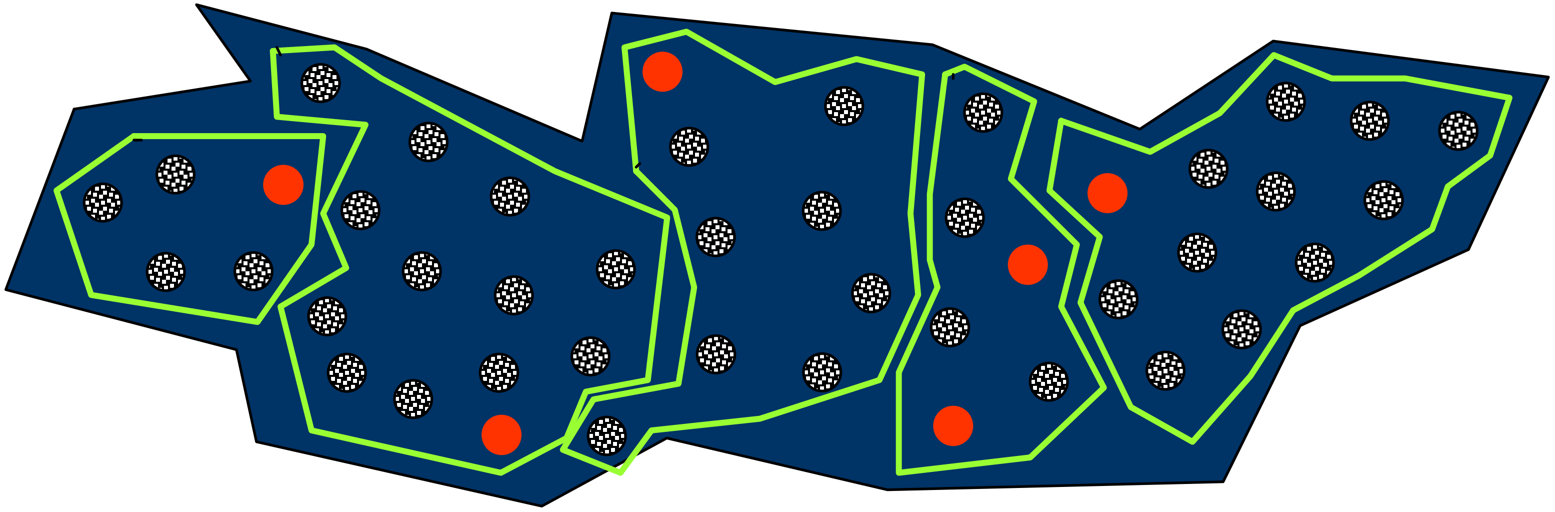
DOMAIN TESTING



Ref: Cem Kaner & James Bach

Domain Testing

- EQUIVALENCE partitioning, equivalence analysis, boundary analysis
- Fundamental question or goal:
  - This confronts the problem that there are too many test cases for anyone to run. This is a stratified sampling strategy that provides a rationale for selecting a few test cases from a huge population.

In domain testing, we partition a domain into sub-domains (equivalence classes) and then test using values from each sub-domain.

General approach

- Divide the set of possible values of a field into subsets, pick values to represent each subset. The goal is to find a "best representative" for each subset, and to run tests with these representatives. Best representatives of ordered fields will typically be boundary values.
- *Multiple variables*: combine tests of several "best representatives" and find a defensible way to sample from the set of combinations.

What is equivalence?

- 4 views of what makes values equivalent. Each has practical implications
  - ***Intuitive Similarity***: two test values are equivalent if they are so similar to each other that it seems pointless to test both
  - ***Specified As Equivalent:*** two test values are equivalent if the specification says that the program handles them in the same way
  - ***Equivalent Paths:*** two test values are equivalent if they would drive the program down the same path (e.g. execute the same branch of an IF)
  - ***Risk-Based***: two test values are equivalent if, given your theory of possible error, you expect the same result from each

Equivalence Partitioning

- If a test case in an equivalence class detects an error, then any other test case in that class should also detect the error.
- If a test case in an equivalence class does not detect an error, then any other test case in that class should not detect an error.

What should we do ?

- Program specifications should identify both valid and invalid inputs for a program.
  - VALID EQUIVALENCE CLASSES are chosen to represent valid inputs.
  - INVALID EQUIVALENCE CLASSES are chosen to represent invalid inputs.

Test which values from the equivalence class?

- The program is more likely to fail at a boundary?
  - ***Suppose program design***:

INPUT < 10	result: Error message
10 <= INPUT < 25	result: Print "hello"
25 >=INPUT	result: Error message

  - ***Some error types***
    - Program doesn't like numbers
      - Any number will do
    - Inequalities mis-specified (e.g. INPUT <= 25 instead of < 25)
      - Detect only at boundary
    - Boundary value mistyped (e.g. INPUT < 52, transposition error)
      - Detect at boundary and any other value that will be handled incorrectly

Boundary or Non-Boundary ?

- Boundary values (here, test at 25) catch all three errors
- Non-boundary values (consider 53) may catch only one of the three errors

Guidelines

- If an input condition specifies a range of values (e.g., “the item count can be from 1 to 999”), identify one valid equivalence class (1<item count<999) and two invalid equivalence classes (item count<1 and item count>999).

Guidelines

- If an input condition specifies the number of values (e.g., “one through six owners can be listed for the automobile”), identify one valid equivalence class and two invalid equivalence classes (no owners and more than six owners).

Guidelines

- If an input condition specifies a set of input values and there is reason to believe that each is handled differently by the program
  - Exp “type of vehicle must be BUS, TRUCK, TAXI-CAB, PASSENGER, or MOTORCYCLE
  - Identify a valid equivalence class for each and one invalid equivalence class (e.g. “TRAILER”).

Guidelines

- If an input condition specifies a "must be" situation (e.g., "first character of the identifier must be a letter"), identify one valid equivalence class (it is a letter) and one invalid equivalence class (it is not a letter).

Guidelines

- If there is any reason to believe that elements in an equivalence class are not handled in an identical manner by the program, split the equivalence class into two or more smaller equivalence classes

Which one ?

- A member of an equivalence class is a best representative (relative to a potential error) if no other member of the class is more likely to expose that error than the best representative.
  - Boundary values are often best representatives
  - We can have best representatives that are not boundary values
  - We can have best representatives in non-ordered domains

Selecting test cases

- Choose at least one test case from each class identified.
- For valid classes, choose test cases to cover as many equivalence classes as possible, until all valid classes have been covered.
- For invalid classes, choose test cases so that each covers one and only one invalid class, until all classes are covered.

Strengths Vs Weaknesses

- Strengths
  - Find highest probability errors with a relatively small set of tests.
  - Intuitively (trực giác) clear approach, easy to teach and understand
  - Extends well to multi-variable situations
- Blind spots or weaknesses
  - Errors that are not at boundaries or in obvious special cases.
  - Also, the actual domains are often unknowable.
  - Reliance (tin tưởng) on best representatives for regression testing leads us to overtest these cases and undertest other values that were as, or almost as, good.