CEN 5070 – Software Verification & Validation

Lecture 3 – "Software Unit Test Coverage and Adequacy" by Zhu et al.

1.1 Objectives

- Cover sections 1-2 of this paper
- Concept of test adequacy – how much testing is enough?
- Categories of test adequacy criteria
- Structural testing (white-box)
- Control-flow based adequacy
- Data-flow based adequacy
- To prepare student for hands-on exercises

1.2 Lecture

1. INTRODUCTION

- Test criterion – what constitutes an adequate test
  - Testing shows the presence of bugs, not their absence [Dijkstra 1972]
  - A test case that finds no error is useless (?)
  - Adequacy criterion ⇒ test case design process
  - Example. Decision Table: 1 test case for each column” is an adequacy criterion for functional testing [Jones 2000]
- Research has identified many criteria and ways to measure satisfaction of criteria
  - Statement coverage – test set contains test cases that, collectively, cause every statement to be executed at least once during testing. Coverage is a metric (measurement) of the percentage of the statements executed by the test set.
  - Mutation adequacy [DeMillo 1978] – measures the ability of the test set to distinguish small seeded faults. A mutant, \( P' = P + \{ \text{seeded fault} \} \). Apply test set T to both P and P’ [notation: R=\( P(T) \) and R’=\( P'(T) \)]. If R=R’, test set T does not adequately distinguish P and P’: mutant \( P' \) is alive. Otherwise, mutant \( P' \) is killed by T. NOTE: Mutation provides a measure of the goodness of the test set.
  - Adequacy criterion: what properties of a program must be exercised to constitute a thorough test, i.e., inspires confidence that no errors were undetected.
    - Reliability of a test criteria ⇒ consistency. When \( T_1 \) and \( T_2 \) satisfy criterion C, success of \( P(T_1) \) ⇒ success of \( P(T_2) \).
    - Validity of a test criteria ⇒ meaningfulness. Satisfying the criterion ensures the ability to detect faults. For fault f in P, there exists a test set \( T_k \) satisfying criterion C, such that \( P(T_k) \) reveals the fault.
- These theoretical criteria are not practical. There is no universal way to compute or prove whether reliability or validity holds.
In a **practical sense**, an adequacy criterion serve as a **stopping rule**, and as a **testing measure** of the degree of success toward satisfying the criterion.

Adequacy criterion also has practical use
- Test case selection criterion => influences design of test cases. Given P, S, C, here is how to **generate an adequate test set**
- Test data adequacy criterion => **acceptance process** for a test set. "Does this test set satisfy the criteria?" Example: Given decision table (specification) and test set T, is T adequate wrt the adequacy criterion "One test case for each column in the decision table?"
- Criteria define **measurement requirements**. Program instrumentation required for white-box testing, so that test data adequacy can be measured.
- When criterion C not met by a given test set T, the criterion provides **guidance** for adding specific test cases to T that meet the criterion.

Comparison of adequacy criteria
- Which has better fault detection capability?
- Test case generation effort, automation, test set size?
- Is there a ranking of adequacy criteria (**subsumes relation**)?

Categories of Test Data Adequacy Criteria
- By source of information used for formulating test cases
  - Specification-based (black-box)
  - Program-based (white-box)
  - Interface-based (input domain driven; random/statistical testing)
- By the underlying testing approach
  - Structural – coverage of elements in the structure of program/spec
  - Fault-based – effort directed at detecting faults
  - Error-based – test cases focus on historical patterns of human errors (fault deposits)

### 2. **STRUCTURAL TESTING**

<table>
<thead>
<tr>
<th>Program Based</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control flow model of program: G=(N,E), N is set of nodes, E:NxN is set of edges.</td>
</tr>
<tr>
<td>Variations in how graph drawn.</td>
</tr>
<tr>
<td>Node = sequential statement terminated by branch</td>
</tr>
<tr>
<td>Edge represents a branch or sequential flow</td>
</tr>
<tr>
<td>Edge may be labeled (as in paper) to reflect branch condition</td>
</tr>
<tr>
<td>Statement coverage =&gt; every node reached during testing</td>
</tr>
<tr>
<td>A set P of execution paths satisfies the statement coverage criterion for each node n in flow graph G, there is at least one path p in P such that node n is on path p.</td>
</tr>
<tr>
<td>Infeasible node =&gt; dead code =&gt; 100% coverage not always possible</td>
</tr>
<tr>
<td>Dead code is detectable from the CFG</td>
</tr>
<tr>
<td>Branch coverage =&gt; each edge is traversed during testing</td>
</tr>
<tr>
<td>Branch coverage =&gt; statement coverage</td>
</tr>
</tbody>
</table>

Branch coverage **subsumes** statement coverage
- Infeasible branches – logically impossible branches
- Example: if (1) { xxx} else { yyy}
- Infeasible branches not detectable from the CFG!!
- Path coverage => all combinations of branches during program executions
- Impossibility of testing when #paths is infinite
- **Finite applicability requirement** – adequacy criterion can be satisfied by a finite test set
- Infeasible nodes/branches violate finite applicability
- HARD PROBLEM: Whether a program statement is feasible is undecidable!!
- We approximate finite applicability by reducing/eliminating redundancy
  - Justification for loop testing approaches
  - Simple path => no repeated edges
  - Elementary path => no repeated nodes
  - Basis path testing [McCabe]: Include another path when it is not the combination of paths already covered. McCabe metric => #basis paths
- Methods based on program text
  - count-K criterion for loops: repeat 0, 1, 2, ..., K times.
  - Multiple condition coverage criterion
    - Example: if (x>5 && y<2) { xxx}
      - **Condition**: (p && q), where p = (x>5), q = (y<2)
    - Decision coverage => each condition evaluates to TRUE and to FALSE
      - Satisfied by TT and FT outcomes of p and q above
    - Condition coverage => each elementary predicate evaluates to T and to F
      - p must evaluate to both T and F (as show above)
      - q must evaluate to both T and F (NOT shown above)
      - **NOTE**: no attention is paid to the BRANCH (a graph-based criterion)
    - Multiple condition coverage => must consider all (feasible) combinations of p and q: TT, TF, FT, FF
      - **Not every combination is FEASIBLE**
- **Linear code sequence and jump** (LCSAJ) coverage criteria
  - Block = linear sequence terminated by a branch (jump)
  - Coverage measure is termed **test effectiveness ration** (TER)
  - TER = (#LCSAJ exercised) / (Total #LCSAJ)
  - STRENGTH: statement < branch < LCSAJ
  - Homework: Find a definitive paper on LCSAJ and a tool that supports the technique.

Assigned: Exercise Set #3 (unit testing with application of test adequacy criteria + Text Readings)