Functional Testing

Black-box testing: Recap

- White-box testing techniques done so far:
  - Graphs and graphs coverage criteria as applied to code and design sequencing constraints.
  - Logic coverage criteria as applied to code and design models as FSMs.
  - Symbolic testing.
- Black-box testing involves testing a code/design without knowledge about its internals (like the actual code structure, statement, design details etc.)
  - Deals only with inputs and outputs applied to code/design/requirements.

Functional Testing

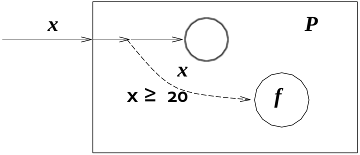
- The term functional testing was introduced by W. E. Howden in the late 1970s.
- A program P is viewed as a function transforming inputs to outputs. Given inputs x_i, P computes outputs y_i such that $y_i = P(x_i)$.
- Examples:
  - A sorting program: the input x_i is an array of numbers and the output y_i is the array in sorted order.
  - A compiler: the input is a program and the output is the resulting object code.

Functional testing: Important steps

1. Precisely identify the *domain* of each input and each output variable.
2. Select values from the data domain of each variable having *important properties*.
3. Consider *combinations* of special values from different input domains to design test cases.
4. Consider input values such that the program under test produces *special* values from the domains of the output variables.

Testing a function in context

- Even though functional testing is a black-box testing technique, sometimes, we need to know minimal *context* information to get relevant values for inputs and outputs.
- Consider a program P and a function f in P as shown in the figure below.



Testing a function in context, contd.

- Suppose x is a floating point variable and we are unaware of the predicate $x \geq 20$.
- We are likely to test for the following input values for x: $x = +k, x = -k, x = 0$, where k is a number with a large magnitude.
- The function f will be invoked just once, for $x = +k$.
- The valid range for input x with respect to testing f will be $x = k$ where k is a number much larger than 20, $x = y$, where $20 < y < k$ and $x = 20$.

Types of functional testing

- Equivalence class partitioning
- Boundary value analysis
- Decision tables
- Random testing
- Pair-wise testing: Orthogonal arrays
- Cause-effect diagram

Equivalence class partitioning

- If the input domain is too large for all its elements to be used as test cases, the input domain is partitioned into a finite number of sub-domains for selecting test inputs.
- Each sub-domain is known as an equivalence class.
- One sub-domain serves as a source for selecting one test input, any one input from each domain is good enough.
- All inputs from one sub-domain have the *same effect* in the program, i.e., output will be the same.
- We will do equivalence class partitioning in detail in the next lecture.

Equivalence class partitioning: Example

Consider a software system that computes income tax based on adjusted gross income (AGI) according to the following rules:

- If AGI is between $1 and $29,500, the tax due is 22% of AGI.
- If AGI is between $29,501 and $58,500, the tax due is 27% of AGI.
- If AGI is between $58,501 and $100 billion, the tax due is 30% of AGI.

Equivalence class partitioning: Example, contd.

We get five partitions as below:

- $1 \leq AGI \leq 29{,}500$: Valid input.
- $AG1 < 1$: Invalid input.
- $29{,}501 \leq AGI \leq 58{,}500$: Valid input.
- $58{,}501 \leq AGI \leq 100$ billion: Valid input.
- $AGI > 1$ billion: Invalid input.

Five test cases, each containing one number for AGI in the above range will suffice for testing the tax requirement based on AGI.

Boundary value analysis

- Boundary Value Analysis (BVA) is about selecting test inputs near the boundary of a data domain so that the data both within and outside an equivalence class are selected.
- BVA produces test inputs near the boundaries to find failures caused by incorrect implementation at the boundaries.
- Once equivalence class partitioning partitions the inputs, boundary values are chosen on and around the boundaries of the partitions to generate test input for BVA.
- Programmers often make mistakes at boundary values and hence BVA helps to test around the boundaries.

Guidelines for BVA

- The partition specifies a range: Construct test cases by considering the boundary points of the range and the points just beyond the boundaries of the range.
- The partition specifies a number of values: Construct test cases for the minimum and the maximum value of the number. In addition, select a value smaller than the minimum and a value larger than the maximum.
- The partition specifies an ordered set: Consider the first and last elements of the set.

BVA: An example

Consider the AGI and the five partitions that were identified for equivalence class partitioning.

BVA test cases for the partitions will be as follows:

- $1 \leq$ AGI $\leq 29{,}500$: BVA values will be 0, 1, -1, 1.5, 29,499.5, 29,500, 29,500.5.
- AG1 < 1: BVA values will be 0,1,-1, -100 billion.
- $29{,}501 \leq$ AGI $\leq 58{,}500$: BVA values will be 29,500, 29,500.5, 29,501, 58,499, 58,500, 58,500.5, 58,501.
- $58{,}501 \leq$ AGI ≤ 100 billion: BVA values will be 58,500, 58,500.5, 58,501, 100 billion, 101 billion.
- AGI > 1 billion: BVA values will be 100 billion, 101 billion, 10000 billion.

Decision tables

- Equivalence partitioning considers each input separately, we cannot combine conditions.
- Decision tables handle multiple inputs by considering different combinations of equivalence classes.
- Very popular to test several different categories of software.

Decision table

		Rules or Combinations							
Conditions	Values	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8
C_1	Y, N, –	Y	Y	Y	Y	N	N	N	N
C_2	Y, N, –	Y	Y	N	N	Y	Y	N	N
C_3	Y, N, –	Y	N	Y	N	Y	N	Y	N
Effects									
E_1		1		2	1				
E_2			2	1			2	1	
E_3		2	1	3		1	1		

Decision table

A decision table has

- A set of conditions and a set of effects arranged in a column.
- Each condition has possible value (yes (Y), no(N), don't care(–)) in the second column.
- For each combination of the three conditions there are a set of rules from R_1 to R_8. Each rule has a Y/N/– response and contains an associated set of effects (E_1, E_2 and E_3).
- For each effect, an effect sequence number specifies the order in which the effect should be carried out if the associated set of conditions are satisfied. For e.g., if C_1 and C_2 are true but C_3 is not true, then E_2 should be followed by E_3. combinations the decision table represents.

Decision table: Example

Conditions	Step 1	Step 2	Step 3	Step 4
Repayment amount has been	Y	Y	N	N
mentioned	Y	N	Y	N
Terms of loan				
has been mentioned				
Effects	Y	Y	–	–
Process loan amount	Y	–	Y	–
Process term	–	–	–	Y
Error message				

Random testing

- In random testing, test inputs are selected randomly from the input domain.
- Example: Consider computing $sqrt(x)$, where x is an integer that takes values from 1 to 10^8 with equal likelihood and that the result must be accurate to within 2×10^{-4}.
- To test this program, one can generate uniformly distributed pseudo-random inputs t and obtain outputs z.
- For each t, we compare outputs z and z^2 and compare z^2 with t. If any of the outputs fails to be within 2×10^{-4} of the desired results, the program must be fixed and test repeated.

COURTESY:MEENAKSHI DSOUZA,IIIT ,BANGLORE