# Lab 3: Three Boundary Value Analysis Techniques

Objective:

- Understand and implement Boundary Value Analysis (BVA).
- Apply Boundary Value Check, Robust Testing, and Worst-Case Testing techniques.
- Design test cases to ensure coverage of edge conditions for given inputs.

Theory:

Boundary Value Analysis (BVA)

Boundary Value Analysis is a black-box test design technique based on testing at boundaries between partitions. This is due to the observation that more errors occur at the boundaries of input domains.

Let the input domain be between a and b (inclusive), then:

- The normal boundaries are: a, a+1, b-1, and b.
- The out-of-boundary values are: a−1, b+1.

Techniques Overview

| Technique | Description |
|---|---|
| Boundary Value Check | Focuses on valid boundaries: min, min+1, max-1, max |
| Robust Testing | Includes both valid and invalid values: min−1, min, min+1, max−1, max, max+1 |
| Worst Case Testing | Tests all combinations of min, min+1, max−1, max (for each variable) |

Equipment/Software Required:

- Computer with Python/Java/C++ (any preferred language)
- Any IDE or text editor
- Compiler/Interpreter

Lab Task:

Problem Statement

Assume a software system that accepts the age of a user in the range 18 to 60. Design and execute test cases using:

- Boundary Value Check
- Robust Testing
- Worst Case Testing

Implementation

Boundary Value Check

Test Cases:

| Test Case ID | Input | Expected Result |
|---|---|---|
| BVC1 | 18 | Valid |
| BVC2 | 19 | Valid |
| BVC3 | 59 | Valid |
| BVC4 | 60 | Valid |

Robust Testing

Test Cases:

| Test Case ID | Input | Expected Result |
|---|---|---|
| RT1 | 17 | Invalid (Below Min) |
| RT2 | 18 | Valid |
| RT3 | 19 | Valid |
| RT4 | 59 | Valid |
| RT5 | 60 | Valid |
| RT6 | 61 | Invalid (Above Max) |

Worst Case Testing

Assume a system with two input variables: Age (18–60) and Experience (0–40).

Boundary Values:

- Age: 18 (min), 19 (min+1), 59 (max−1), 60 (max)
- Experience: 0 (min), 1 (min+1), 39 (max−1), 40 (max)

Total Test Cases:

4 (Age) × 4 (Experience) = 16 combinations

Sample Test Cases:

| Test Case ID | Age | Experience | Expected Result |
|---|---|---|---|
| WC1 | 18 | 0 | Valid |
| WC2 | 18 | 1 | Valid |
| WC3 | 18 | 39 | Valid |
| WC4 | 18 | 40 | Valid |
| WC5 | 19 | 0 | Valid |
| ... | ... | ... | ... |
| WC16 | 60 | 40 | Valid |

Sample Code (Python)

```
def is_valid_age(age):
    return 18 <= age <= 60

def is_valid_experience(exp):
    return 0 <= exp <= 40

def test_boundary_value_check():
    test_ages = [18, 19, 59, 60]

    for age in test_ages:
        print(f"Age {age} is", "Valid" if is_valid_age(age) else "Invalid")

def test_robust_testing():
    test_ages = [17, 18, 19, 59, 60, 61]

    for age in test_ages:
        print(f"Age {age} is", "Valid" if is_valid_age(age) else "Invalid")

def test_worst_case():
    ages = [18, 19, 59, 60]
    experiences = [0, 1, 39, 40]

    for age in ages:
        for exp in experiences:
            print(f"Age: {age}, Experience: {exp} =>",
                  "Valid" if is_valid_age(age) and is_valid_experience(exp)
else "Invalid")

# Run Tests
print("Boundary Value Check:")
test_boundary_value_check()

print("\nRobust Testing:")
test_robust_testing()

print("\nWorst Case Testing:")
test_worst_case()
```

Result:

- Implemented and verified the three boundary analysis techniques.
- Observed how edge values and combinations affect the behavior of software input validation.

Viva Questions:

1. What is Boundary Value Analysis?
2. Why are boundaries important in test case design?
3. How does robust testing improve test coverage?
4. What's the difference between robust and worst-case testing?

5. Can you apply BVA for non-numeric data? How?

Conclusion:

Through this lab, students gained hands-on experience in:

- Designing effective boundary test cases.
- Understanding how to uncover hidden bugs at edges of input domains.
- Applying systematic testing strategies for robust software development.