Software Testing

Romenia Silva

Faculty of Computing – University of Sri Jayewardenepura

romaniasilva@sjp.ac.lk

Definition of Testing

"Testing is the process of executing a program with the intent of finding errors"

— Glen Myers

Testing Objectives

Testing is obviously concerned with errors, faults, failures and incidents. A test is the act of exercising software with test cases with an objective of

- *Finding failure*
- *Demonstrate correct execution*

— Paul Jorgensen

Systems Concern

According to Paul Jorgensen, software testing is obviously and deeply concerned with identifying, analyzing, and mitigating **errors, faults, failures,** and **incidents**.

Dual Objective

A test represents the systematic act of exercising software with carefully constructed test cases with the primary target objectives of:

- Finding failure
- Demonstrate correct execution

What Is a "Good" Test?

- A good test has a **high probability of finding an error**.
- A good test is **not redundant**, ensuring execution paths are optimized.
- A good test should be **"best of breed"** to yield maximum quality output.
- A good test should be **neither too simple nor too complex**.

TEST CASE DESIGN

"Bugs lurk in corners and congregate at boundaries..."

by Boris Beizer

OBJECTIVE:
to uncover errors

CRITERIA:
in a complete manner

CONSTRAINT:
with a minimum of effort and time

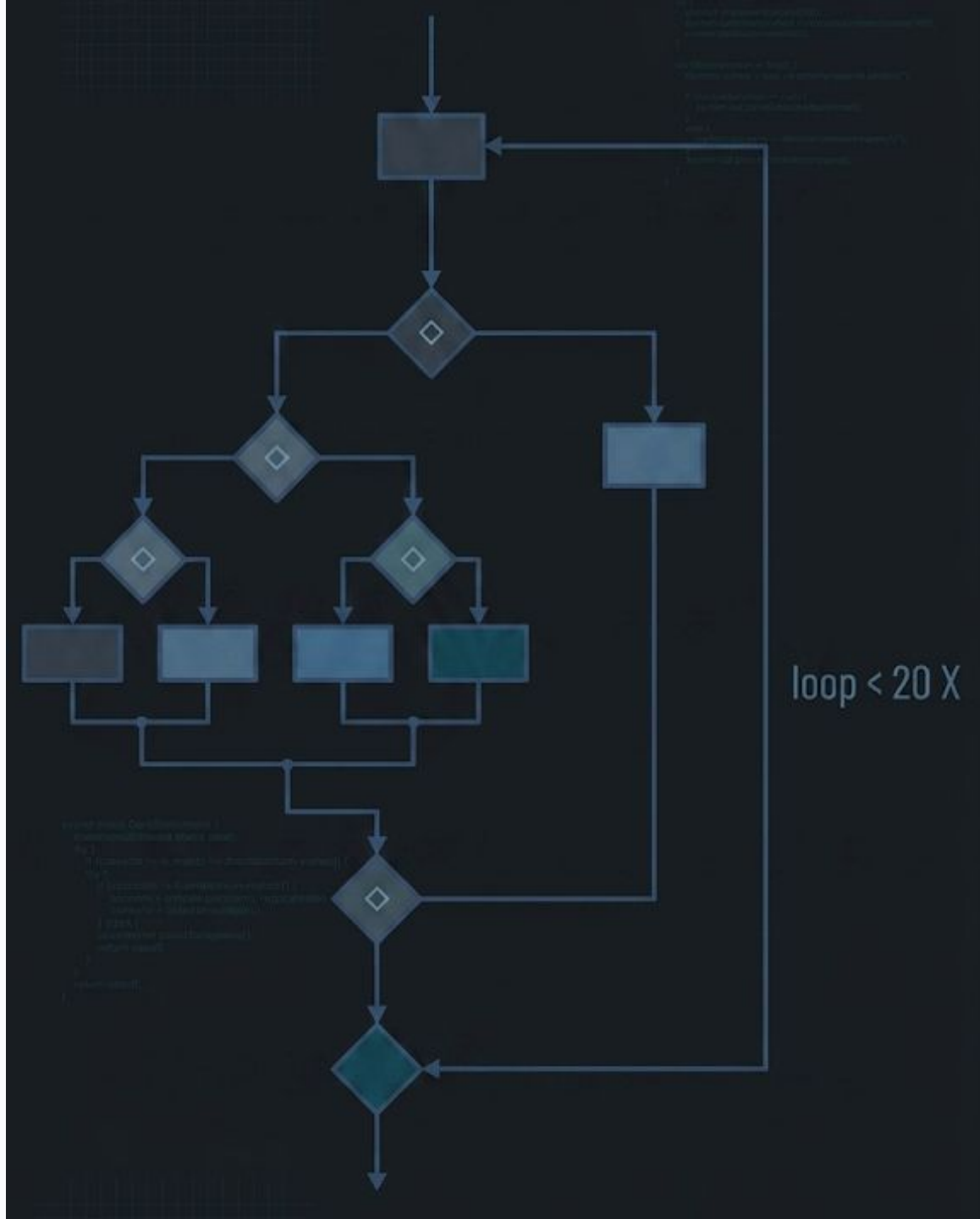
Exhaustive Testing

For a basic logical condition of size:

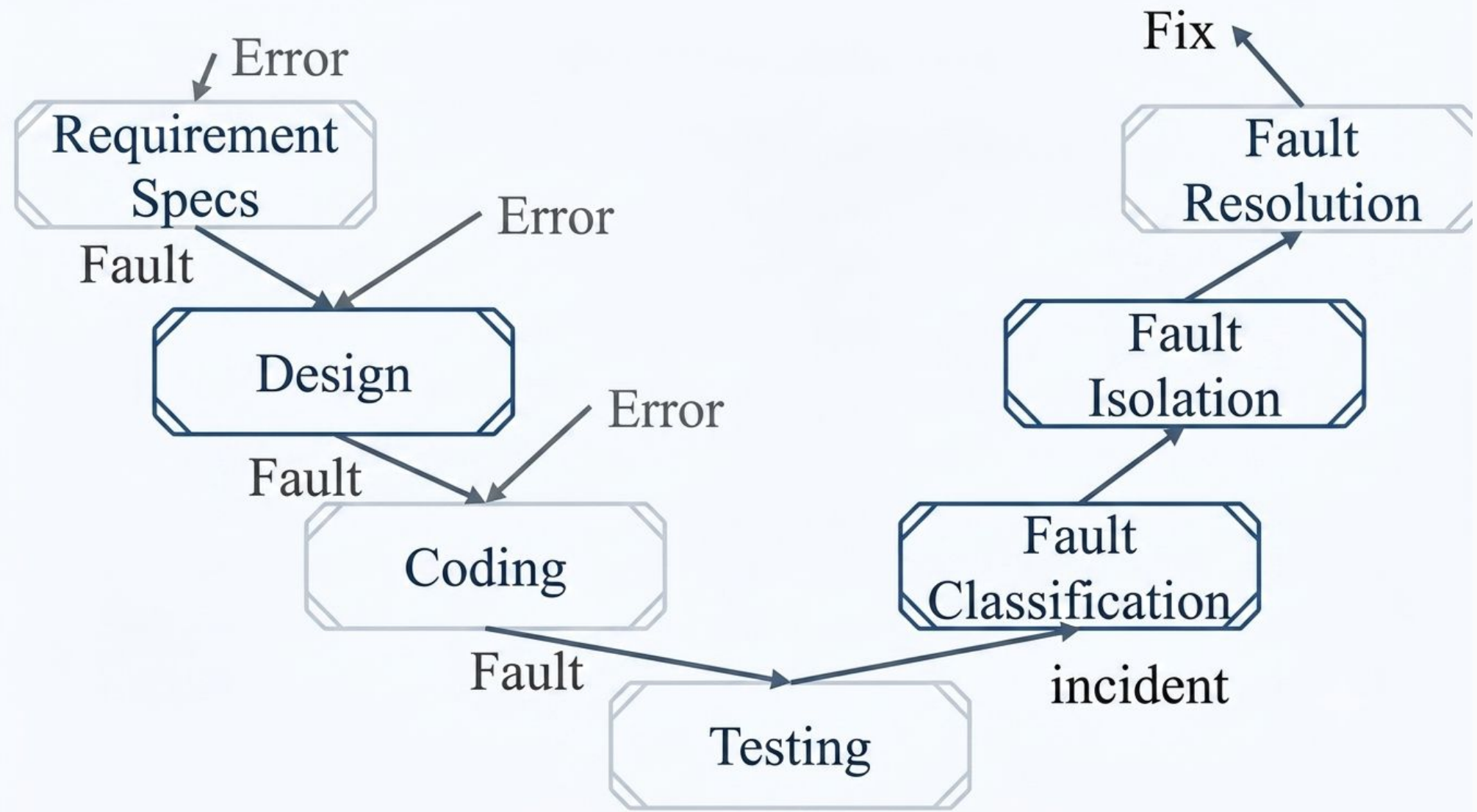
```
loop < 20
```

There are 10^{14} possible paths!

If we execute one test per millisecond, it would take **3,170 years** to test this program!! This shows exhaustive coverage is practically impossible.



A Testing Life Cycle



Testing Terminology

Error

Represents **mistakes made by people** (developers, analysts, or designers) during the system construction or specification process.

Fault

The concrete result of an error. It may be categorized as:

- **Fault of Commission:** We enter something into a representation that is incorrect.
- **Fault of Omission:** Designer makes an error of omission; the resulting fault is that something is missing that should have been present in the representation.

More Key Terminology

Failure

Occurs specifically when a dormant fault is executed at runtime, causing the system to behave in an unexpected manner.

Incident

Behavior of fault. An incident is the symptom(s) associated with a failure that alerts user to the occurrence of a failure

Test Case

Associated directly with program behavior. It carries a set of inputs and a list of expected outputs.

Verification & Validation

Verification

Are we building the product right?

The detailed process of determining whether the output of one phase of software development conforms to its previous phase.

Validation

Are we building the right product?

The high-level process of determining whether a fully developed software system conforms directly to its Software Requirements Specification (SRS) document.

Verification vs Validation

Phase Containment

Verification is primarily concerned with the phase containment of errors.

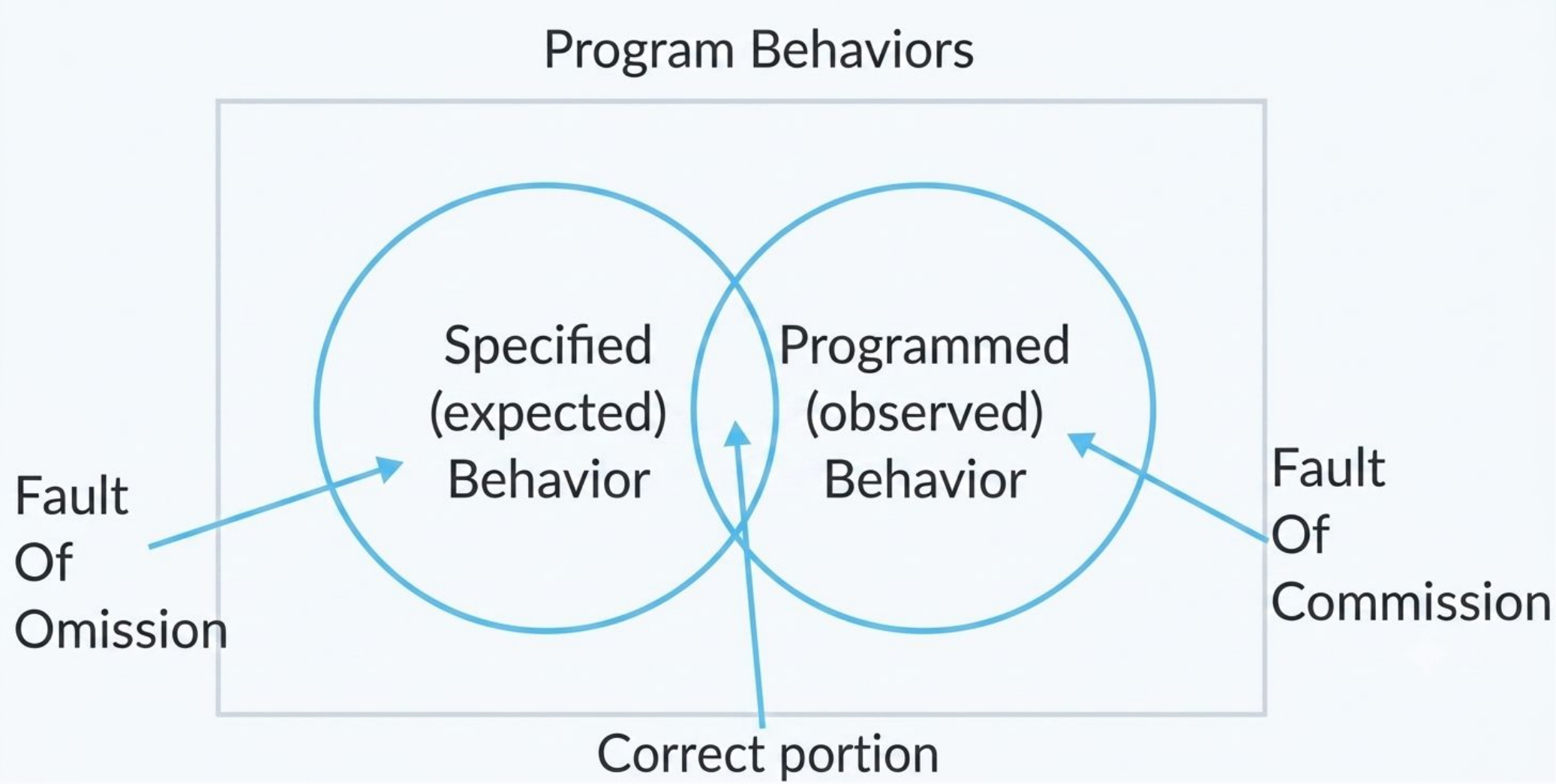
It ensures that software is validated incrementally at each developmental milestone, preventing errors from passing down the lifecycle.

Final Deliverable

Validation is concerned about the final product being error-free.

It checks user expectation matching and operational sanity so that the end product works precisely as desired in production.

Relationship – Program Behaviors



Classification of Tests

Granularity Level

Distinguishes the testing depth and integration scopes:

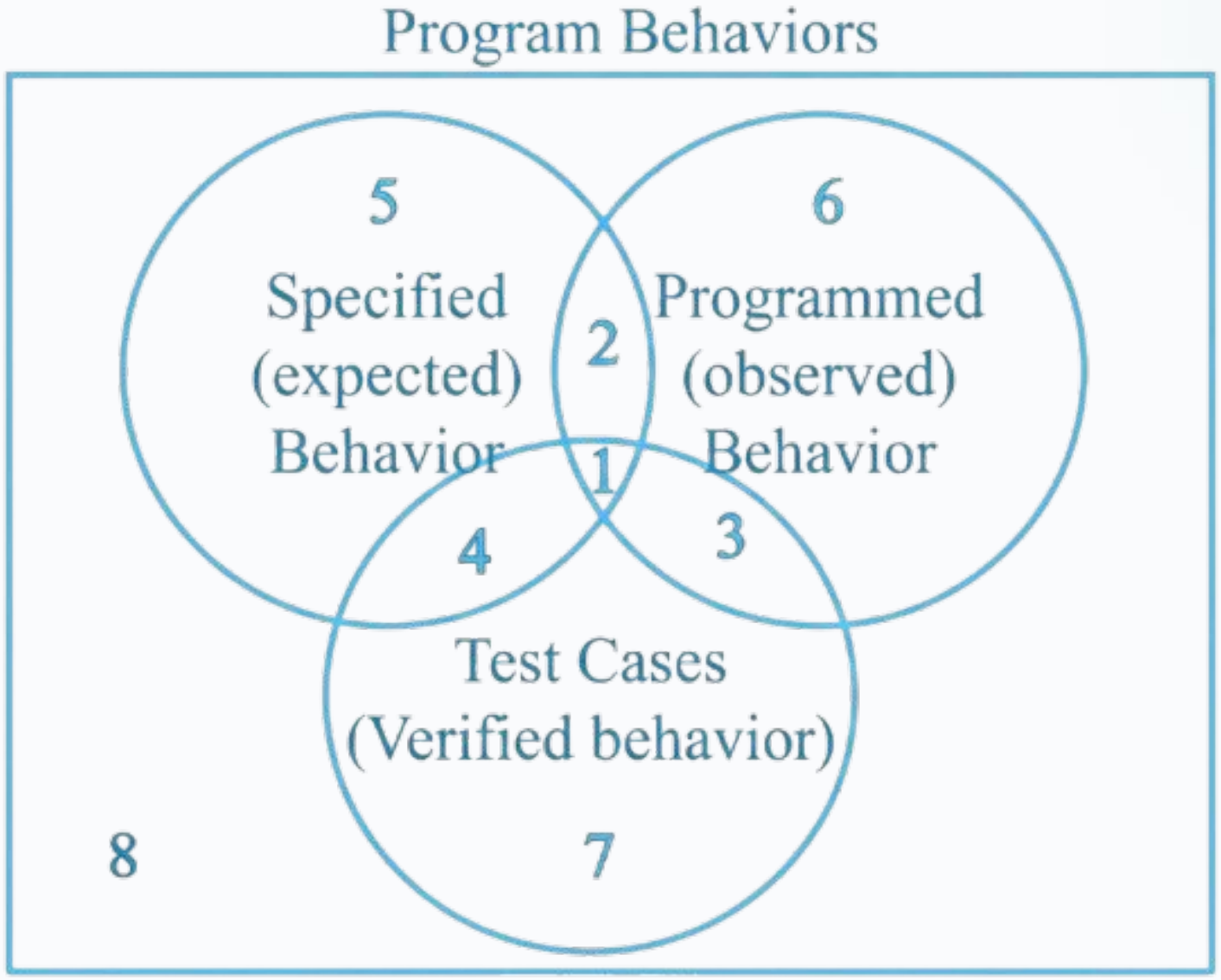
- **Unit level:** Isolating individual classes or functions.
- **Integration level:** Testing interface points between modules.
- **System level:** Fully compiled, end-to-end flow evaluation.

Methodology Level

Classification mostly applied at the unit level based on strategies:

- **Black box (Functional) Testing:** Focuses purely on inputs and outputs.
- **White box (Structural) Testing:** Looks into programmatic execution paths.

Testing wrt Behavior



Testing Methodologies

Functional (Black Box)

Inspects **specified behavior**.

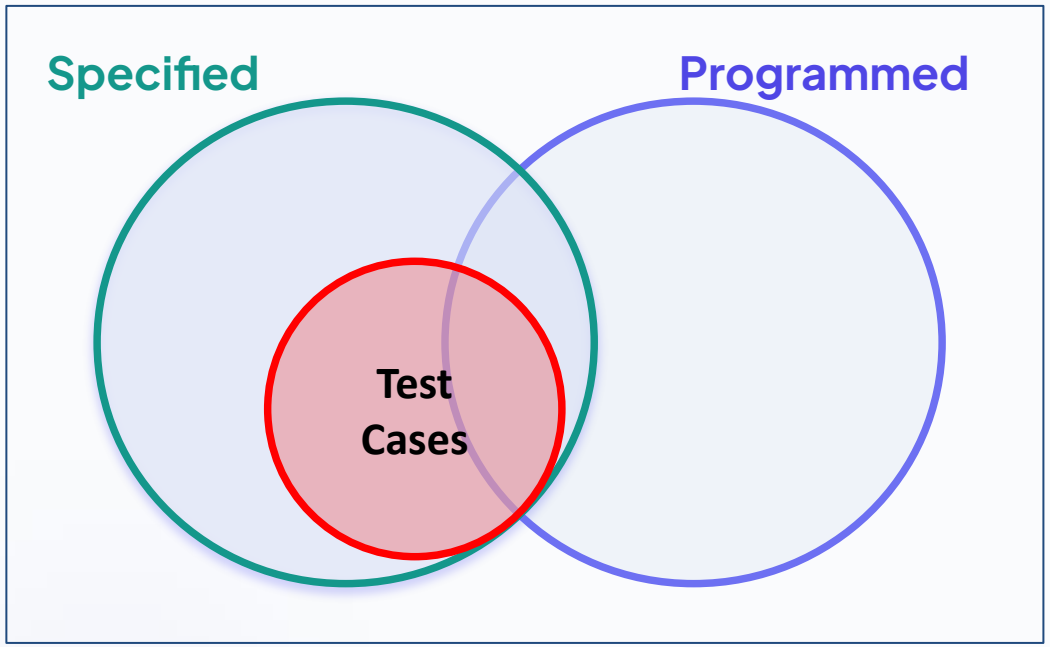
Tests are designed strictly from the requirements documentation without any knowledge of internal code statements, paths, or loops.

Structural (White Box)

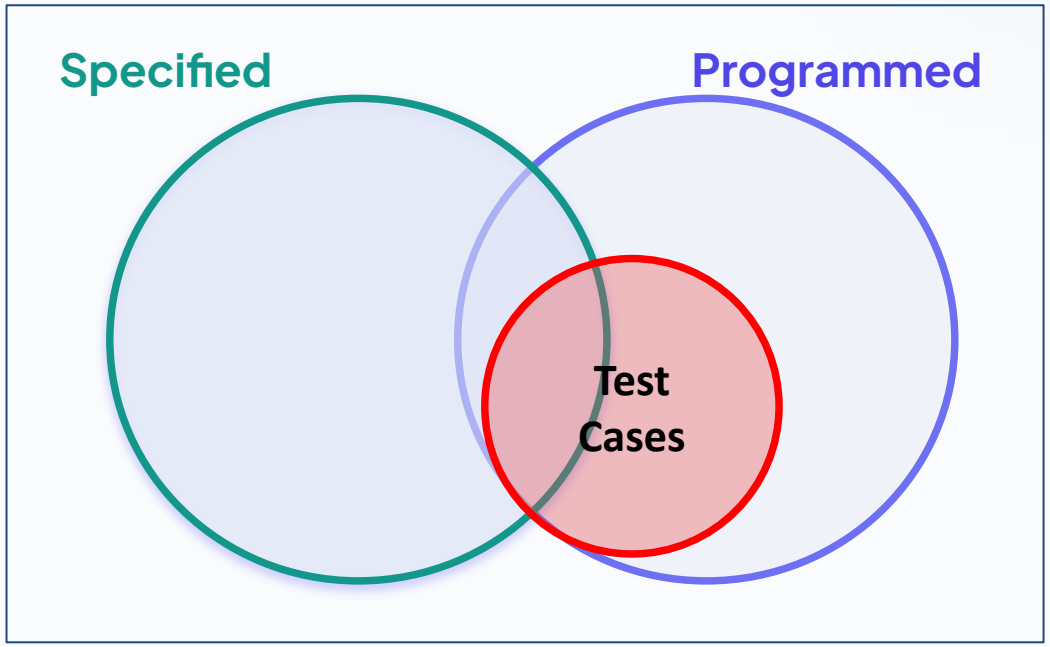
Inspects **programmed behavior**.

Tests are crafted with complete transparency of internal system logic, prioritizing statement coverage and decision path evaluation.

Functional Test Cases

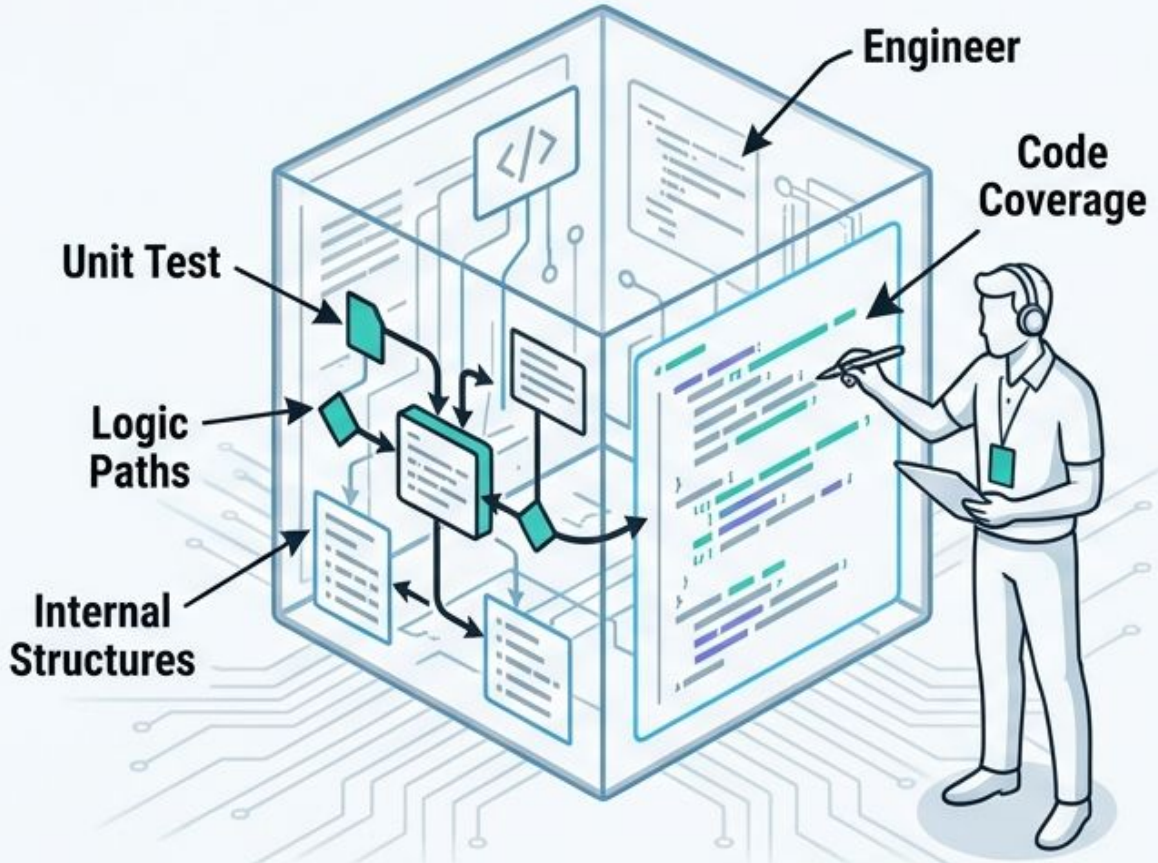


Structural Test Cases



UNDERSTANDING SOFTWARE TESTING: WHITE BOX VS. BLACK BOX

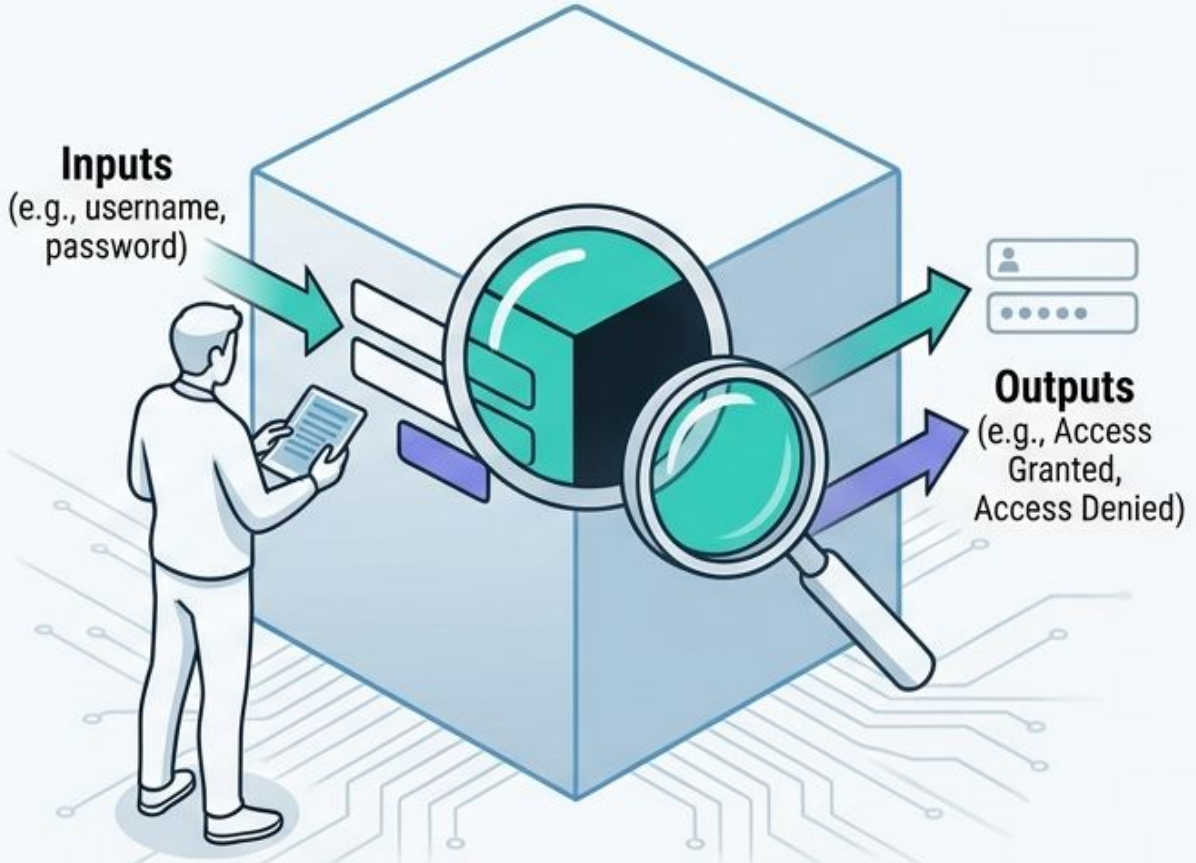
WHITE BOX TESTING



INTERNAL STRUCTURE IS KNOWN.
Focuses on code, branches, and logic.
Test cases derived from internal design.

Developer/Tester

BLACK BOX TESTING



INTERNAL STRUCTURE IS UNKNOWN.
Focuses on functionality and user requirements.
Test cases derived from external specifications.

QA Analyst/End User

COMMON GOAL: ENSURE SOFTWARE QUALITY

White-Box Testing

> *Our goal is to ensure that all statements and conditions have been executed at least once*

White-box testing relies on looking straight into the internal code loops. It guarantees logical execution path safety, forcing every conditional branch to evaluate at least once during unit testing phases.

Black-Box Workflow

1. Requirements

System requirements documents are evaluated to establish the target expected functional boundaries.

2. Input

Specific inputs are carefully selected to trigger logical branches in the target system interface.

3. Events

System actions and process scenarios are triggered securely within the black-box interface.

4. Output

Observable outputs are captured and systematically validated against specified outcomes.

Black-Box Testing

- How is functional validity tested?
- How is system behavior and performance tested?
- What classes of input will make good test cases?
- Is the system particularly sensitive to certain input values?
- How are the boundaries of a data class isolated?
- What data rates and data volume can the system tolerate?
- What effect will specific combinations of data have on system operation?

Consequences of Bugs

Bug Categories:

Function-related bugs, system-related bugs, data bugs, coding bugs, design bugs, documentation bugs, standards violations, and other critical compliance issues.

As bug classification scales upward toward systemic and infectious levels, logical risk and downstream corrective development costs rise exponentially.