UNIVERSITAS
GADJAH MADA

Blackbox Testing
Equivalence Classes & Boundary Values Analysis

Margareta Hardiyanti, S.Kom., M.Eng

Blackbox testing

Blackbox testing

Black-Box refers to the software which is treated as a black-box. The source code is not checked at all.

The test engineer unawares of how the inputs are transformed into outputs by the software.

Equivalence Classes : Background

1. Specifications and designs sometime partition the set of all possible inputs that receive equivalent treatment.
2. An input domain may be too large for all its elements to be used as test input. However, the input domain can be partitioned into a finite number of subdomains for selecting test inputs.

Equivalence Classes : Steps to do

1. Identifying the equivalence classes
2. Remove overlapping Equivalence Classes (EC)
3. Defining the test cases.

Equivalence Classes

Each subdomain is known as an equivalence class (EC). It aims to divide the input domain of the system under test into classes/groups of inputs.

An EC is a set of inputs that the system treats identically when the system is tested.

Equivalence Classes : Guidelines (1)

An input condition specifies a range [a,b] :

Identify one EC for $a \leq X \leq b$

and

two other classes for $X < a$ and $X > b$ to test the system with invalid inputs.

For example :

The item count can be from 1 to 999

Identify the EC!

Equivalence Classes : Guidelines (2)

An input condition specifies the number of N valid values

Create one EC for the correct number of inputs

and

two ECs for invalid inputs

and

one for less than N values and one for more than N values.

For example :
The input condition specifies that the user should input exactly three preferences
Identify the EC!

Equivalence Classes : Guidelines (3)

An input condition specifies a set of values and each is handled differently

Create one EC for each element of the set

and

one EC for an invalid member or element outside the set.

For example : type of simaster user must be STUDENT, LECTURER and STAFF

Identify the EC!

Equivalence Classes : Guidelines (4)

An input condition specifies a "must-be" value

Create one EC for a must-be value

and

one EC for something that is not a must-be value.

For example : First character of the identifier must be a letter

Identify the EC!

Equivalence Classes : Guidelines (5)

The input condition specifies that an input is a collection of items, and the collection can be of varying size;

for example, a list or set of elements;

define :

one valid equivalence class for a collection of size 0,

one valid equivalence class for a collection of size 1,

and

one valid equivalence class for a collection of size > 1.

For example : The user should input a list of hobbies. Define the EC!

Equivalence Classes

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 36% of AGI.

Define the EC

UNIVERSITAS
GADJAH MADA

Equivalence Classes : Benefit

Equivalence Classes : Benefit

- A small number of test cases are needed to adequately cover a large input domain.
- The probability of uncovering defects with the selected test cases based on EC partitioning is higher than that with a randomly chosen test suite of the same size.
- The EC partitioning approach is not restricted to input conditions alone; the technique may also be used for output domains.

Equivalence Classes : Test Case

In the EC partition technique, a single test input is arbitrarily selected to cover a specific EC. It’s required to generate specific test input, either inside or outside of the defined EC partitions.

To identify the test inputs, there is known a technique named boundary value analysis, which focuses on the boundary of the ECs.

Boundary Values Analysis : Background

Requirement :

Mahasiswa semester 6 ke atas dapat mengikuti program MBKM

Boundary Values Analysis : Background

```
if (semester > 6)
then
…
else
…
```

Boundary Values Analysis : Background

Programmers usually are not able to decide whether they have to use <= operator or < operator when trying to make comparisons.

Different terminating conditions of for-loops, while loops, and repeat loops may cause defects to move around the boundary conditions.

The requirements themselves may not be clearly understood, especially around the boundaries.

Boundary Values Analysis

Boundary Values Analysis :

Boundary-value analysis is both a refinement of input partitioning and an extension of it.

It aims to select test cases to explore the boundary conditions of a program.

Boundary Values Analysis

Boundary conditions are those situations at the edge of the planned operational limits of the software.

Boundary Values Analysis : Items

- Numeric
- Speed
- Character
- Location
- Position
- Size
- Quantity

Boundary Values Analysis

And, think about the following characteristics of those types:

- First/Last
- Start/Finish
- Empty/Full
- Slowest/Fastest
- Largest/Smallest
- Next-To/Farthest-From
- Min/Max
- Over/Under
- Shortest/Longest
- Soonest/Latest
- Highest/Lowest

Boundary Values Analysis : Guidelines (1)

When presented with a boundary condition, always test:

- *the valid data just inside the boundary*
- *the last possible valid data,*

and

Test the invalid data just outside the boundary:

- *adding one, or a bit more, to the maximum value, and*
- *subtracting one, or a bit more, from the minimum value.*

For example:

- A student dormitory specify that a housing unit can be shared by one to four students
- A text entry field allows 1 to 255 characters

Boundary Values Analysis : Guidelines(2)

If the input or output of a program is an ordered set (a sequential file, for example, or a linear list or a table), focus attention on :

- *the first,* and
- *last elements of the set.*

Equivalence Classes x Boundary Analysis

McDonald can only process up to 10 ordered burger

The use of this strategy will not guarantee that all errors will be found, but it has been found to represent a reasonable compromise.

Boundary Analysis

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 36% of AGI.

Define the test case from Boundary Analysis!

Thank you