Software Testing and Quality Assurance

Module-3

Control Flow Testing

By Prof. Pallavi Mahajan

Contains

- Data Flow Testing
- Data Flow Anomaly
- Overview of Dynamic Data Flow Testing,
- Data Flow Graph,
- Data Flow Terms,
- Data Flow Testing Criteria,
- Comparison of Data Flow Test Selection Criteria,
- Feasible Paths and Test Selection Criteria,
- Comparison of Testing Techniques.

Data Flow Testing

Data Flow Testing

Introduction to Data Flow Testing

- Definition: Examines the flow of data values between variables during execution.
- Purpose: Ensures proper handling of variable assignments and uses.
- Key Focus:
- Data assignment and usage verification.
- Preventing incorrect memory accesses.
- Ensuring correct computation results.

Data Flow Testing

Motivation for Data Flow Testing

- Detects unverified variable assignments.
- Identifies variables assigned twice without intermediate use.
- Ensures correctness of variable values across execution paths.

Concept of Data Flow Testing

- Data values flow between variables along program execution paths.
- Ensures variables are properly assigned, stored, and used.
- Example: Opening a file and later using its file pointer correctly.

Data Flow Testing

Levels of Data Flow Testing

Static Data Flow Testing

- Conducted without executing the program.
- Identifies potential defects through code analysis.

Dynamic Data Flow Testing

- Focuses on identifying program paths based on test criteria.
- Involves actual program execution.

Data Flow Testing

Key Differences Between Control Flow and Data Flow Testing

Similarities:

- Both identify program paths.
- Both emphasize generating test cases from program paths.

Differences:

- Control flow testing uses control-based test selection criteria.
- Data flow testing uses data-based test selection criteria.

Data Flow Testing

What is Data Flow Anomalies?

- Defined as potential program defects detected through static analysis.
- Common anomalies:
- Using a variable before assignment.
- Assigning a variable twice without intermediate use.

Data Flow Testing

Static vs. Dynamic Data Flow Testing

Static Testing:

- Identifies anomalies without executing code.
- Useful for early defect detection.

- ***Dynamic Testing:***
- Focuses on execution-based validation.
- Ensures test cases cover important data interactions.

Data Flow Testing

Example of Data Flow Testing

- Case study of a function with multiple variable assignments.
- Identifies redundant or missing assignments.
- Ensures correctness of data propagation.

Data Flow Testing

Data Flow Testing Techniques

- Definition-Use (DU) Chains:
- Tracks variable definitions and uses across execution paths.
- Variable Anomaly Detection:
- Identifies improper variable usage (e.g., use before assignment).

Data Flow Anomalies

Data Flow Anomalies

Data Flow Anomalies

- Defined as potential program defects detected through static analysis.
- ***Common anomalies:***
  - Using a variable before assignment.
  - Assigning a variable twice without intermediate use.
  - Defining a variable but never using it.

Data Flow Anomalies

Types of Data Flow Anomalies

- **Type 1: Defined and Then Defined Again (dd)**
- **Example: Assigning a variable twice without using the first value.**
- **Possible reasons:**
- **Redundant computation.**
- **Incorrect assignment.**
- **Missing intermediate statement.**

Data Flow Anomalies

Undefined but Referenced (ur)

- **Example: Using a variable before assigning a value.**
- **Possible reasons:**
- **Programmer intended to use a different initialized variable.**
- **Missed initialization of the variable.**

Data Flow Anomalies

Defined but Not Referenced (du)

- **Example: Assigning a variable but never using it.**
- **Indicates an unnecessary computation or a missing usage statement.**

Data Flow Anomalies

State Transitions in Data Flow

Variable States:

- U: Undefined
- D: Defined but not referenced
- R: Defined and referenced
- A: Abnormal

State Transitions:

- Define (d)
- Reference (r)
- Undefine (u)

Abnormal Transitions:

- dd (Redefining a variable before use)
- ur (Using an undefined variable)
- du (Defining a variable but never using it)

Data Flow Anomalies

Detecting Data Flow Anomalies

Program Instrumentation:

- **Adding extra code to monitor variable states during execution.**
- **Identifies occurrences of dd, ur, and du anomalies.**

Importance:

- **Prevents potential programming errors.**
- **Helps improve code clarity and maintainability.**

Data Flow Anomalies

Example of Data Flow Anomalies in Code

- **Example 1: Using an uninitialized variable in a computation.**
- **Example 2: Assigning a variable twice without usage in between.**
- **Example 3: Assigning a variable but never using it.**

Data Flow Anomalies

Key Points of Data Flow Anomaly Detection

- Data Flow Anomalies indicate potential errors but may not always cause failures.

Key anomaly types:

- dd (Defined then Defined Again)
- ur (Undefined but Referenced)
- du (Defined but Not Referenced)

Instrumentation techniques help in early detection.

Overview of Dynamic Data Flow Testing

Overview of Dynamic Data Flow Testing

Definition: Ensures that variables are assigned correct values and used properly.

Key Principles:

- Variables must be initialized before use.
- Data definitions should be followed by valid usages.
- Control flow should reflect the intended computational logic.

Process:

- Draw a data flow graph from the program.
- Select one or more data flow testing criteria.
- Identify paths in the data flow graph satisfying the criteria.
- Derive and solve path predicate expressions for test inputs.

Thank You..