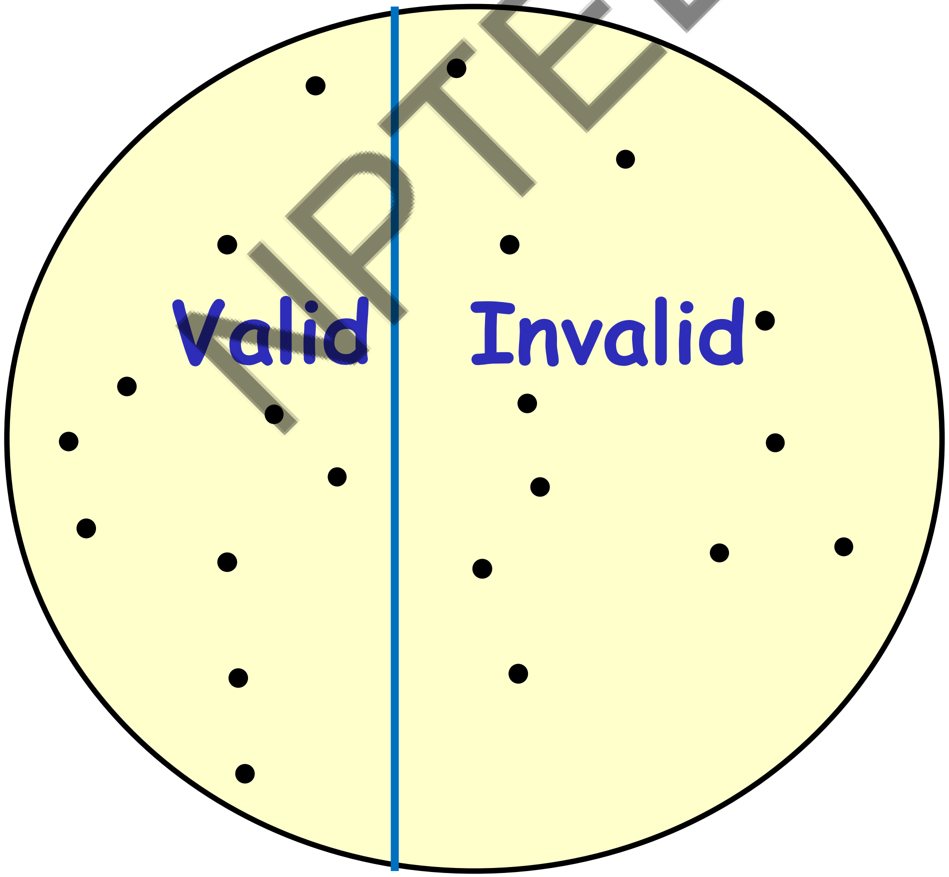
- Password: value - six character string.

Equivalent class partition: Example

- Given three sides, determine the type of the triangle:
  - Isosceles
  - Scalene
  - Equilateral, etc.
- Hint: scenarios expressed in output in this case.

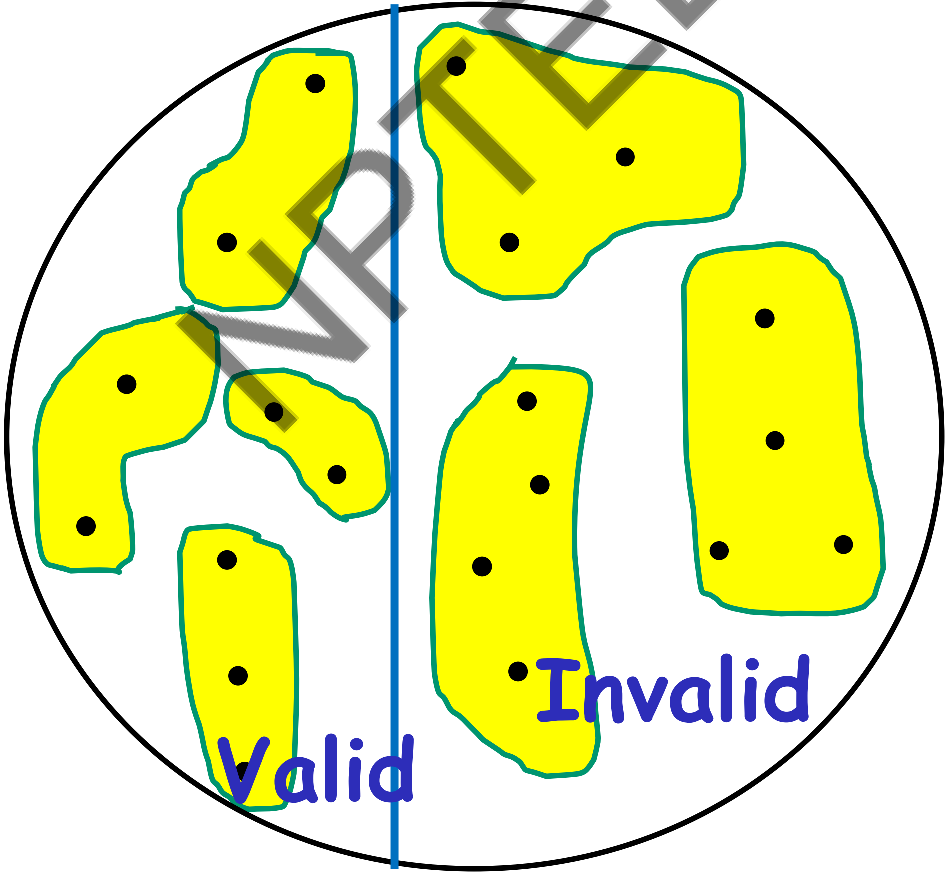
Equivalence Partitioning

- First-level partitioning:
  - Valid vs. Invalid test cases



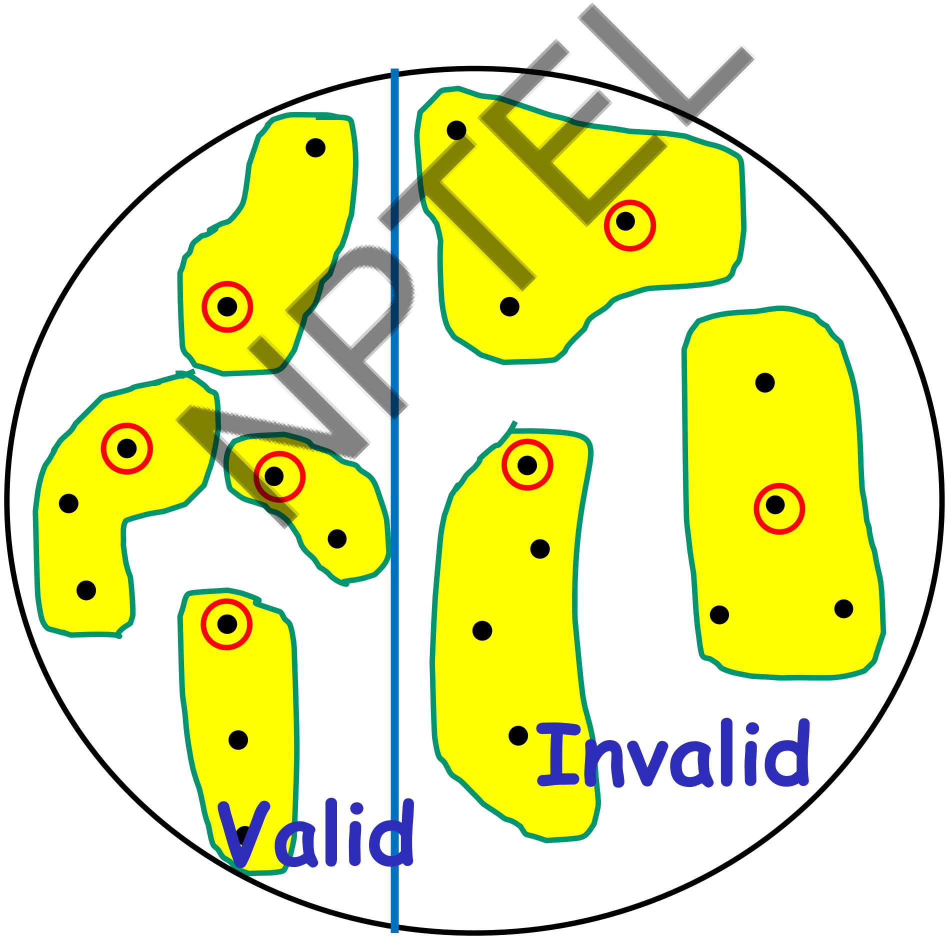
Equivalence Partitioning

- Further partition valid and invalid test cases into equivalence classes



Equivalence Partitioning

- Create a test case for at least one value from each equivalence class


Valid
Invalid