Test Adequacy Assessment using Control Flow and Data Flow

What is Test Adequacy?

- Context
  - Consider a program $P$ written to meet a set $R$ of $n$ functional requirements.
  - Suppose now that a set $T$ containing $k$ tests has been constructed to test $P$ to determine whether or not it meets all the requirements in $R$.
  - Also, $P$ has been executed against each test in $T$ and has produced correct behavior.

- Question
  - We now ask: Is $T$ good enough? i.e., Has $P$ been tested thoroughly?, or Is $T$ adequate?

Measurement of adequacy

- Adequacy is measured for a given test set designed to test $P$ to determine whether or not $P$ meets its requirements.
- This measurement is done against a given criterion $C$.
  - A test set is considered adequate with respect to criterion $C$ when it satisfies $C$.
  - The determination of whether or not a test set $T$ for program $P$ satisfies criterion $C$ depends on the criterion itself and is explained later.

Example

- Program `sumProduct` must meet the following requirements:
  - R1: Input two integers, say $x$ and $y$, from the standard input device.
  - R2.1 Find and print to the standard output device the sum of $x$ and $y$, if $x < y$.
  - R2.2 Find and print to the standard output device the product of $a$ and $y$, if $x \geq y$. 
Example (cont.)

- Suppose now that the test adequacy criterion $C$ is specified as:
  - $C: A$ test $T$ for program $(P, R)$ is considered adequate if for each requirement $r$ in $R$ there is at least one test case in $T$ that tests the correctness of $P$ with respect to $r$.
  - Obviously, $T=\{t: <x=2, y=3>\}$ is inadequate with respect to $C$ for program `sumProduct`.
  - The lone test case $t$ in $T$ tests $R1$ and $R2.1$, but not $R2.2$.

Test Coverage

- Consider a test suite $T$. Given that $C_e$ has $n \geq 0$ elements, we say that $T$ covers $C_e$ if for each element $e' \in C_e$ there is 1 test case in $T$ that tests $e'$.
  - $T$ is considered adequate with respect to $C$ if it covers all elements in $C_e$.
  - $T$ is considered inadequate with respect to $C$ if it covers $k$ elements of $C_e$ where $k < n$.
  - The fraction $k/n$ is a measure of the adequacy level of $T$ with respect to $C$. This fraction is also known as the coverage of $T$ with respect to $C, P$, and $R$.

Measurement of Test Adequacy

- Measured against a finite set of elements, known as coverage domain, derived from:
  - Requirements, or
  - Program under test

- A criterion $C$ is a white-box test adequacy criteria if the coverage domain $C_e$ depends solely on the program under test.
- A criterion $C$ is a black-box test adequacy criteria if the coverage domain $C_e$ depends solely on requirements of the program under test.

Example

- Let us again consider the following criterion:
  - "A test $T$ for program $(P, R)$ is considered adequate if for each requirement $r$ in $R$ there is at least one test case in $T$ that tests the correctness of $P$ with respect to $r$.”
  - $C_e = \{R1, R2.1, R2.2\}$.
  - $T$ covers $R1$ and $R2.1$ but not $R2.2$. Hence $T$ is not adequate with respect to $C$. The coverage of $T$ with respect to $C, P$, and $R$ is $0.66$.

Example (cont.)

- Let us again consider the following criterion:
  - "A test $T$ for program $(P, R)$ is considered adequate if for each requirement $r$ in $R$ there is at least one test case in $T$ that tests the correctness of $P$ with respect to $r$.”
  - $C_e = \{R1, R2.1, R2.2\}$.
  - $T$ covers $R1$ and $R2.1$ but not $R2.2$. Hence $T$ is not adequate with respect to $C$. The coverage of $T$ with respect to $C, P$, and $R$ is $0.66$.

Example (cont.)

- Let us again consider the following criterion:
  - "A test $T$ for program $(P, R)$ is considered adequate if for each requirement $r$ in $R$ there is at least one test case in $T$ that tests the correctness of $P$ with respect to $r$.”
  - $C_e = \{R1, R2.1, R2.2\}$.
  - $T$ covers $R1$ and $R2.1$ but not $R2.2$. Hence $T$ is not adequate with respect to $C$. The coverage of $T$ with respect to $C, P$, and $R$ is $0.66$.
Let us consider another criterion (path coverage).

A test $T$ for program $(P, R)$ is considered adequate if each path in $P$ is traversed at least once.

Assume that $P$ has exactly two paths, one $(p_1)$ corresponding to condition $x < y$ and the other $(p_2)$ to $x \geq y$. Then, the coverage domain $C_e$ is the set $\{p_1, p_2\}$.

To measure the adequacy of $T$ against $C$, we execute $P$ against each test case in $T$.

$T$ contains only one test for which $x < y$, so only the path $p_1$ is executed.

Thus, the coverage of $T$ with respect to $C, P,$ and $R$ is 0.5 and hence $T$ is not adequate with respect to $C$.

Example

Consider the following $R, P,$ and $T$.

- **sumProduct1**
  
  ```
  begin
  int x, y;
  input(x, y);
  sum=x+y;
  output(sum);
  end
  ```

  $T=\{t: <x=2, y=3>\}$

  This program is obviously incorrect as per the requirements of sumProduct.

  There is only one path denoted as $p_1$ that traverses all the statements.

  Using the path-based coverage criterion $C$, we get coverage domain $C_e=\{p_1\}$. $T$ is adequate w.r.t. $C$, but does not reveal the error.

Example (contd.)

The program sumProduct2 is correct as per the requirements of sumProduct. It has two paths denoted by $p_1$ and $p_2$.

$C_e=\{p_1, p_2\}$. $T=\{t: <x=2, y=3>\}$ is inadequate w.r.t. the path-based coverage criterion $C$.

- **sumProduct2**
  
  ```
  begin
  int x, y;
  input(x, y);
  if(x<y)
  then
  output(x+y);
  else
  output(x*y);
  end
  ```

  $R_1$: Input two integers, say $x$ and $y$, from the standard input device.
  $R_2.1$ Find and print to the standard output device the sum of $x$ and $y$, if $x < y$.
  $R_2.2$ Find and print to the standard output device the product of $x$ and $y$, if $x \geq y$.
An adequate test set might not reveal even the most obvious error in a program.

However, measuring test adequacy is still needed, as increasing coverage help reveal errors!

Test Enhancement

- A test set adequate with respect to some criterion does not guarantee an error-free program.
- However, an inadequate test set with respect to any criterion often implies deficiency.
- Identifying this deficiency can help enhance test set.
- Enhancement is likely to be achieved by testing a program in ways it has not been tested before
  - testing untested portion
  - testing features in a sequence different from the one used previously.
- Testing the program differently than before will raise the chances of discovering uncovered errors.

Test Enhancement: Example

- Consider the example sumProduct2.
- We can add a new test case to make $T$ adequate with respect to the path coverage criterion.
  \[
  T' = \{< x = 3, y = 4 >, < x = 3, y = 1 >\}
  \]
  ```plaintext
sumProduct2
begin
  int x, y;
  input (x, y);
  if(x<y)
  then
    output(x+y);
  else
    output(x*y);
end
  ```

Test Enhancement: Procedure
Consider a program intended to compute $x^y$ given integers $x$ and $y$. For $y < 0$ the program skips the computation and outputs a suitable error message.

```c
1 begin
2 int x, y;
3 int product, count;
4 input (x, y);
5 if(y<0) {
6 product=1; count=-y;
7 while(count>0) {
8 product=product*x;
9 count=count-1;
10 }
11 output(product);
12 }
13 else
14 output ("Input does not match its specification.");
15 end
```

Consider a test adequacy criterion $C$.

- Test $T$ is considered adequate if it tests the program for at least one zero and one non-zero value of each of the two inputs $x$ and $y$.

The coverage domain for $C$.

- Without any inspection of the program, we get $C_e = \{x=0, y=0, x=0, y \neq 0\}$, and can derive an adequate test set.

$$T = \{ \langle x=0, y=1 \rangle, \langle x=1, y=0 \rangle \}$$

- **Black-box coverage criterion; Any Issues?**

Test Enhancement: Example (cont.)

Consider the path coverage criterion

- Size of the coverage domain for $C$?
  - Given that $y$ is any non-negative integer, the number of paths can be arbitrarily large due to the while loop. -- infinite domain size

- Simplified path coverage criterion
  - A test set $T$ is considered adequate if it tests all paths. In case the program contains a loop, then it is adequate to traverse the loop body zero times and once.

The simplified path coverage criterion leads to $C'_e = \{p_1, p_2, p_3\}$.

$$p_1: [1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 7 \rightarrow 9]$$
$$p_2: [1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 5 \rightarrow 7 \rightarrow 9]$$
$$p_3: [1 \rightarrow 2 \rightarrow 3 \rightarrow 8 \rightarrow 9]$$
Test Enhancement: Example (cont.)

1. \( T = \{< x = 0, y = 1 >, < x = 1, y = 0 >\} \)

Measure the coverage of \( T \) with respect to \( C' \).

2. \( T \) does not contain any test with \( y < 0 \) and \( p_3 \) remains uncovered.

3. Thus the coverage is \( \frac{2}{3} = 0.66 \).

Test Enhancement: Example (cont.)

\[ T = \{< x = 0, y = 1 >, < x = 1, y = 0 >, < x = 5, y = -1 >\} \]

Any test case with \( y < 0 \) will cause \( p_3 \) to be traversed.

Infeasibility and test adequacy

- An element of the coverage domain is **infeasible** if it cannot be covered by any test in the input domain of the program under test.

- There is no generic algorithm that would analyze a given program and determine if a given element in the coverage domain is infeasible or not.

- Thus it is usually the tester who determines whether or not an element of the coverage domain is infeasible.

Demonstrating feasibility

- **Feasibility can be demonstrated** by executing the program under test against a test case and showing that indeed the element under consideration is covered.

- **Infeasibility cannot be demonstrated** by program execution against a finite number of test cases.
Infeasible Path: Example

This program inputs two integers $x$ and $y$ and computes $z$. $C_{e} = \{p_{1}, p_{2}, p_{3}\}$.

- $p_{1}$: [$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 8 \rightarrow 9$]
- $p_{2}$: [$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 8 \rightarrow 9$]
- $p_{3}$: [$1 \rightarrow 2 \rightarrow 3 \rightarrow 7 \rightarrow 8 \rightarrow 9$]

Adequacy and Infeasibility

- A test is considered adequate when all feasible elements in the domain have been covered.
- Prior to test enhancement, a tester usually does not know which elements of a coverage domain are infeasible. It is only during an attempt to construct a test case to cover an element that one might realize the infeasibility of an element.
- Unfortunately, there is no automatic way to identify all infeasible elements in a coverage domain.

Error Detection and Test Enhancement

- The purpose of test enhancement is to determine test cases that test the untested parts of a program or exercise the program using uncovered portions of the input domain.
- Even the most carefully designed tests based exclusively on requirements can be enhanced.
- The more complex the set of requirements, the more likely it is that a test set designed using requirements is inadequate with respect to even the simplest of various test adequacy criteria.
Example (cont.)

- Consider the following program written to meet the above requirements.

```c
begin
int x, y, product, request;
define exp=1
define exit=3
get_request (request); // Get (one of three possibilities).
product=1; // Initialize prod
while (request != exit) {
  if (request == exp) {
    input(x, y); count=y;
    while (count > 0) {
      product-product * x; count=count-1;
    }
  } // End of processing the "exponentiation" request.
  else if (request == fact) {
    input(x); count=x;
    while (count > 0) {
      product-product * count; count=count-1;
    }
  } // End of processing the "factorial" request.
  else if (request == exit) {
    output("Thanks for using this program. Bye!");
    break; // Exit the loop.
  } // End of if.
  output(product); // Output the value of exponential or factorial.
get_request (request); // Get user request once again and jump to loop begin.
}
end
```

Example (contd.)

- Assume the following test set derived from requirements review.

\[ T = \{ t_1: <request = 1, x=2, y=3, request = 3 > \} \]

\[ t_2: <request = 2, x=4, request = 3 > \]  

\[ t_3: <request = 3 > \]

- Execute the program against all tests.

is the program behavior correct for all tests?

Example (contd.)

- Assume the path coverage criterion.

A test set \( T \) is considered adequate if it tests all paths. In case the program contains a loop, then it is adequate to traverse the loop body zero times and once.

- A tricky path

Consider the path \( p \) that begins execution at line 1, reaches the outermost while at line 10, then the first if at line 12, followed by the statements that compute the factorial starting at line 20, and then the code to compute the exponential starting at line 13.

To cover the path \( p \), we need to add a new test.

\[ T' = \{ <request = 2, x=4 >, <request = 1, x=2, y=3 >, <request = 3 > \} \]

What is wrong?
Statement and Block Coverage

- Recall control flow graph
  - Basic block
    - A sequence of consecutive statements that has exactly one entry point and one exit point.

- Statement coverage
  - The statement coverage of $T$ with respect to $(P, R)$ is computed as $S_c/(S_e-S_i)$ where
    - $S_c$ is the number of statements covered
    - $S_i$ is the number of unreachable statements, and
    - $S_e$ is the total number of statements in the program
  - $T$ is considered adequate with respect to the statement coverage criterion if the statement coverage of $T$ with respect to $(P, R)$ is 1.

Example: Statement Coverage

- Coverage domain
  - $S_e$={$2, 3, 4, 5, 6, 7b, 9, 10$}
  - Let $T_1$={$t1:<x=-1, y=-1>, t2:<x=1, y=1>$}

- Statements covered:
  - $t1$: $2, 3, 4, 5, 6, 7, and 10$
  - $t2$: $2, 3, 4, 9, and 10$
  - $S_c=8, S_i=1, S_e=9$.

- The statement coverage for $T$ is $8/(9-1)=1$. Hence $T_1$ is adequate for $(P, R)$ with respect to the statement coverage criterion.

Example: Block Coverage

- Coverage domain
  - $B_e$={$1, 2, 3, 4, 5$}
  - Assume test cases
    - $T_2$={$t1: \begin{cases} l_1 & x \leq 0; \\ l_2: \begin{cases} x<0 \quad & \quad y>-3 \quad ; \\ y\leq0 \quad & \quad \text{false} \end{cases} \\ \end{cases}$
    - $B_c=3/(5-1)=0.75$.
    - $T_2$ is not adequate for $(P, R)$ with respect to the block coverage criterion.

- Blocks covered
  - $t1$: 1, 2, 5
  - $t2, t3$: same to $t1$
  - Block coverage for $T_2$: 3/(5-1)=0.75.
  - Hence $T_2$ is not adequate for $(P, R)$ with respect to the block coverage criterion.

©Aditya P. Mathur 2009